SANDIA REPORT
SAND2022-3755
Printed March 2022

Sandia National Laboratories

A Review of Technologies that can Provide a ‘Root of Trust’ for Operational Technologies

Michael Rowland, Benjamin Karch

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

ABSTRACT

The supply chain attack pathway is being increasingly used by adversaries to bypass security controls and gain unauthorized access to sensitive networks and equipment (e.g., Critical Digital Assets). Cyber-attacks targeting supply chain generally aim to compromise the environments, products, or services of vendors and suppliers to inject, add, or substitute authentic software and hardware with malicious elements. These malicious elements are deemed to be authentic as they arise from the vendor or supplier (i.e., the supply chain). This research aims at providing a survey of technologies that have the potential to reduce exposure of sensitive networks and equipment to these attacks, thereby improving tamper resistance.

The recent advances in the performance and capabilities of these technologies in recent years has increased their potential applications to reduce or mitigate exposure of the supply chain attack pathway. The focus being on providing an analysis of the benefits and disadvantages of smart cards, secure tokens, and elements to provide root of trust. This analysis provides evidence that these roots of trust can increase the technical capability of equipment and networks to authenticate changes to software and configuration thereby increasing resilience to some supply chain attacks, such as those related to logistics and ICT channels, but not development environment attacks.

ACKNOWLEDGEMENTS

The authors would like to thank Shannon Eggers, Idaho National Laboratory, and Alexandria Haddad, Sandia National Laboratories, for their critical reviews and insights.

This research was funded by the U.S. Department of Energy Office of Nuclear Energy Cybersecurity research and development program under milestone M3CT-22SN1105013.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
3GPP	3rd Generation Partnership Project
AES	Advanced Encryption Standard
AID	Application Identifier
ALC	Application Load Certificate
ALU	Application Load Unit
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAP	Converted Applet
CC	Common Criteria
CPU	Central Processing Unit
CRTM	Core Root of Trust for Management
DLL	Dynamic-link Library
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electronically Erasable Programmable Read-Only Memory
EK	Endorsement Key
ENISA	European Union Agency for Cybersecurity
EPRI	Electric Power Research Institute
ETSI	European Telecommunications Standards Institute
FPGA	Field Programmable Gate Array
GSM	Global System for Mobiles
HCE	Host Card Emulation
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IDE	Integrated Development Environment
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
IT	Information Technology
JCOP	Java Card Open Platform
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
MAC	Message Authentication Code
MEL	MULTOS Executable Language

Abbreviation	Definition
NEI	Nuclear Energy Institute
NFC	Near Field Communications
NPP	Nuclear Power Plant
NRC	Nuclear Regulatory Commission
OEM	Original Equipment Manufacturer
OS	Operating System
OT	Operational Technology
PCR	Platform Configuration Register
PIN	Personal Identification Number
PLC	Programmable Logic Controller
RAM	Random Access Memory
RFID	Radio Frequency Identification
ROM	Read Only Memory
RSA	Rivest Shamir Adleman
SCP	Smart Card Platform
SIM	Subscriber Identification Module
SPI	Serial Peripheral Interface
SSL	Secure Sockets Layer
TAM	Technical Assessment Methodology
TCG	Trusted Computing Group
TLS	Transport Layer Security
TPM	Trusted Platform Module
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunications Service
USIM	UMTS Subscriber Identification Module

1. INTRODUCTION

Cybersecurity for Nuclear Power Plants (NPP) has been a major focus of regulators, operators, and suppliers around the world for decades. The release of the US Nuclear Regulatory Commission's (NRC) Title 10 of the Code of Federal Regulations (CFR) Part 73.54 [1], required all NPP licensees to provide a cybersecurity plan to the NRC for review and approval. The objective of the regulation is for each NPP licensee to provide high assurance that digital computer and communication systems and networks are adequately protected against cyber-attacks, up to and including the Design Basis Threat (DBT).

As organizations have become more capable at defending their networks from direct attacks, adversaries have exploited the supply chain to gain unauthorized access. A European Union Agency for Cybersecurity (ENISA) report titled "ENISA Threat Landscape for Supply Chain attacks" noted that "supply chain attacks increased in number and sophistication in the year 2020 and this trend is continuing in 2021, posing an increasing risk for organizations. It is estimated that there will be four times more supply chain attacks in 2021 than in 2020." [2]

Supply Chain Risk Management (SCRM), at its core, involves risk transfer agreements and verification. The transfer of risk to the organization most able, or best positioned, to manage the risk is a prudent decision for the acquirer to make. However, supply chains are complex and have many interdependencies. One of the key attributes in supply chain relationships is "trust." Trust of suppliers' products, key components, software, and services is a critical component for NPP operators. They rely on these systems to maintain, configure, and/or monitor plant systems and ensure operational performance.

This report investigates technologies used in other industries such as telecommunications, personal computing, and distributed systems that provide some sort of "trust anchor," or root of trust, which provide a verifiable basis to validate trust in products and outcomes of services. Leveraging root of trust could provide tamper-resistance to the supply chain both from Information and Communications Technology (ICT) and logistics based attacks [3].

The following sections provide definitions and context of supply chain security approaches used currently by the domestic nuclear power industry. This report aims to enhance the abilities of NPP operators to autonomously detect malicious or unintended changes to digital components or subcomponents, thereby increasing confidence in trust relationships that span the supply chain.

1.1. Supply Chain

Supply chain considerations for Operational Technology (OT), specifically nuclear power, come with fundamental differences than the software (Information Technology (IT)) supply chain. The most obvious difference is that OT devices control physical (and often safety critical) processes. Generally, these processes are controlled by Programmable Logic Controllers (PLCs) or Field Programmable Gate Arrays (FPGAs). With physical devices there are additional opportunities for tampering because of multiple exchanges of personnel responsible for the equipment after manufacturing and before final operations.

The supply chain is often defined, for example by ENISA [2], as two endpoints: the supplier and its assets, and the customer and its assets. This is useful when considering the supply chain for general products or services but should be expanded for OT. Expanded definitions can be found in more specialized documentation, such as the Electronic Power Research Institute (EPRI) Technical Assessment Methodology (TAM) [4]. These expanded definitions add the important distinction that

OT devices require a stage of integration. Integration in this context involves a third party who designs a larger system, with multiple components, to control some physical process like coolant flow. For this paper, the supply chain will be considered at all stages prior to the device's final operational environment. This includes manufacturing, shipping, and integration, as well as maintenance or upgrade activities, such as firmware updates, where reliability cannot be reasonably placed on an operator.

1.2. Current Domestic Industry Approach

Following 10 CFR 73.54 [1], the NRC published Reg Guide 5.71 [5] identifying one acceptable approach meeting the requirements of the regulation. The Nuclear Energy Institute (NEI), developed its own guidance document NEI 08-09 [6] Revision 6, which was approved by the NRC to be another acceptable approach [7]. While both guides address cybersecurity concerns in a comprehensive manner, it is the supply chain considerations in NEI 08-09 Rev 6 that are the most applicable to this report.

Currently, there are supply chain security controls provided by NEI 08-09 Addendum 3, in which the licensee has responsibility within the supply chain [8]. These controls are applicable for a device's lifecycle stages between factory acceptance testing (FAT) through decommissioning. A device may be compromised earlier in its lifecycle where the licensee is both unable to mitigate or detect such a compromise, as previously mentioned. Additionally, Addendum 3 does not provide security controls at the subcomponent level, i.e., there is no verification of factors such as included software libraries or integrated chips (IC) on the serial bus.

The EPRI technical report 'Cyber Security in the Supply Chain,'[9] which integrates the EPRI TAM[4], outlines supplier questionnaires, cyber-related procurement language, secure product transitions, supplier certifications, testing, configuration management, and installation considerations for procurement based on the type of device [9]. This document establishes a risk-informed supply chain, but may not be representative of all risks or attack vectors within the supply chain [8].

1.3. Examples of Supply Chain Attacks

A recent and high impact example of a supply chain attack is the attack on SolarWinds Orion, depicted in Figure 1 (adopted from [10] and [8]). SolarWinds is an American company offering enterprise networking products and solutions. Orion is a SolarWinds network management system. In December 2020, FireEye discovered that a supply chain attack had compromised SolarWinds Orion to distribute malware [11]. Victims received a digitally signed update, which gave the attackers backdoor access to the victims' devices. This access allowed the attackers lateral movement into the victims' system and subsequent data theft. Because the compromise occurred at the supplier, and victims received a digitally signed update, there was little to no detection of the compromise to the system(s). The attack was eventually discovered by advanced analysis of the software and its behavior.

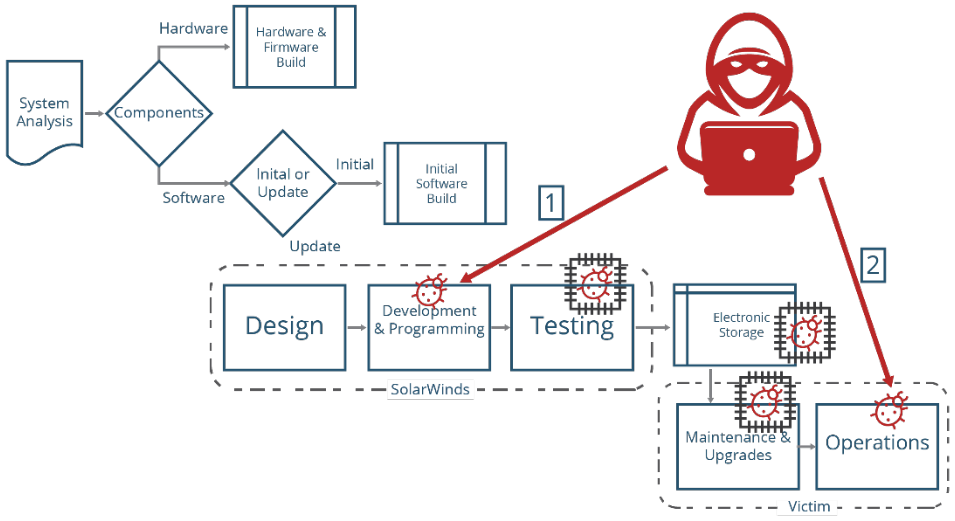

Figure 1 SolarWinds Orion Infection [8, 10]

Another example of a supply chain attack is the Stuxnet attack on the Natanz Iranian nuclear facility. Stuxnet was a highly selective, large, and sophisticated malware computer worm which attacked PLCs in the facility, causing physical damage. Once introduced, Stuxnet replicated through the OT network finding a vulnerable computer connected to a PLC. Stuxnet then replaced the PLC's Dynamic-Link Library (DLL) responsible for writing compiled code to the PLC, with a malicious DLL performing a man in the middle attack on the code in transit [12]. This attack on communications to the PLC is shown in Figure 2. The outcome of the attack was malicious software substitution made to the target device (PLC), causing speed fluctuations in a centrifuge damaging and/or destroying the centrifuge. Most likely, the malware was initially introduced to the network via a USB memory device [13], either by during routine maintenance or by a malicious insider. This attack would be considered a supply chain attack if the malicious code was introduced during maintenance on a device.

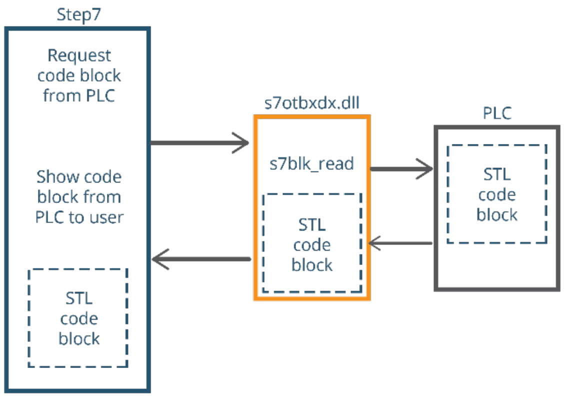

Figure 2 Stuxnet Man-in-the-Middle [12]

Additionally, there is the growing issue of counterfeit hardware. Counterfeit Integrated Circuits (ICs) have grown more prevalent and common in the supply chain in recent years, with reported counterfeit parts quadrupling between 2009 and 2011 [14]. Serious consideration should be given to counterfeit devices, especially as operational environments utilize digital components to administer safety critical systems. Counterfeiting may come from a different motivation than the strictly malicious attacks noted above, but it still brings potential for devastating consequences. A counterfeit may simply lead to premature end of life for a device, reduction of design margins (e.g., robustness, resilience), causing unexpected behavior, or it could potentially include a malicious code, such as a backdoor or logic bomb, intentionally placed by an adversary.

2. ROOT OF TRUST DESCRIPTION

It is not practical nor efficient for plant operators to disassemble and manually inspect each piece of digital equipment as it is delivered. Plant operators do not have the capability or competence to perform the advanced and sophisticated tasks related to in-depth cyber security inspection. Also, simple inspections would not protect against a malicious substitution, or insertion, that occurs during the lifetime of the device in its operational environment. There are multiple proposed solutions which involve monitoring traffic to and from devices, but these rely on an implicit trust in the device's initial state or an additional hardware component (see Appendix B.2). Thus, the device must perform some reporting on its configuration. However, there is potential of device compromise at the time of reporting. To guarantee the device's reports are genuine, there must be some sort of trust embedded in, and bound irreversibly, to the device that is tamper resistant and capable of performing common cryptographic functions that may occur in an encryption or authentication procedure. A "root of trust" is a tamper-resistant element in a digital system that can always be trusted, and therefore can be depended on as the root of all trusted operations. The core protection of trust provided by the root of trust is the secure (i.e., tamper resistant) storage of a private key (i.e., a secret that is not shared) which can be used to manage other generated keys and sign data to be sent out of the secure element.

The root of trust protection involves asymmetric cryptographic mechanisms that involves a public key which is shared widely and openly and a private key that is generated on the root of trust and never leaves the root of trust. Asymmetric cryptographic mechanisms can provide for protection of confidentiality and integrity requirements, as well as provide authentication and non-repudiation (via digital signature standards/algorithms) as the root of trust is the only component that has access to the unique private key.

The root of trust is not a Trusted Execution Environment (TEE), though it is an essential building block for creating one [15]. The root of trust for a system's secure boot is often considered to be the Read Only Memory (ROM) which stores the initial stages of the booting process. For this paper, the root of trust is considered as separate ICs which could either be embedded onto the host device's serial bus or communicated with through some other means. The root of trust component also requires a trustworthy manufacturer to produce the private key, which must never leave the secure element's non-volatile memory, along with a certificate attesting to the fact that the secure element's manufacturer assigned that key to that unit. This general process is common across industries, being used for Transport Layer Security (TLS) and Secure Socket Layer (SSL) certificates, telecommunications, and secure credit card transactions. Though digital signatures and certificates are quite mature and well accepted in many industries, it can be difficult to achieve a consensus on a particular public key infrastructure (PKI) scheme.

2.1. What Can be Protected

It is important to note that automated or semi-automated detection of all supply chain attacks, or irregularities, is not feasible given the current state of the art in technology. At some point, in any given system, there must be some human trust; a trusted person with sufficient authority may change a product in a malicious manner that goes undetected until utilized. This section (summarized in Table 1) outlines the protections that are feasible offerings for a root of trust. The SolarWinds Orion attack mentioned previously was not feasible for a customer to detect because the supplier failed to detect the compromise internally, and subsequently provided the customer with a product which could be cryptographically verified. For this reason, the compromise was able persist for months before it was noticed. It is possible that an Intrusion Detection System (IDS) upstream may detect the command-and-control behavior of the embedded backdoor, but this would be a post-intrusion detection. There is also potential for the supplier to implement more stringent procedures for authorization and accountability for product changes; confidence can be given to these procedures using a root of trust.

Table 1 Summary of Protection (Adapted from [8])

Attack Type	Description	Root of Trust Protection Confidence	Protection Method
Theft of IP, design, or data	Unauthorized disclosure of information from a stakeholder who has a trust relationship with the end target, enabling future attacks and/or causing economic loss. This may include but is not limited to intellectual property (IP), design information, operational / configuration data, or stored secrets (i.e., private key, digital certificates).	Low	Full disk encryption using stored keys on the secure element will help to prevent unauthorized access to device information. A supplier may implement access control and auditing to prevent unauthorized access while a device is in development / manufacturing stages. A secure element will not prevent an authorized user from maliciously leveraging their authorized access.
Malicious substitution	Complete replacement of digital technology, including hardware, firmware, and/or software. Hardware clones or counterfeits may not impact all end users depending on the distribution, whereas a substituted software package may compromise all end users even if only a few were targeted.	High	A root of trust will be able to provide the operator with real time trusted information on the devices hardware, firmware, and software, making unauthorized substitutions detectable. A complete hardware swap will not report correctly or at all without the secure element, as any report will not be able to be signed with a key which may be confirmed through a certificate issued by the manufacturer.
Design, specification, or requirements alteration	Unauthorized modification of design, specifications, or requirements that compromises the design stages and results in the purposeful inclusion of latent design deficiencies (e.g., requirements that result in vulnerabilities) or built-in backdoors.	Low	Like theft of IP, design or data, a supplier may implement an access control or auditing system that utilizes root of trust technologies, but an authorized change made during design stages will lead to a trusted but insecure state at the customer.

Attack Type	Description	Root of Trust Protection Confidence	Protection Method
Development, build, or programming tool alteration	Unauthorized modification of the development environment, including platform, build and programming tools, with the intent to corrupt the device under development.	Medium	Code should be signed by the supplier, and then verified on the end device. Alterations that attempt to alter code after it has been signed will be detected. Though, a supplier with bad security practices may improperly implement signing procedures and sign code after it has been manipulated. Additionally, any tools / programs (e.g., a compiler) used to assist in development can also be signed and verified by the supplier.
Malicious insertion	Addition or modification of information, code, or functionality directly into a device to cause malicious intent, such as impairing or altering device operation or function.	High	Similar to malicious substitutions, a malicious insertion during the logistics and ICT transfer to the customer or during operation would be detected by system. Authorized and approved malicious insertions directly from the developer or OEM would be part of the approved component.
Tampering, configuration manipulation	Unauthorized alteration or fabrication of configuration, non-executable data, or sending of unauthorized commands with the goal of impacting device operation or function.	Medium/High	Changes to non-executable data that is measured and reported will be detected, though, it is likely not feasible to measure all data on the device. Properly defining which files and data is important to verify will allow for this information to be measured and reported. A root of trust also allows for the device to send commands securely (encrypted and authenticated) to other devices in the system.

Stuxnet, on the other hand, could have been reasonably prevented at two different points. The first point was when the malicious DLL file which was loaded by PLC configuration software. This file would have to be different from the standard DLL which comes packaged with the software. An integrity check would have found this to be either a malicious or corrupted version. The error could also have been detected using an integrity check between the PC and the PLC. The DLL changes the data written to the PLC while in transit, so there must be some discrepancy between the digest of the data that the user at the PC sees and the data that the PLC receives.

It's also reasonable to suppose that clones/counterfeits of devices could be detected. In the same manner, it may be possible to detect malicious substitutions or insertions of other microchips onto a device. However, this may require an unreasonable amount of computational time or power to be done by a central processor on low power devices used in OT. Malicious or invalid firmware installations can be detected during the boot of the device, protecting against software-based attacks after the device leaves the Original Equipment Manufacturer (OEM) and before reaching the operational environment. A secure and trusted boot procedure would also maintain that the device does not receive an improper firmware installation at any point during its lifetime, given that the manufacturer does not attest to the firmware's authenticity and reliability.

It should be noted that it is possible for a supplier to implement a system using root of trust/smart card technology to avoid unauthorized changes to their code base, product design, specifications, etc. For example, any change made to the code base could require a user to present a smart card and enter an associated Personal Identification Number (PIN). This would mean that it would be highly unlikely for an adversary to successfully make a malicious insertion, deletion, or substitution without first becoming or compromising a trusted insider. This is, however, outside of the scope of this paper.

In summary, an attack that takes place at any point in the supply chain after design can be detected by the operator using root of trust. Design errors and malicious design cannot be expected to be detected in a timely manner at the endpoint. There must exist some vetting process between end-user and supplier in which the supplier is trusted to perform regular security assessments which would prevent malicious insertion of vulnerabilities or backdoors into the hardware and software of a device.

2.2. Trends in Root of Trust Technology

The most typical forms of root of trust are smart cards and Trusted Platform Module (TPM) chips. Smart cards have grown immensely in popularity over recent years because they are secure, programmable, and upgradable [16]. Smart cards allow for a flexible implementation that is still able to provide security and a root of trust. These cards come in many form factors but rely on the same technology of an embedded microcontroller within card stock. A form factor many consumers are familiar with is modern credit cards, which are used as a root of trust to prove that the issued card was present at the time of a given transaction. Smart cards often utilize an IC with pins like the high-level pinout diagram illustrated in Figure 3. 0 includes several detailed descriptions of current root of trust technologies.

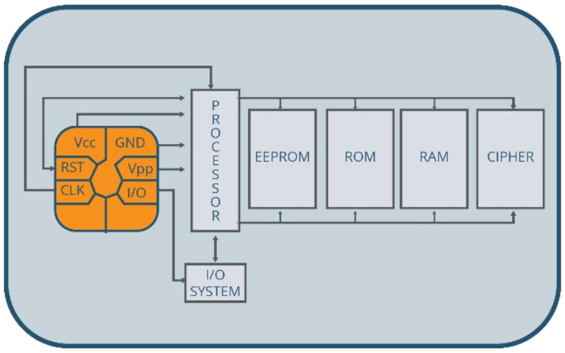

Figure 3 Smart Card Architecture [17]

The pins on the left side of Figure 3 are used to exchange information with the host computer. In the microcontroller, the smart card contains a similar component to that of a typical system on a chip. The card's operating system is stored within the chip's ROM. The ROM cannot be changed, as it is physically hard wired into the circuit. Unlike the operating system, smart card applications are stored in electronically erasable read only memory (EEPROM). This allows cards to be updated with additional functionality without issuing a new card each time an update or change is made. Smart card chips also contain Random Access Memory (RAM) for fast memory access during computation and dedicated crypto processors to speed up cryptographic operations such as symmetric key generation and stream cipher encryption. Smart cards create a viable root of trust by implementing secure key storage. Keys are flashed onto the card at the time of manufacturing and cannot be subsequently changed or read without an invasive procedure which is likely to render the attacked card unusable [18].

Smart card hardware can be utilized to accomplish many different applications. The most familiar of these is contact cards, in which the chip is exposed. The contacts of the smart card chip in Figure 3 physically meet a smart card reader to exchange. A card may also be implemented with a smaller physical footprint for the purpose of embedding the card in a larger device as a subscriber identification module (SIM). Commonly used in telecommunication devices, SIM cards reliably identify and store information related to devices [19]. Cards may also contain a hidden secure element under the card stock which requires no physical contact and communicates over some wireless protocol, likely Near Field Communication (NFC). These are commonly referred to as Radio-Frequency Identification (RFID) cards and may also be emulated by an NFC capable device such as an Android phone through Host Card Emulation. Manufacturers tend to be more likely to embrace contactless card technology, or at least implement dual-interface (both contact and contactless) functionality into the smart card in many use cases, such as finance or identification. Contact smart cards are growing in popularity and there continues to be more diversification of form factors for smart cards as the technology evolves [20].

Table 2 outlines the improvements made in smart card performance, power, and security over the decade from 1996 to 2016. These improvements are largely attributed to the use of smart card technology in SIM cards, which greatly increased demand for a powerful smart card in a small and affordable form factor. The speed of the processor has also made asymmetric cryptographic operations feasible on a large scale and has already been leveraged by subsequent generations of mobile telecommunications technologies (see Appendix A.3). Additionally, increases in storage capacity allow the cards to maintain thousands of keys of historical transactions. For these reasons, smart cards are becoming increasingly common for the purposes of securing Internet of Things

(IoT) devices. These devices have a wide range of security requirements, which can be supported by multi-application smart card technologies. One use case for these include industrial devices, for example enabling secure monitoring and management solar panels using embedded MULTOS chips [21]. These advancements in smart card chip performance has enabled new functionalities and use cases to provide tamper resistant roots of trust to secure the Nuclear Power Industry's supply chain.

Table 2 Smart Card Chip Advancements 1996-2016 [20]

	1996	**2016**
Geometry	700 nm	65 nm
CPU	8 bit	32 bit
ROM	20 KB	200 KB
RAM (KB)	.5	8
EEPROM	8 KB	500 KB
Voltage (V)	5	1.5-5
Clock Speed	3.5 MHz	30 MHz
I/O Speed	19.2 kbps	2 Mbps
Crypto Speed (512 bit RSA with Chinese Remainder Theorem)	66 mS	4 mS
Security Certification	Rare	CC EAL5+

Regardless of form factor or operating system, smart cards are implemented as the server in a client-server relationship. This means the card cannot generate its own commands but must wait to receive a command from the host. Communications between the smart card and the host take the form of application protocol data unit (APDU) frames, and are defined in ISO/IEC 7816-4 [22], which is defined in multiple parts. Many of these are specifically for the use case of identification and are not necessarily applicable for OT. However, the APDU data exchange, detailed in Table 3, is needed to enable standardized communications between the smart card and the host device. The standard also outlines the registration of application providers, i.e., the process in which an application ID is selected for use in the card. A card used in an OT use case should also follow card management procedures, which include interindustry commands for file management and managing the card's lifecycle. Card lifecycles are composed of multiple states, indicating important information in a smart card such as whether it has been initialized. Lifecycle states allow a card to be entered into an end-of-life state should the card or device be considered to have suffered an irreversible compromise or error. This end-of-life state prevents the card from being accessed or used ever again.

Table 3 APDU Command and Response Structure [22]

Field	Description	# of bytes	Direction
Command header	Class byte denoted CLA	1	To the card
	Instruction byte denoted INS	1	
	Parameter bytes denoted P1, P2	2	
L_c field	Encodes the number of bytes (N_c) in the command	0,1, or 3	
Command data field	Data passed to the card to be used in command processing	N_c	
L_e field	Encodes the number of expected bytes (N_e) in response	0,1,2, or 3	
Response data field	Response data	At most N_e	From the card
Response trailer	Status bytes denoted SW1, SW2	2	

Increased technological advancements have benefitted TPMs also; however, this evolution has resulted in fewer advancements than seen in smart cards since TPMs have a smaller number of features. The latest improvements in TPM technology come in the form of the TPM 2.0 family, which has introduced a significant increase in required cryptographic functions. This improvement comes as a direct response to the deprecation of the SHA-1 algorithm, which previous versions of the TPM relied on heavily. Increase in TPM adoption has also happened in recent years, and a TPM 2.0 compliant module are now a system requirement in order to run Microsoft Windows™ 11 [23].

Overall, secure elements have been and continue to be trending towards increased functionality and power. Historically used in telecommunications and finances, the technology is now more suited than ever for a wide range of use cases. Market trends confirm this, as secure element technology is growing significantly in industrial and automotive IoT use cases [24].

This page left blank

3. TECHNOLOGY COMPARISON

In the previous section, an overview of root of trust technologies was presented. This section (summarized by Table 4) evaluates the potential of these technologies to act as a root of trust specifically in the context of an OT environment. Chiefly, a given root of trust should provide a level of security over potential supply chain attacks that involve both physical and software-based malice. For this section, the subject of the root of trust will be a PLC, which is a common OT component used to process digital signals from sensors or similar assets and then control some physical process, such as the opening and closing of a valve. PLCs typically are implemented with a Linux based Real-time Operating System.

Table 4 Technology Comparison

Technology	Features/Description	Security
TPM	• Common Criteria (CC) Evaluation Assurance Level (EAL)[1] of EAL4+ (Methodically Designed, Tested and Reviewed) • Wide range of hardware implementations are accepted • Secure key storage and easily accessible cryptographic functions • Ownership is required to enable full functionality • Functionality limited to set of instructions defined by TCG or additional implemented by manufacturer	Medium/High
Java Card	• CC EAL 5+ (Semiformally Designed and Tested) • Implemented as secure IC • Secure key storage and easily accessible cryptographic functions • Multi-application OS capable of loading and installing custom implementation-specific applications • Application development process is easy with many free options for IDEs	High
MULTOS	• CC EAL 5+ (Semiformally Designed and Tested) • Secure key storage and easily accessible cryptographic functions • Implemented as secure IC • Secure key storage and easily accessible cryptographic functions • Multi-application OS capable of loading and installing custom implementation-specific applications • Application development process forces additional security procedures to ensure that applications are signed and verified	Very High

There are two technologies which are practical for use as a root of trust in an embedded device: smart cards and TPMs. For a detailed survey of all technologies reviewed during the development of this report, see Appendix A.

[1] CC methods and EALs are described in Section 3.3.

3.1. TPM

To provide hardware root of trust in devices with multiple components, often the chosen technology is TPMs, due to cost and availability. The TPM can provide the host system with secure key storage, and secure implementations of various cryptographic algorithms in a separate tamper resistant hardware module. Also, TPM 2.0 requires manufacturers to implement both Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC) asymmetric cryptographic mechanisms for a variety of key lengths and TPM also makes available secure non-volatile storage. This is important as it allows for the system to store information, such as keys that are assured to be stored in a tamper resistant module and stay persistent through a power cycle. Cryptographic functions are also available for applications running on the host device with TPMs. This allows any future application to take advantage of the secure cryptographic processor embedded into the TPM during integration with the platform. Additionally, the TPM does not require any application loading or updating, as it is deployed with a set list of functions to be utilized by the system's software. This also provides the added benefit that a TPM does not need to consider inter-application data leaking or manipulation, though this responsibility would be assumed by the host system.

A core concept in TPM security is ownership. It is assumed that a user would take ownership of the TPM for the necessary keys to be produced, thereby taking advantage of TPM capabilities such as remote attestation of a measured boot (security report digests taken at each boot stage of the boot process for review by an external entity). Allowing for multiple users, with TPM 2.0 multiple operators in the plant can authenticate to the TPM. However, this still presents all the challenges of password administration for each approved users, i.e., managing when there are changes in employment that require changing the device's owner. Additionally, the Trusted Computing Group (TCG) specifications for the TPM provide a lot of latitude for manufacturers to implement the TPM and its services in various ways.

In fact, the TPM is not necessarily even restricted to being implemented as an IC. The TCG also does not define how exactly the TPM must be manufactured to be tamper-resistant in or to what degree, only the fact that the TPM must be tamper-resistant. A listing of cryptographic functions such as hashing, symmetric and asymmetric encryption, and digital signature algorithms must be implemented, as specified by TPM, no guidance is provided on the security of the functions other than some recommended minimum key sizes. It could be argued that this flexibility gives manufacturers an opportunity for distinction, encouraging competition and innovation. However, this leaves quite a lot of responsibility on the consumer to make an informed decision on a particular manufacturer and integration method.

There has been some notable work done to show the possible attacks should a TPM be implemented incorrectly, particularly by Justin Boone and the NCC Group's TPM Genie [25]. As shown in Figure 4, the TPM Genie is a proof of concept showing a hardware man in the middle that eavesdrops and manipulates data between the host's Central Processing Unit (CPU) and TPM. One of the successful attacks posed is a man in the middle attack on the take ownership process. This then gives the interposer essentially full control over communications with the TPM. There are mitigations for this attack, the simplest is requiring the host to validate the certificate of the Endorsement Key (EK), used to initially establish a secure communication channel between the host and the TPM. There are also hardware manufacturing considerations that could be used to make such attacks impossible or at least quite expensive, requiring invasive modifications to the device's circuitry. Examples of this include hiding the communications bus between the components under a layer of silicon, or even embedding the TPM itself in the host device's CPU.

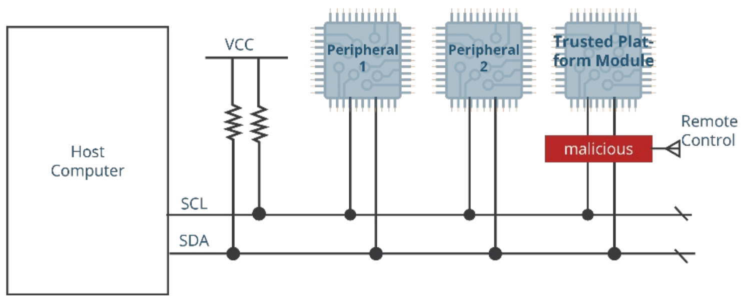

Figure 4 TPM Man-in-the- Middle [25]

Hardware attacks are typically not considered in the cybersecurity sector, under the assumption that if an adversary has physical access to a device, they will most certainly gain unfettered access. While this assumption may be valid, the correct use of a TPM can increase the duration and costs to overcome its protections. This report's objective is to provide assurance through the supply chain for OT assets, and therefore the core assumption of this effort is that an adversary will gain physical access to the device during supply chain activities for a period of time and this access must be protected against.

While there are hurdles which need consideration and mitigation for a TPM to provide sufficient security, it is still a viable candidate for such an application. A properly implemented TPM can provide some security assurances, such as boot attestation and full disk encryption. These procedures are often found in IT environments but lacking in a typical OT environment.

3.2. Smart Cards

Alternatively, the smart card technology discussed in the previous section can achieve the goal of a hardware root of trust. Smart cards are typically implemented as an IC embedded within a typical credit card form factor. A growing trend in smart card technology is integrating the smart card IC into a different form factor other than a card, such as an embedded chip on a larger board [20]. This is often done by wire bonding the chip and housing it in an IC's housing. There is also the addition of a serial interface for communications with the card over serial bus communications like Inter-integrated Circuit (I2C) or Serial Peripheral Interface (SPI). This would be done to avoid software overhead and therefore meet the performance impact of requiring other chips to conform to data framing and timing of ISO/IEC 7816. An example of this trend is the NXP A700x family [26], a chip meant to be included on a board with other chips and act as a secure authentication microcontroller. This chip family is capable of 100 kbit/s I2C communications, has 76 kB of EEPROM, and runs Java Card Open Platform (JCOP) 2.4.2 which is discussed in 0.

These smart card chips can provide the same basic cryptographic functions as a TPM, i.e., key generation, hashing, and digital signatures, as well as secure non-volatile memory storage. There is also not a necessary need for a user or group of users to take ownership of the smart card, as this is handled by the issuer. Like with a UMTS (Universal Mobile Telecommunications Service) Subscriber Identity Module (USIM), devices can be flashed with a secure unique identifier; this would enable for the general approach of the USIM's mutual authentication with the base station to be molded for the use case of protected OT systems.

This does require a central configuration manager to be in contact with each of the PLCs in the OT network to act as the base station which will authenticate the PLCs. Just as with the USIM, there is no need to assume that the configuration manager is an inherently trusted device. A similar protocol can be implemented in which a configuration manager must first authenticate itself to a PLC's embedded smart card, which will then authenticate to the configuration manager. After authentication, an encrypted secure communication channel is established in which measured boot information is transferred to verify the devices' health and state. Other issues include understanding physical considerations for this chip, just as with the TPM. For example, making the exposed serial bus connection between the host device CPU and the smart card chip as inaccessible as possible to a potential attacker, avoiding the possibility of a man in the middle attack on inter-chip communications.

These smart card chips apply to both MULTOS and Java Card devices, as there are options for both ecosystems available in this form factor. MULTOS is generally considered to be of a greater security than Java Card, as MULTOS forces additional security constraints, e.g., only allowing signed and verified application code to be uploaded onto the device. Typically, the downside to additional security measures is reduced ease of use. It is easier to develop for Java Card without these constraints, and there are also multiple free Java IDEs available which allow for writing custom applets for Java Cards.

There is also the possibility of using contactless smart cards, which have seen significant growth in recent years. The most common implementation of contactless smart cards is through NFC, though this does open potential security concerns, mainly of man in the middle attacks or eavesdropping. Since NFC communication is a wireless protocol, it is prone to eavesdropping attacks. Communication between two devices over the NFC channel could be intercepted or received by an attacker in the vicinity of the devices by using a bigger or more powerful antenna [27]. The best solution against attacks on the wireless channel is to implement an encrypted channel between the two devices using a key exchange protocol. Security is not addressed by NFC in a sufficient manner, but with the inclusion of common cryptographic protocols it could be secured. It is unlikely that NFC/RFID devices would be appropriate in terms of use or cost in the use case of securing OT devices. It would be more prudent to have these devices ship with an embedded chip to act as the root of trust, and removing the need for wireless communications, as they could take use of the serial bus on the board.

3.3. Security Evaluations and Assurance

Security evaluations for smart cards and TPMs can be provided by the Common Criteria for Information Technology Security Evaluation, referred to as simply the Common Criteria [28]. The Common Criteria is an assessment methodology which provides end users and manufacturers with clear definitions of the security level of a given product. Levels of security are defined as Evaluation Assurance Levels (EALs) ranked from 1 to 7, where EAL7 is the most secure and EAL1 is the least secure. There are also augmentations that can be added to the EAL denoted by a "+" indicating additional security measures provided by the target of evaluation. For example, AVA_VAN.5 indicates that the target has been analyzed at a higher degree of scrutiny for potential vulnerabilities. MULTOS and Java Card ICs are routinely rated at EAL5+, where TPMs are rated at EAL4+. This indicates that the smart card products are typically considered to be more secure or more resistant to attacks than a TPM.

A product which receives rating EAL4 is defined as methodically designed, tested, and reviewed [29]. Giving assurance of good commercial development practices which are rigorous, EAL4 does not require substantial specialist knowledge. The EAL4 level is the highest which it is likely practical to retrofit into an existing product line. This assurance level is intended to demonstrate resistance to penetration against an attacker who is considered to have an Enhanced-Basic attack potential. The SOG-IS attack potential documentation [30] can provide more detail on definitions of attack potentials, but an attacker at this level would likely have access to some expert knowledge, a large number (tens) of samples of the product, some potentially sensitive or critical information, and specialized tools.

One level higher, EAL5 provides a substantial increase in assurance over EAL4 by requiring semiformal design descriptions, a more structured architecture, and improved mechanisms / procedures which assure that the product will not be tampered with during its development [29]. Semi-formally designed and tested, EAL5 provides confidence against penetration by an attacker with moderate attack potential.

3.4. Applicability for OT

Any secure element chosen would allow for a PLC to engage in a measured boot. Measured boot differs from what is typically referred to as secure boot as all stages of the boot process are stored in secure memory in a separate secure element and can be signed and sent to a third party for verification. Typically, secure boot is simply the process by which each stage of the boot process is verified before being loaded. A measured boot allows for the device to attest to its state after the boot process is finished, and because the secure element has a secure and random number generation method, it is resistant to replay attacks using a nonce[2]. A replay attack would consist of a compromised device simply replaying a known good measure of the boot to masquerade as a device in a good state. A nonce is a random integer to be included in the measurement of the boot state. A third party would keep track of previous nonces, and if one appears twice the device should be investigated for compromise. This boot state is trustworthy because at each stage of the boot process, the host device will report the digest of the next stage to the secure element before loading and executing it. Because the process starts at the boot ROM, which cannot be physically altered, this is a secure process.

Additionally, new firmware can be securely verified by the secure element before the device is allowed to update to avoid a malicious firmware substitution during its lifetime. Firmware should be signed by the device manufacturer, whose public key is securely stored in the secure element non-volatile storage. This allows the device to verify that the firmware is trusted and from the manufacturer. However, if there is a vulnerability in the firmware coming from the manufacturer, this could still be exploited while the device is running. Periodic state checks could be implemented in which the device sends some information about its state to help prevent against this, though it could be challenging for a third-party device to maintain a complete listing of the allowed states (i.e., whitelist) in real-time. This could also impact the required performance of the PLC.

On boot, the PLC should also poll its serial bus for installed devices to detect malicious hardware substitution or insertion. This information is measurable, signed, and reported along with the measured boot information. If a device reports back different information about the system bus than expected, it can be investigated to determine if it has been tampered with.

---

[2] A nonce is random or non-repeating value that is included in data exchanged by a protocol, usually for the purpose of guaranteeing the transmittal of live data rather than replayed data, thus detecting and protecting against replay attacks.
31. Shirey, R.W., *Internet Security Glossary, Version 2.* 2007(4949).

3.5. Recommendation

In conclusion, the best solution to act as a root of trust would be an on board embedded smart card chip in the final device. These smart card chips offer both increased functionality in the form of multi-application card operating systems, and increased assurance of security elements and protections as provided by Common Criteria evaluations. Historical supply chain attacks make clear the need for increased security in OT devices. This can be achieved in a cost effective and low intervention means through the implementation of an embedded smart card chip. This methodology has been successful over the course of decades in the telecommunications industry and is beginning to provide use across multiple new industries which adopt the technology. Secure key storage and cryptographic processing in a secure element allows for devices to be easily and trustworthily verified in real time without the need for any operator intervention. Smart card architecture also allows for additional security functions to be transferred, verified, and loaded in the case that there is a change, removal, or addition that needs to be made to ensure security.

4. PROPOSED EFFORTS

A smart card can provide a viable solution as a root of trust specifically for OT environments and products to provide tamper resistance to protect the Nuclear Supply Chain. There must be future work done to test this hypothesis. This section details a proposed environment to prototype a PLC that is protected by a smart card-based root of trust. Additionally testing procedures that simulate supply chain-based attacks are detailed to verify that the system can detect and potentially autonomously mitigate such attacks.

4.1. Test Bed Design

The chosen test bed to prototype the system is based on the J2A040 smart card chip. This is a JCOP based smart card in typical card stock form factor with 40 kilobytes of EEPROM. This is ideal because applications for the JCOP platform can be developed quickly using free and widely used integrated development environments (IDEs). JCOP is also GlobalPlatform compliant, meaning that once an application has been developed, it is loaded onto the card and placed into the environment for further testing. There is also a free Java Card emulator that can be used to test smart card applications within the IDE to verify functionality before loading onto the physical card. The JCOP card communicates with the host device using an HID OMNIKEY [32] card reader, connected via USB cable. This reader allows for APDU commands and responses, as well as card management such as loading of applications using GlobalPlatform specifications.

The JCOP root of trust is used to secure a virtual PLC which is accomplished using the open source PLC software, OpenPLC [33]. OpenPLC was installed onto a Raspberry Pi 4 Model B [34]. Raspberry Pi's are low-cost and very portable computers running ARM CPUs. In this case, the Raspberry Pi is running Ubuntu [35]. Ubuntu is an open source operating system that is commonly used in IoT devices. This set up differs from that of a typical PLC, in that typical PLCs operate on a real-time operating system (RTOS), which is preferred for time sensitive operations. This is not necessary for this use case because the PLC will not be implemented to control a physical process but will be monitored to ensure that the expected outcome is not altered in the case of an attack.

A configuration manager is also implemented using Ubuntu and a Raspberry Pi. The configuration manager is essential for the PLC to make trusted reports to for verification. The configuration manager is responsible for authenticating the PLC's root of trust (JCOP card) and information about the configuration of the machine (configuration of the OpenPLC software).

A diagram of the system can be seen in Figure 5. This system is designed using open source components wherever available to allow maximum customization and predictability. The OpenPLC software, for example, can easily be edited and recompiled to allow for real time communications with the configuration manager and root of trust wherever necessary.

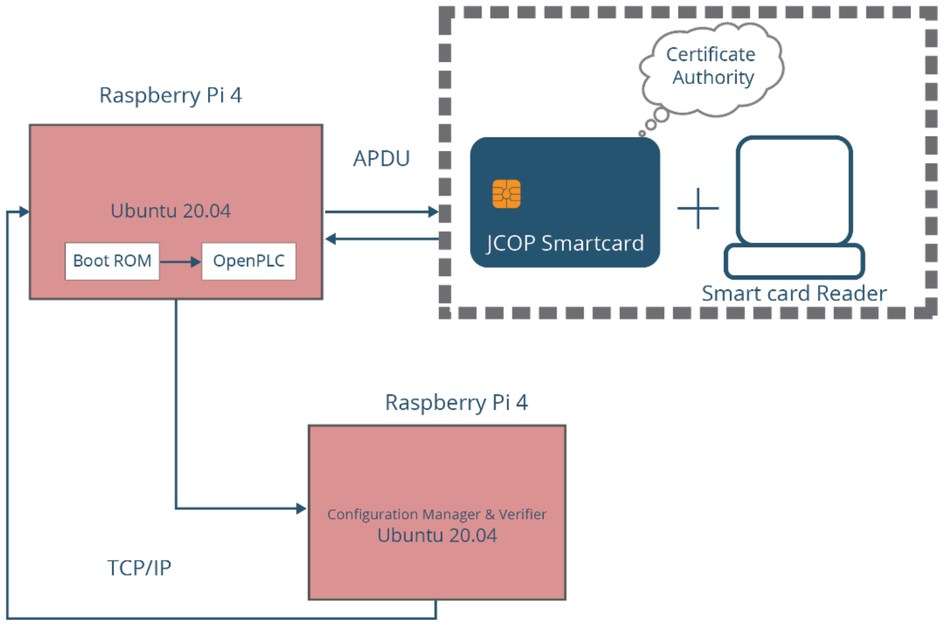

Figure 5 Test Bed Platform

A program meant to emulate a physical PLC's boot ROM executes prior to the starting of the OpenPLC application. The boot ROM takes a measurement of the OpenPLC components and provides the measurements to the smartcard. The measurement is appended to a "number used once" (nonce) and stored in non-volatile memory. This allows for the card to be polled at any time for the most recent boot configuration information, which can then be digitally signed and provided back to the host PLC for forwarding onto the configuration manager. There is also a measurement of the system bus to poll attached hardware so that the system's hardware can also be tracked. The configuration manager then determines the trustworthiness of the provided information and can provide an alert to an operator if the PLC state is determined to be untrusted or unknown. Future work using this system could also allow for the configuration manager to query the PLC for information at any time, receiving a cryptographically secure response. The PLC could also be provided with a known good firmware in real time that could be verified to have been signed by the OEM should the PLC measure some component to be untrusted and refuse to step into the next boot stage.

4.2. Verification Tests

To ensure that the test bed detailed previously functions as required, a series of test attacks will be performed on the system. The first test is a simulated change to PLC firmware. This is accomplished by making an addition or change to the OpenPLC software and recompiling. Such an attack could occur during shipment or the maintenance lifetime of the PLC, loading with either custom malicious firmware or an old version which is known to be vulnerable. This attack is detected by the configuration manager due to the reported boot measurement. Because the boot ROM is physically burned onto the board of the device, it cannot be changed, and therefore is guaranteed to make an accurate measurement of the PLC firmware. This is then signed and stored by the JCOP card, which the PLC firmware has no control over. Once this information is forwarded to the configuration manager, it will not match the expected measurement and therefore a change from the trusted state will be detected. A more advanced attack on the PLC may involve a replay attack to supply the configuration manager with a known good state. However, this is also detected by the configuration manager. The measurement that is provided to the configuration manager includes a nonce, meaning that any subsequent replay of this to the configuration manager indicates that an attack has taken place.

In addition to software/firmware based attacks it is necessary to protect against possible hardware attacks such as malicious substitutions or additions. This attack is simulated by adding a camera module to the raspberry pi in its designated slot on the board or adding an unexpected peripheral or memory device in the USB slot. Before the PLC firmware is launched, the device should poll these serial busses and send the measurements to the JCOP card for storage along with boot measurements. This information is all then signed and provided to the configuration manager, which will detect additional or different components in the hardware measurements. The final verification is when the associated JCOP card is swapped with one whose private key does not match the previous one. This will result in the configuration manager receiving a report from the device signed by an unknown key, and therefore untrusted. The signature verification algorithm should return a failure to verify error, and alert that the PLC has been compromised.

This page left blank

REFERENCES

1. *10 CFR § 73.54 Protection of Digital Computer and Communication Systems and Networks*, U. NRC, Editor. 2021.
2. ENISA, *ENISA Threat Landscape for Supply Chain Attacks*. 2021, European Union Agency for Cybersecurity (ENISA).
3. Miller, J.F., *Supply Chain Attack Framework and Attack Patterns*. 2013, The MITRE Corporation. p. 86.
4. *Cyber Security Technical Assessment Methodology: Risk Informed Exploit Sequence Identification and Mitigation, Revision 1*. Electric Power Research Institute.
5. *Regulatory Guide 5.71, Cyber Security Programs for Nuclear Facilities*. 2010, U.S. Nuclear Regulatory Commission.
6. *NEI 08-09, Cyber security plan for nuclear power reactors, Revision 6*. 2010, Nuclear Energy Institute.
7. C.I.S.A, *Nuclear Sector: Cybersecurity Framework Implementation Guidance*. 2020: US Department of Homeland Security.
8. Eggers, S. and M. Rowland. *Deconstructing the Nuclear Supply Chain Cyber-Attack Surface*. in *Proceedings of the INMM 61st Annual Meeting*. 2020.
9. EPRI, *Cyber Security in the Supply Chain: Cyber Security Procurement Methodology, Revision 2*. 2018, EPRI.
10. White, G. and M. Rowland, *Supply Chain Attacks and Solarwinds*. 2021.
11. FireEye. *Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor*. 2020; Available from: https://www.mandiant.com/resources/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.
12. Falliere, N., L.O. Murchu, and E. Chien, *W32.Stuxnet Dossier Version 1.3*. 2010, Symantec Security Response.
13. Chen, T.M. and S. Abu-Nimeh, *Lessons from Stuxnet*. Computer, 2011. **44**(4): p. 91-93.
14. Guin, U., et al., *Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain*. Proceedings of the IEEE, 2014. **102**(8): p. 1207-1228.
15. Sabt, M., M. Achemlal, and A. Bouabdallah. *Trusted execution environment: What it is, and what it is not*. in *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*. 2015.
16. Markantonakis, K. and K. Mayes, *An overview of the GlobalPlatform smart card specification*. Information Security Technical Report, 2003. **8**(1): p. 17-29.
17. Magiera, J. and A. Pawlak, *Security Frameworks for Virtual Organizations*. 2005. p. 133-148.
18. Thomas, O. *Advanced IC Reverse Engineering Techniques: In Depth Analysis of a Modern Smart Card*. in *BlackHat USA*. 2015. Las Vegas, NV.
19. Cadonau, J., D. Jayasinghe, and S. Cobourne, *OTA and secure SIM lifecycle management*. 2017. p. 283-304.
20. Shire, C., *Smart Card Technology Trends*. 2017.
21. Gan, H., et al., *Embedded Trusted Monitoring and Management Modules for Smart Solar Panels*. 2017.
22. Commission, I.O.f.S.I.E., *ISO/IEC 7816-4*, in *Organization, security and commands for interchange*. 2020.
23. *Windows 11 System Requirements*. Available from: https://www.microsoft.com/en-us/windows/windows-11-specifications?r=1.

24. Eurosmart, *2019 Shipments and 2020 Outlook - Eurosmart Confirms a Mature Worldwide Market in 2019 for Secure Elements*. 2019.
25. Boone, J., *TPM Genie: Interposer Attacks Against the Trusted Platform Module Serial Bus*. 2018, NCC Group.
26. NXP, *A700x Family Secure Authentication Microcontroller - Rev 3.1*. 2013, NXP.
27. Chattha, N.A. *NFC — Vulnerabilities and defense*. in *2014 Conference on Information Assurance and Cyber Security (CIACS)*. 2014.
28. *Common Criteria*. Available from: https://www.commoncriteriaportal.org/.
29. Criteria, C., *Common Criteria for Information Technology Security Evaluation - Part 3: Security Assurance Components (CCMB-2012009-003)*. 2012.
30. *Application of Attack Potential to Smartcards and Similar Devices v3.1*, in *Joint Interpretation Library*. 2020. p. 64.
31. Shirey, R.W., *Internet Security Glossary, Version 2*. 2007(4949).
32. *HID OMNIKEY*. Available from: https://www.hidglobal.com/products/readers/omnikey/3121.
33. Alves, T. *OpenPLC*. Available from: https://www.openplcproject.com/.
34. *Raspberry Pi 4 Model B*. Available from: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.
35. Canonical. *Ubuntu*. Available from: https://ubuntu.com/.
36. *Java Card: The Open Application Platform for Secure Elements*. 2019, Oracle.
37. *Security Target Lite NXP P531G072V0P/Q (JCOP 31 v2.3.1) Secure Smart Card Controller*. 2007, IBM.
38. *Java Card Platform Runtime Environment Specification, Classic Edition, Version 3,1*. 2021, Oracle.
39. Markantonakis, K., *Multi Application Smart Card Platforms and Operating Systems*, in *Smart Cards, Tokens, Security and Applications*, K.E. Mayes and K. Markantonakis, Editors. 2008, Springer US: Boston, MA. p. 51-83.
40. *Java Card 3.1 Documentation*. Available from: https://docs.oracle.com/en/java/javacard/3.1/index.html.
41. Platform, G., *GlobalPlatform Technology Card Specification Version 2.3.1*, in *GPC_SPE_034*. 2018.
42. Rankl, W. and W. Effing, *Smart Card Handbook, 4th Edition*. 2010: Wiley.
43. Akram, R.N., K. Markantonakis, and K. Mayes, *Firewall Mechanism in a User Centric Smart Card Ownership Model*. 2010. 118-132.
44. *MULTOS Developer's Guide*, in *MAO-DOC-TEC-005 v1.43*. 2019, MAOSCO Limited.
45. *MULTOS Guide to Loading and Deleting*, in *MAO-DOC-TEC-008 v2.29*. 2019, MAOSCO Limited.
46. *GlobalPlatform on MULTOS API*, in *GPC_SPE_032*. GlobalPlatform Inc.
47. *History*. Available from: https://www.multosinternational.com.
48. Evans, T., K. Mayes, and K. Markantonakis, *Smart Cards for Mobile Communications*. 2007. p. 85-113.
49. *3GPP*. Available from: https://www.3gpp.org/.
50. Project, T.G.P., *Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (Release 1999)*. 2007.
51. *Univeral Serial Bus Implementers Forum (USB-IF)*. Available from: https://www.usb.org/.
52. *Java Card Forum*. Available from: https://javacardforum.com/.
53. 3GPP, *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); 3G security; Security architecture (3GPP TS 33.102 version 11.5.1 Release 11)*. 2013.
54. Mayes, K. and K. Markantonakis, *Mobile communication security controllers*, in *Secure Smart Embedded Devices, Platforms and Applications*. 2014. p. 227-266.

55. Commission, I.S.O.I.E., *ISO/IEC 14443: Cards and security devices for personal identification — Contactless proximity objects*.
56. Commission, I.S.O.I.E., *ISO/IEC 15693: Identification cards — Contactless integrated circuit cards — Vicinity cards*.
57. *Host-based Card Emulation Overview*. Android Developers.
58. Yang, A. and G. Hancke, *RFID and Contactless Technology*. 2017. p. 351-385.
59. *EMVCo*. Available from: https://www.emvco.com/.
60. *Trusted Computing Group*. Available from: https://trustedcomputinggroup.org/.
61. Tomlinson, A., K. Mayes, and K. Markantonakis, *Introduction to the TPM*. 2007. p. 155-172.
62. TCG, *TPM Main - Part 1 Design Principles - Revision 1.2*. 2011, TCG.
63. TCG, *TCG Specification Architecture Overview - Revision 1.4*. 2007, TCG.
64. TCG, *Trusted Platform Module Library - Part 3: Commands - Family "2.0", Revision 01.38* 2016, TCG.
65. Arthur, W., D. Challener, and K. Goldman, *History of the TPM*, in *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*, W. Arthur, D. Challener, and K. Goldman, Editors. 2015, Apress: Berkeley, CA. p. 1-5.
66. Wang, X., Y. Yin, and H. Yu, *Finding Collisions in the Full SHA-1*. Vol. 3621. 2005. 17-36.
67. Semiconductor, L., *Mach-NX Device Family Preliminary Data Sheet*. 2020.
68. Unterstein, F., et al., *Secure Update of FPGA-based Secure Elements using Partial Reconfiguration*. 2020.
69. Group, S.-I.C.W., *SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms*. 2020.
70. Burmester, M., et al., *An RFID-Based Smart Structure for the Supply Chain: Resilient Scanning Proofs and Ownership Transfer with Positive Secrecy Capacity Channels*. Sensors, 2017. **17**: p. 1562.
71. Omitola, T. and G. Wills. *Towards Mapping the security Challenges of the Internet of Things (IoT) supply Chain*. in *Procedia Computer Science*. 2018.
72. Xiao, K., et al., *Hardware trojans: Lessons learned after one decade of research*. ACM Transactions on Design Automation of Electronic Systems, 2016. **22**(1).
73. Hadad, B., G. Kauffman, and B. Seri, *Exploring and Exploiting Programmable Logic Controllers with URGENT/11 Vulnerabilities*, in *URGENT/11*. 2020, Armis, Inc. p. 23.
74. Jin, C., S. Valizadeh, and M.v. Dijk. *Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems*. in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. 2018.
75. Mulder, J.C., et al., *WeaselBoard*. 2013.
76. Malchow, J.O., et al. *PLC Guard: A practical defense against attacks on cyber-physical systems*. in *2015 IEEE Conference on Communications and NetworkSecurity, CNS 2015*. 2015.

This page left blank

APPENDIX A. SURVEYED TECHNOLOGIES

A.1. Java Card / JCOP

Java Card is an operating system that allows for secure elements to host multiple applications, known as applets, which run using the familiar Java programming language [36]. Java Card applets share many of the same benefits as a typical Java program. Java Card applets can be executed on any Java Card complicit device because Java Card applets run on top of the Java Card Runtime Environment (JCRE), which handles interactions with the underlying hardware. The JCRE comprises the Java Card Virtual Machine (JCVM), Application Programming Interface (API), and any support services. The architecture of a sample Java Card (NXP P531G072V0P/Q) is illustrated in Figure 6.

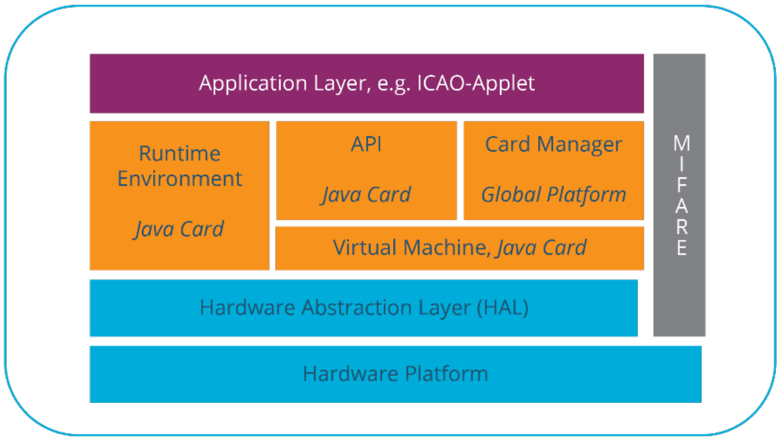

Figure 6 Java Card Abstraction Architecture [37]

The JCRE specifies an applet firewall to support the isolation of contexts and applets, meaning that one applet cannot access the objects or memory held by another applet [38]. Applets may still be able to pass information between contexts, but this must occur over specified and secure mechanisms known as interfaces. Constraining applets to a given contexts provides protection against bugs and malicious code that seeks to change or read values of another applet. JCRE also provides security to memory through atomic transaction design. A transaction is defined as "a logical set of updates of persistent data." JCRE ensures that for a given transaction either all data fields are updated or none of them are. This provides protection against program crashes or power loss where a transaction might end before its completion. Should persistent data items be only partly updated, data corruption may occur that renders the card in need of some reset. It is important for the card to always have a reliable set of information to maintain its status as a root of trust.

The development process for Java Card applets (shown in Figure 7) can utilize any standard development environment used in typical Java application development [39]. A compiler converts the source code into a class file and provides an export file with additional information. These two files then are passed on to a converter, which converts the class file bytecode into a Converted Applet (CAP) file compatible with the Java Card. The converter performs many verification procedures to make sure the application adheres to the Java Card API and framework security requirements [39]. The CAP file and export file can then be passed to the on-card installer for the applet to be loaded onto the card's EEPROM. Often, will take place according to a set of standards for managing smart cards known as GlobalPlatform.

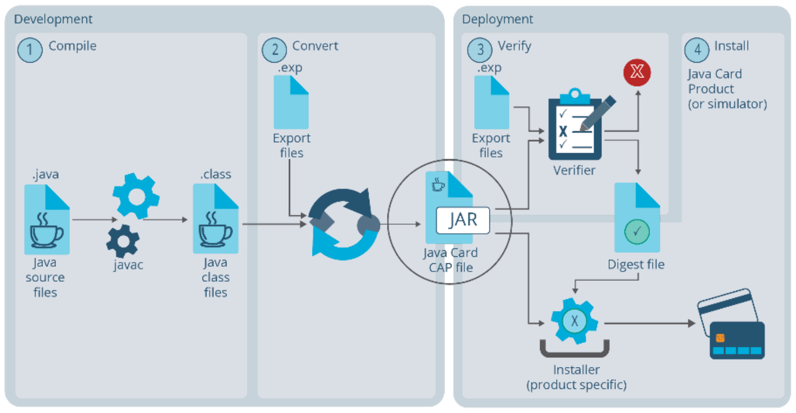

Figure 7 Java Card Development Process [40]

Many Java Cards are managed by the GlobalPlatform standard, these are referred to as JCOP cards. JCOP stands for Java Card Open Platform, indicating that the card is complicit both with java card and GlobalPlatform. GlobalPlatform functionality allows the card to be managed securely with isolated application instances and secure data loading and modification. "GlobalPlatform is a cross-industry membership organization created specifically to maintain and promote multi-application smart card standards and more specifically the GlobalPlatform specifications" [16]. The organization started its life under the name Visa OpenPlatform, a set of standards created by Visa International who had become interested in the prospect of multi-application smart cards in the late 1990's [39]. Visa donated ownership of the standards to the OpenPlatform consortium, later renaming itself to GlobalPlatform around 1999.

GlobalPlatform and therefore JCOP cards are defined by five lifecycle states [41]:

- OP_READY states that the runtime environment is available, and the card can receive and respond to APDU commands. Installation of applications is also possible in this state.
- INITIALIZED is the state which is (irreversibly) transitioned to from OP_READY. This state is used as an indication that initial data is populated, but the card is not ready to be issued to the card holder.
- SECURED indicates that the operating environment contains all necessary keys and security elements to function. This is the state that the cards are intended be in after it is issued. Once this state is entered, the card may not revert to INITIALIZED.
- CARD_LOCKED places the card in a more constrained state that SECURED, as application selection is not available in this state. This lifecycle change can be reversed with the appropriate permissions.

- TERMINATED is the final state in the card's lifecycle. A card may enter this life cycle state at any time, but this is an irreversible change. This state disables all card functionality and exists to allow for the card to be logically destroyed should there be any grave security threat detected. The only command the card will respond to in this state is the GET DATA command, which will return card capability information, and this command can only be run with certain credentials.

Also under initial development in the late 1990's was the beginnings of the Java Card. Schlumberger among other companies had announced that they would be undertaking the development of Java interpreter housed in a smart card [39]. The cards were set to support a subset of the Java programming language and bytecodes. Schlumberger had some success in this undertaking, and in October 1996 passed on ownership of the Java Card specification to Sun Microsystems and released the Java Card API version 1.0. The API was able to handle APDU commands, supported typical primitive data types, object-oriented programming, all control flow statements, unidimensional arrays etc.

Since version 1.0, Java Card has in both software and hardware capabilities. Version 2.2.1 included the addition of Advanced Encryption Standard (AES) and ECC support. Physical device characteristics have also been subject to dramatic improvements. Specifications for version 1.0 required the underlying hardware to have at least 4 kilobytes of EEPROM, where now it is easy to find JCOP card stock with upwards of ten times that amount.

A.2. MULTOS

MULTOS is another very popular multi-application smart card operating system. Most MULTOS specifications are confidential, and therefore unable to be reviewed in detail in this section [42]. MULTOS is compliant with ISO/IEC 7816-4, meaning that communications between the host and card will take the same form as all other compliant contact cards such as the Java Card discussed above. Program code for MULTOS is typically written in C and compiled into an intermediate language known as MULTOS executable language (MEL). This achieves a similar result to Java Card's hardware independence because the MEL code is then executed by an "Application Abstraction Machine" which handles program functions requiring interaction with either the operating system or underlying hardware. MULTOS applications may not be loaded onto the card without being first signed by a licensed MULTOS certification service.

Figure 8 shows the architecture of the MULTOS environment which is defined by the following components:

- Silicon / Hardware: The physical microcontroller which carries out functions for the operating system. Said functions are written in code native to the microcontroller's design, but then accessed via a virtual machine, allowing them to be addressed in the same manner regardless of underlying hardware.
- MULTOS Operating System (OS): The OS provides communications, memory management, and the virtual machine. The OS also handles installation, selection, deletion, etc. for applications as well as APDU commands and responses.
- Application Abstract Machine (AAM): provides a set of built-in instructions and functions known as primitives.
- MEL API: provides support for MEL code.

- Application Load Certificates: indicates that all applications must include a valid digital signature to be loaded onto the MULTOS card.
- MEL Application: program code written in C (or sometimes Java) and then compiled and loaded onto the MULTOS card.
- Firewalls: MULTOS separates applications with firewalls to prevent bugs or malicious code from accessing the context of another application.

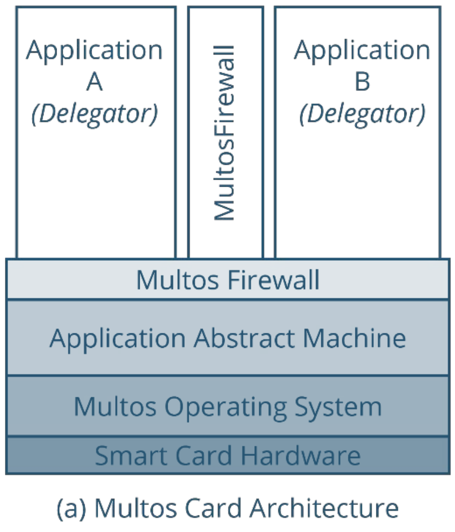
Application A (Delegator)
MultosFirewall
Application B (Delegator)
Multos Firewall
Application Abstract Machine
Multos Operating System
Smart Card Hardware
(a) Multos Card Architecture

Figure 8 MUTLOS Abstraction Architecture [43]

To allow MULTOS devices to operate in a wider range of environments, a set of operating modes are implemented: standard, shell, default, and proprietary [44]. The standard operating mode is conducive to a typical multiapplication smart card. Multiple applications exist on the card and must be selected by the host to use it. Not all existing infrastructure is designed for use with multi-application smart cards, though, and for this reason there exists shell mode. Shell mode implements a shell application that is designed to either handle APDU commands itself or route them to the correct application so that there is no need for application selection. Default mode can be thought of as a sort of in between of shell and standard. There is a default application that will accept and process APDU commands without need of selection, but the select capability is still available. Proprietary mode, implemented in MULTOS 4.5.1, allows implementers to define proprietary operational modes to meet specific product requirements [45].

Application loading on MULTOS cards is defined by five steps as shown in Figure 9 [45]. The first of which is to gather information from the destination MULTOS card. This step queries the card for information manufacturer data to determine hardware and capabilities of the card as well as MULTOS data regarding card identification and permissions required to load applications to the card. The next step is to acquire the Application Load Unit (ALU). Application code is formatted into the ALU format before it can be loaded onto the card. The ALU may be unencrypted, asymmetrically encrypted, or symmetrically encrypted during the loading process. Generally, an ALU is encrypted if it contains confidential data such as personal information. The Application Load Certificate (ALC) must also be loaded onto the card. The ALC such as the application code's hash that is used to verify that the card owner or issuer has granted permission for the application to be loaded onto the card. Next, the MULTOS card performs a series of integrity checks to verify the

ALC corresponds to the ALU and has valid permissions to load the application on the card. Finally, the application is loaded onto the MULTOS card. MULTOS cards may also be compliant with GlobalPlatform specifications, implementing the GlobalPlatform API within the MULTOS Runtime Environment [46].

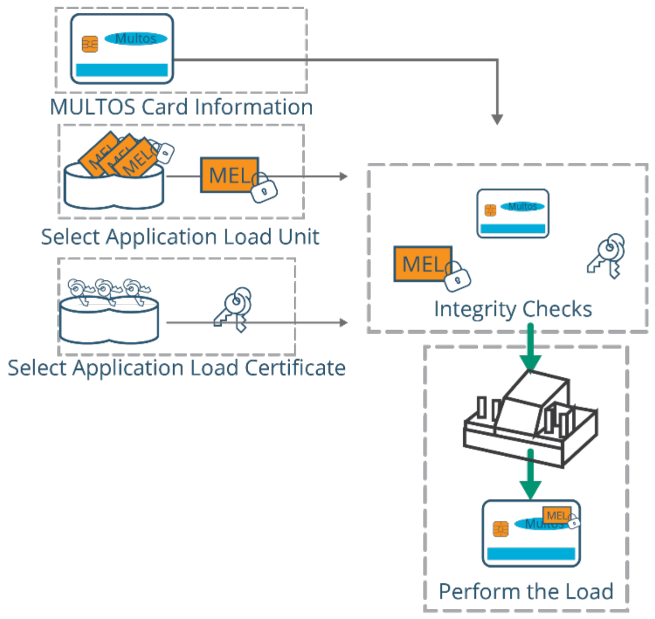

Figure 9 MULTOS Application Loading Process [45]

MULTOS has a long and rich history of smart card development dating back to the early 1990's during the initial development of smart cards. MULTOS International started as the smartcard division of an Australian company, KeyCorp Limited, in 1993 [47]. In 2008 KeyCorp's smart card business assets were purchased by Gimalto, an international digital security company, to create MULTOS International. MULTOS has garnered great popularity and become a leader in the smart card industry, hitting 1 billion devices sold in 2017.

A.3. SIM

One of the most prolific forms of smart card technology are SIM cards, which are used in mobile communications. Around 6 billion of these devices are issued every year [48]. Originally created to support authentication and confidentiality in mobile networks, SIM cards have evolved to support additional security features. SIM card success is largely due to extensive standardization, work on which is continuing. Like many things related to smart cards, there is some confusion regarding the naming conventions of SIM cards. Originally the SIM card was a standalone smart card application, whereas now the telecommunications card carries out other functions related to the smart card as well (such as payment). This new form of SIM was proposed around the same time that 3G networks were becoming standardized. Thus, the smart card application that allows 3G communication is the USIM, the evolved version of the SIM. This hardware that this application lies on is the Universal Integrated Circuit Card (UICC). It is possible for a UICC to have USIM and SIM applications on it, though the USIM can carry out the duties of a SIM for 2G communications. For this chapter, "SIM" is considered a catch-all term for telecommunication related smart cards, unless otherwise specified.

The SIM card is one of the most common applications of smart card technology, benefitting from a wide range of standardizations ensuring consumers and network providers can expect the same ease-of-use time and time again. Many of these standards are stewarded by the Third Generation Partnership Project (3GPP), responsible for developing and maintaining standardizations in 3G, 4G, and 5G networks [49]. 3GPP generally develops standards regarding the applications of SIM cards rather than their design or communications. For example, 3GPP TS 11.14 [50] defines technical requirements for the SIM (2G) Application Toolkit commands and processes. SIM Application Toolkit is a set of commands and procedures for use during the network operation phase of the Global System for Mobiles (GSM).

The UICC itself is not standardized by 3GPP, but rather the European Telecommunications Standardization Institute (ETSI). ETSI's Smart Card Platforms (SCP) group meets several times a year, and includes mobile network operators, handset manufacturers, UICC manufacturers, smart card chip manufacturers, and test equipment vendors [48]. ETSI defines access, Secure Remote Management procedures, and an API for the UICC. In order to achieve these definitions, the ETSI SCP utilizes other standards: ISO 7816-4 [22], the USB forum [51], and the Java Card Forum [52].

The SIM card's most important functionality is to provide authentication of a device to the mobile network before it is given access. Before the SIM was introduced, network providers were losing millions in revenue due to cloning fraud [48]. Since the introduction of the SIM, cloning fraud has become negligible. In short, SIM (2G) authentication works by providing a PIN protected unique identifier (IMSI) which only the network operator should know how to correlate with a phone number. The IMSI should remain a secret, and not be transmitted in clear text; in this case, a temporary version (TMSI) should be used instead. When the phone powers up, it polls the SIM for information stored in PIN protected files including the IMSI. The phone sends the IMSI to the network, and requests to use the network. The network will then issue a challenge to the phone consisting of a 128-bit random number. The phone then passes on this challenge to the SIM to calculate a 32-bit response, and a 64-bit cipher key (like a session key in TLS) using its secret key, which is also known by the network.

The 2G SIM protocol poses a few security concerns for a modern ecosystem. For example, there is no authentication of the station / network by the phone. This could lead to a potential man in the middle attack where a malicious station is created to relay communications between a phone and legitimate station to eavesdrop or potentially alter data. To mitigate these and other concerns, improvements were made in the 3G/USIM approach. For example, the USIM authentication challenge now includes a Message Authentication Code (MAC), to ensure that data has not changed while in transit [48]. Additionally, the challenge includes a sequence number which will aid in prevention of replay attacks. Additional keys were also introduced to the protocol to ensure integrity and anonymity on the network. The USIM in the 3G network can authenticate the network using the process shown in Figure 10, computing an expected MAC (XMAC), a response to the challenge (RES), and 128-bit keys for ciphering (CK) and integrity (IK).

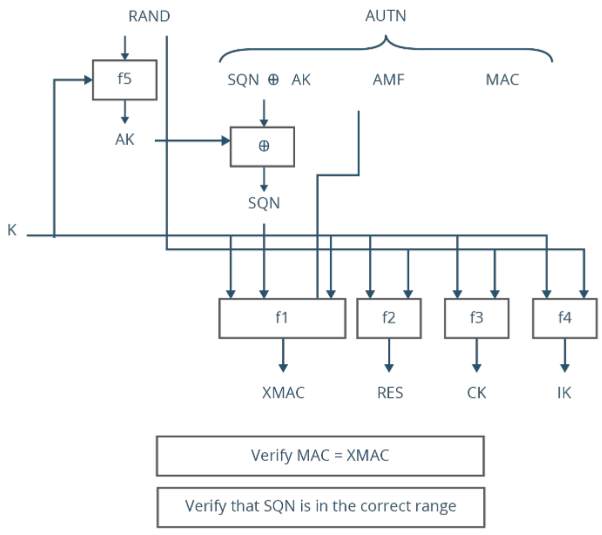

Figure 10 User Authentication Function in the USIM [53]

The SIM was originally defined in 1990 by ETSI, and the USIM was defined in 1999 by 3GPP. In 1988, before the SIM, there was the C-NET system that used smart cards in telecommunication [48]. There is also a telecommunications enabling smart card design called RUIM in some competing cellular standards [54]. The SIM has been able to provide great value to telecommunications providers for decades. Further, it has matured along with those networks to provide a reasonable amount of security to users throughout those years. Not only has UICC hardware greatly improved, but the SIM card applications that utilize them have as well. One example of this has been shown in the improvements to authentication made between 2G and 3G generation SIMs. Subsequent generations of telecommunications technology have incorporated many additional improvements.

A.4. Contactless Cards

In recent years, RFID technology has grown quite quickly in popularity. Many contact cards are either switching to be contactless or implementing contactless communications along with contact communications. Often, contactless smart cards are referred to as RFID cards. This is somewhat misleading, though, as RFID only indicates Radio Frequency Identification. RFID cards may simply provide a unique identifier over NFC when polled with no need for a secure challenge and response or no capability of any other tasks like multi-application smart cards. Contactless smart cards, referred to as proximity cards are defined by ISO/IEC 14443 [55]. Though, there are other standards for RFID tokens; for example, ISO/IEC 15693 [56] standardizes "vicinity cards," generally used for applications such as item tracking rather than secure transactions or identification.

ISO 14443 is defined by four parts. Part one defines the physical characteristics of the card to be the same as that of ISO 7810: a typical credit card or ID card size that consumers are familiar with. Part two defines the radio frequency power and signal interface. The interfaces enable communications and power transfer. Since contactless cards never contact an external device, they must be powered over an alternating magnetic field. Data transfer is possible in two different modes defined as Type A and Type B. The basic data rate for both modes is 106 kbps. The third part of the standard details the initialization and anti-collision measures. Initializing communications between contactless card and reader becomes more complicated because the reader must constantly poll the space for available cards and when setting up communications, must be sure not to interfere with any other potential cards or readers in the area. In both Type A and Type B, the smart card will act as a state machine in the "Power Off" state until power is received, then entering the "IDLE" state. The reader will then send a request to establish a connection, and a transaction ensues in which the reader and card must agree on a channel for communication. Part four of the standard defines the transmission protocol between the card and reader. For Type A, some additional parameters such as frame size and card identifier are required; this information will have already been exchanged in Type B. The reader should then transmit a Request for Answer to Select command, to which the card should respond with an Answer to Select Command. This may be followed by a Protocol and Parameter Selection command, allowing for the card and reader to change the data rate for communications. The card and reader may now transmit data between each other in the form of APDU commands and responses.

Another use for contactless technology is for the emulation of smart cards. As illustrated in Figure 11, Android Host Card Emulation (HCE) [57] allows for Android smartphones to emulate a smart card and communicate directly with an NFC smart card reader. Typically, the card emulation is provided using a secure element. This can be on the device itself or using the SIM card in the device. In this case, the secure element and the terminal communicate directly through the phone's NFC controller, and there is no Android application involved in the communications or computations. Though, it is possible for an Android application to poll the secure element to get transaction status to provide that information to the user. However, to support the possibility that an Android device does not have a secure element, host card emulation will emulate a smart card in the form of an Android application and handle all APDU reception, processing, and response of APDU commands.

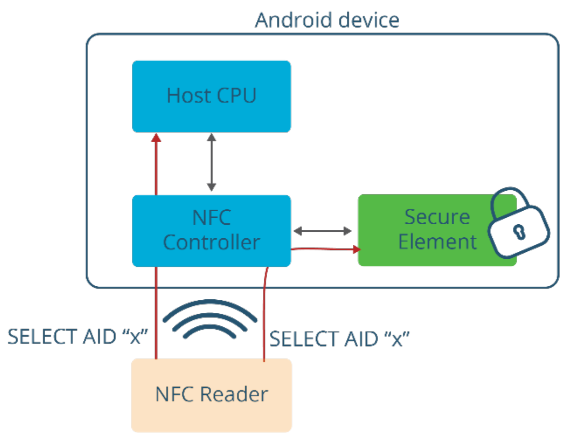

Figure 11 Android Host Card Emulation [57]

It is also possible to have run HCE in conjunction with a secure element on the device. The NFC terminal should provide the Android device's NFC controller with a SELECT AID command to communicate what smart card application will be needed. The AID denotes the unique application

ID of the desired application. The NFC controller contains a routing table which will then be queried, sending the SELECT command and all subsequent APDU commands to the correct destination until either another SELECT command is received or the link with the terminal is broken. The Android device will also contain some default routing in the case that the AID provided does not correspond with anything in its routing table, though this may differ by device manufacturer [43]. Android applications do not need to define anything other than the AID that they are meant to service, and the operating system will then handle the routing table from there. Android provides security for HCE by making the NFC service a system level privilege, meaning that any information routed to the card application comes from the OS. Android applications may also implement the Android Application Sandbox, which isolates application data from other applications, like the firewall implementation of the Java Card OS and MULTOS. Of course, if a device is compromised with system level privileges, there is a possibility that malicious APDU commands could be sent to the HCE application to process fraudulent transactions or extract sensitive information.

Although contactless cards have recently experienced a rise in popularity, the idea has been around for quite a while. Charles Walton first invented an RFID-based physical access control system in 1973 [58]. RFID has also become a popular solution for ticketing at events or locations like ski resorts. Contactless cards have also been commonplace as payment solutions since the 1980s. RFID cards could be used as an identifier for a user correlated with an amount of money available in closed systems such as a laundromat. Though these capabilities were available, they were not secure unless coupled with a secure integrated chip, which would not happen until the late 1990's / early 2000's. Since then, contact cards have seen massive interest for the purpose of payment in more safety critical situations like credit and debit card payments. Since 2007 EMVCo [59] has been the steward of the EMV (Europay, Mastercard, Visa) standard which defines the application of ISO 14443 compliant contactless smart cards in credit card payments.

A.5. Trusted Platform Module

The TPM is the result of a common desire across industries and manufacturers for a way to provide some secure processing which is implemented using some secure hardware component; it is specified by the Trusted Computing Group (TCG) [60]. TPMs are not meant to be cryptographic accelerators. The TPM is meant to be a cost effect secure module that provides the basic building block of this capability. The TPM differs from a typical smart card IC in several ways, one of which being that the TPM is typically integrated to the larger system's bus as a chip or sometimes implemented as removable media. TPM specifications do not require that the TPM be implemented as an IC but define the TPM's required functionalities. TPM specifications require that they be physically protected from tampering and that the design be tamper resistant, the former is typically accomplished by physically binding the TPM to the motherboard [61]. The required architecture of the TPM (v1.2) is shown in Figure 12.

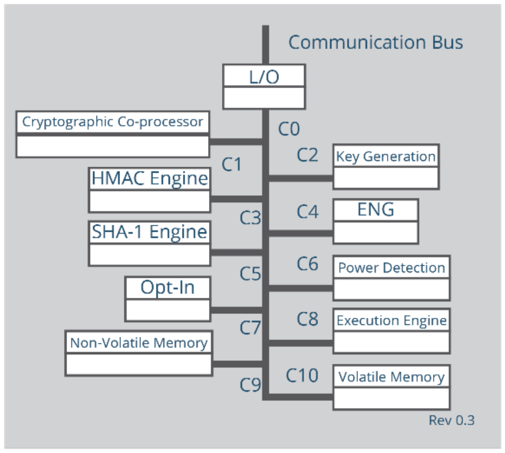

Figure 12 TPM Architecture [62]

- I/O Component: Responsible for managing information flow to and from the communication bus. The I/O component must perform encoding and decoding of communications protocols and route information; accordingly, it also enforces access policies associated with the Opt-In component as well as other TPM functions requiring access control.
- Cryptographic Co-Processor: Implements cryptographic operations within the TPM. This must include RSA encryption, decryption, and key generation for use with key sizes of 512, 1024, and 2048 bits. The standard does not mandate a specific implementation of the RSA algorithm. The minimum recommended key size is 2048, and minimum required key size (required only in FIPS mode) is 1024. The TPM may implement other asymmetric encryption protocols such as elliptic curve, but this is not a requirement. The TPM may also use symmetric encryption algorithms for communications on the TPM communications bus, but none of these algorithms will be made available to general users of the TPM.
- Key Generation: Creates RSA key pairs and symmetric keys. There is no minimum requirement on key generation times for asymmetric or symmetric keys.
- HMAC Engine: Provides assurance that the request arriving is authorized and has no modifications made to the data while in transit. The hashing algorithm used is SHA-1.
- RNG: The random number generator component is the source of randomness for the TPM. These values are used for nonces, key generation, and randomness in signatures. Pseudorandom number generators are allowed, and a specific source of entropy is not defined; it is left up to the manufacturer.
- SHA-1 Engine: Provides the TPM with a trusted implementation of a hashing algorithm. The hash function is also available for access outside of the TPM, which will support measurement of boot phases.
- Power Detection: Responsible for managing the TPM power state. It is required for the TPM to be notified of all power state changes. This can be used to restrict command execution during restricted times such as during boot.
- Opt-In: Allows the TPM to be enabled/disabled and maintains the state of non-volatile and volatile flags. Changing such flags requires authorization by the TPM owner or assertion of physical presence. Physical presence detection techniques are open to be designed by the TPM manufacturer.

- Execution Engine: Responsible for executing the TPM commands received from the I/O port.
- Non-Volatile Memory: Used to store persistent information for the TPM such as identity or state information. Non-Volatile Memory is also available for use by entities external to the TPM that are authorized for use by the TPM owner. TPM manufacturers are instructed to consider the use case of the TPM and avoid high volume writing to non-volatile memory to avoid premature wearing out of the TPM as non-volatile memory has a limited life.
- Volatile Memory: Used to store temporary information by the TPM that is not required to be persistent through a power cycle.

Using a TPM as a root of trust in a device is helpful for many security features, one of which is securing the boot process of a device. This process is shown in Figure 13. The first step in the system boot process is the BIOS boot block or Trusted Boot Block (TBB). This contains the Core Root of Trust for Measurement (CRTM), which is physically embedded onto the host chip, is the first set of instructions run and implicitly trusted by the TPM. CRTM is not measured by any external code, but it may perform integrity checks on itself [61]. The CRTM will measure the integrity of the rest of the BIOS before loading it. Then, the integrity of the OS Loader is measured, and booted into once validated. The OS Loader is then given control, and able to measure the integrity of the OS Code. This process of measuring and passing off execution continues until applications are loaded. The TPM is responsible for producing SHA-1 measurements of the code to be executed in each step. At each stage of the boot process the platform configuration register (PCR) is updated with the current hash of the stage to be loaded. The PCR starts at 0 and is appended with the next stage: $PCR \leftarrow H(PCR||H(next\ stage))$. The digest of each stage is stored in what the TCG refers to as the Stored Measurement Log. This way, the value of the trusted PCR can be verified by repeating the hashing at each stage with the known digest of each stage.

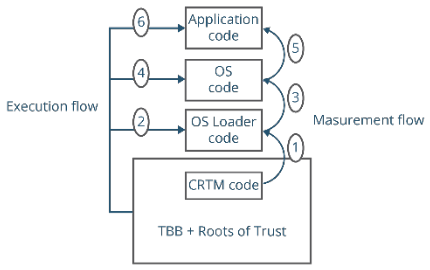

Figure 13 Secure Boot Process [63]

TPMs can process a multitude of commands related to encryption, random number generation, and reading of non-volatile memory [64]. TCG specifies a long set of commands that each TPM is required to implement. Response codes and error codes are also specified. TPMs may also implement vendor specific commands on top of those required by TCG. However, TPM's cannot be flashed with new applications like a multi-application smart card. A TPM may receive a firmware update that includes additional functionality, but typically the TPM must be cleared before it can receive any updates to firmware.

Another key concept of TPMs is that of ownership. When a user takes ownership of a TPM they establish a shared secret, which is referred to as owner authorization data [61]. This information is then placed into secure storage on the TPM. Ownership allows for access to be regulated to only the specified operator of the device, requiring a proof of ownership by requesting the owner authorization data. For a user to be able to take ownership of the TPM in a secure manner, the owner authorization data must not be transmitted in clear text. To achieve this, unowned TPMs contain a private EK, which can be used to initially establish a secure connection to the TPM. Once ownership of the TPM is assumed by a user, the TPM will then create a storage root key to be stored in non-volatile memory, and act as the root of key hierarchy on the TPM. It is vital that neither the EK nor the storage root key ever leave the TPM.

The first widely deployed TPM was 1.1b, released in 2003 [65]. These included RSA key generation, storage, secure authorization, and device-health attestation. PCRs were used to attest to the system's boot sequence. The TCG specifically decided not to define any specifications to defend against physical tampering attacks, leaving that to specific manufacturers. TPM 1.1b did have issues, one of which being the lack of standardization for software interfaces. Different vendors used different interfaces, requiring different drivers, and the pinout of the chips was not standardized. TPM 1.2 was developed from 2005-2009 to mitigate some of these concerns; this version would go through several releases. This included a well-defined software interface and some standardization to the pinout. TPM 1.2 also specified that there must be protection against brute force and dictionary attacks, which are exhaustive attempts to gain access to the TPM. TPM 1.2 relied quite heavily on the security of the SHA-1 hashing algorithm, which was shown to lack proper collision resistance in 2005 by Wang et al. [66]. Subsequently, SHA-1 was officially considered deprecated by NIST in 2011, which led to the development of TPM 2.0, released in 2014. TPM 2.0 addressed the SHA-1 function by requiring additional hash functions, added support for additional cryptographic algorithms (such as elliptic curve), enhanced the availability of the TPM to applications, enhanced authorization methods, and simplified TPM management [61].

A.6. FPGA

An emerging capability of FPGAs is to perform the duties of a root of trust. An FPGA is a common asset used in the OT space. They are ICs, either meant to be configured by the customer after they have been manufactured or loaded with a one-time programmable memory. FPGAs are made up of an array of programmable logic blocks which can be configured to carry out some logical operation and produce an output. Most of the time, FPGAs are implemented using some flash memory to store configuration data. However, there is the security concern of verifying configuration data of the FPGA. Some manufacturers are now producing FPGAs with a build in secure enclave, making it possible to verify signed configurations before execution. One such example of this is the Mach-NX device family produced by Lattice Semiconductor.

Figure 14 shows the layout of the Lattice Mach-NX device family and the included secure enclave. The features supported include AES-128/256, SHA-256/384, Elliptic Curve Digital Signature Algorithm (ECDSA), public key cryptography, and a unique secure ID [67]. The devices are low power, instant-on, flash-based FPGAs. They also include I2C and SPI ports, allowing them to communicate with other chips easily. Some work has been done on using FPGA as a root of trust [68], though the research is quite new and there is a lack of standardization both for the secure enclaves included in the FPGAs and for the use of FPGAs as a secure root of trust.

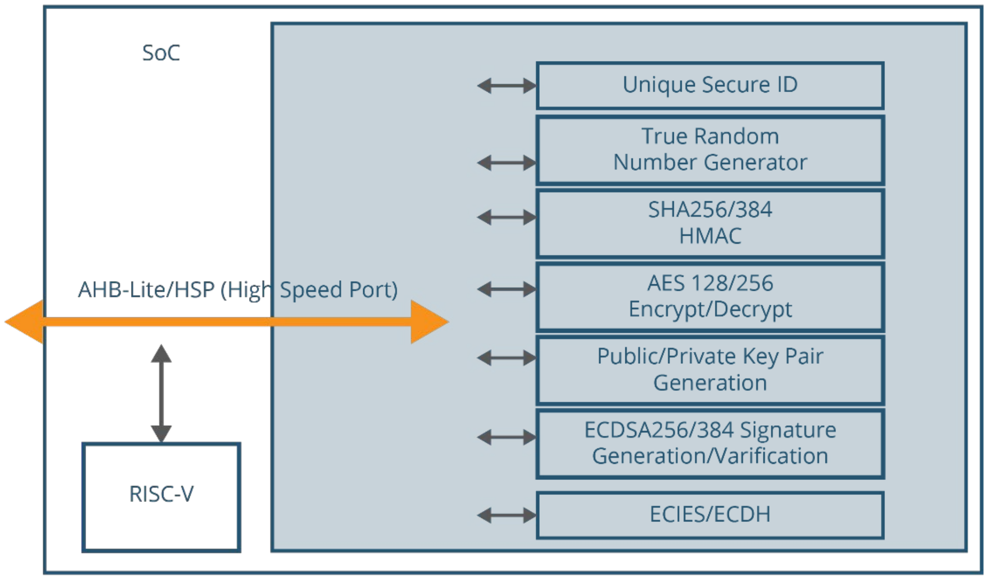
SoC
AHB-Lite/HSP (High Speed Port)
RISC-V
Unique Secure ID
True Random Number Generator
SHA256/384 HMAC
AES 128/256 Encrypt/Decrypt
Public/Private Key Pair Generation
ECDSA256/384 Signature Generation/Varification
ECIES/ECDH

Figure 14 Lattice Mach-NX FPGA Architecture [67]

There is some aversion amongst the industry regarding the inclusion of secure enclaves in FPGAs. The reason for this is that what makes the FPGA such a popular solution is its simplicity. Inclusion of a secure enclave in the FPGA means that it will require some sort of processing unit capable of routing commands to the secure enclave if necessary. This increases complexity, and therefore may have an adverse effect on the time to boot and time to process commands within the device.

This page left blank

APPENDIX B. LITERATURE REVIEW

B.1. Summary

Considerable effort has been posed across multiple industries as well as government and academic institutions in root of trust. Technologies to support roots of trust are mature and well standardized technology, for which standardization and innovation efforts continue. Historically, the driving force behind this has been telecommunications. There is also a wide range of standards and guidelines for computer security in nuclear power plants. Generally, these tend to acknowledge the criticality of supply chain security but leave an unrealistic responsibility on the licensee or are too complex to provide assurance that licensees will be able to accurately apply security measures in their supply chain procedures. There is also a substantial effort in proving concepts for attacks on OT devices and developing countermeasures, though there is a lack of consideration in the literature specifically for securing devices throughout the entire supply chain.

B.2. Findings

An array of standards and documentations exist that are informative of the state of the art in both root of trust technology and OT security. A review of said documents is presented in Table 5. There are many documents providing guidance on various aspects related to root of trust technology, including communications to and from smart cards, smart card OS descriptions and application development therefor, and requirements laid out by the TCG for development and use of the TPM. Additionally, there is guidance on preferred or required cryptographic mechanisms for use in secure systems.

Table 5 Relevant Standards and Documents

Name	Description / Relevance
ISO/IEC 7816 [22]	Describes the protocol and implementation for communications with a contact based smart card. This requires a physical connection between card contacts and the reader. This standard is used for communications between the host and the smart card based root of trust.
ISO/IEC 14443 [55]	Describes the protocol and implementation for communications with a contactless proximity card. The range of a proximity card is typically around one foot or less.
ISO/IEC 15693 [56]	Describes the protocol and implementation for communications with a vicinity card. A vicinity card communicates with a greater range than a proximity card, up to about six feet.
GlobalPlatform Technology Card Specification Version 2.3.1 [41]	GlobalPlatform defines a smart card specification that can be adhered to by any smart card platform. It defines a standard that allows for cards to be managed (updated to add or remove applications, changed to different lifecycle states, etc.) regardless of underlying hardware, OS, or vendor.
Java Card 3.1 Documentation [40]	Java Card specifies a multi-application smart card OS that allows for Java-based applets to be run on smart cards. Many cryptographic functions are supported by Java Card and application contexts are separated by a firewall.
MUTLOS Developers Guide [44]	MULTOS specifies a multi-application smart card OS. MUTLOS runs applications typically developed in C and includes a firewall between

Name	Description / Relevance
	application contexts. MULTOS requires additional security measures when loading applications on a smart card.
SOG-IS Agreed Cryptographic Mechanisms [69]	This document outlines many cryptographic mechanisms used in smart cards and whether they are acceptable and are likely to remain acceptable in the future. This document should be referenced while deciding on any cryptographic mechanism to ensure that it is not likely to be deprecated and that a sufficient key size is used.
TPM Main - Part 1 Design Principles - Revision 1.2 [62]	TCG defines the design principles of the TPM. The TPM requires a set of cryptographic functions to be supported as well as the components in a TPM.
Common Criteria for Information Technology Security Evaluation [29]	Common Criteria provides a standardization for evaluating the cybersecurity of hardware, software, or a combination of the two. This allows for a consistent measurement of the security provided by a given root of trust implementation.
EPRI TAM [4]	The TAM provides a "bottom up" methodology intended to assess and mitigate cybersecurity threats to equipment in power plants.
EPRI Cyber Security in the Supply Chain: Cyber Security Procurement Methodology, Revision 2 [9]	This report integrates the TAM into the supply chain specifically. It outlines steps that the end-user can take (such as supplier questionnaires) to ensure that components purchased are secure throughout the supply chain. This leaves the end-user with liability for points in the supply chain which they do not have control or visibility of.
NEI 08-09 [6]	Guidance on cybersecurity for Critical Digital Asset (CDA) cybersecurity. Licensees are required to employ cybersecurity in the supply chain by establishing trusted distribution paths, validate vendors, and require tamper proof products or tamper evident seals. These requirements also leave the licensee with the ultimate responsibility and are lacking sufficient detail for supply chain security.
US NRC Regulatory Guide 5.71 [5]	

There has been some research done on cybersecurity in the supply chain, but typically not with the specific outlook on OT or NPPs. Eggers [8] has proposed a succinct definition for the attack surface of the supply chain for digital instrumentation and control devices used in nuclear power plants. Eggers, Rowland [8] expanded on this later and applied the concept to a number of publicly acknowledged supply chain attacks. This work is vital in understanding what points of the supply chain are most vulnerable and where security can be strengthened to provide assurance to consumers without placing much additional work or liability on the consumer. The supply chain is defined from the initial design of devices to decommission, with likelihood of targeted attacks increasing as the device progresses in its lifecycle. There are also several attack types listed, which are useful in describing what types of attacks a system is capable of mitigating.

There are also several works in general supply chain security. Burmester, et. Al. [70] propose a solution for transfers of ownership of devices during shipment based on RFID cards. Because of the nature of vicinity cards, an embedded contactless card in a device could be extended to make use of this capability, which would allow for a secure audit trail of devices after leaving the manufacturer and before reaching the consumer. Omitola, Wills [71] have worked on mapping supply chain vulnerabilities of IoT devices. This work outlines the complexity of supply chain security, detailing the many different manufacturers of subcomponents that are included in a device such as the iPhone. Xiao, et. Al. [72] have analyzed the growing threat of hardware trojans, malicious modifications of circuit designs. A key takeaway is the need for developing devices which can resist such modifications. run

Notable work in PLC security has also been underway throughout recent years. Hadad, et. Al. [73] explore the possibilities of exploiting PLCs based on the URGENT/11 vulnerabilities, which are present in VxWorks. VxWorks is an OS which many PLCs rely on. Researchers were able to gain remote code execution on the PLC based on these vulnerabilities. Proper implementation of a certified firmware and configurations using a root of trust would allow for a PLC to be resistant against the consequences of such an attack. Jin, et. Al. [74] propose "Snapshotter," a system which would monitor input and output of a PLC in order to detect a potential compromise. Researchers indicate that on modern PLCs, firmware can be modified to include the Snapshotter agent. A root of trust should be included to guarantee the integrity of such a solution. Similarly, Mulder et. Al. [75] developed "Weaselboard" capable of monitoring the backplane of the PLC and analyzing traffic between components for the purpose of detecting compromise. Another such work is "PLC Guard" by Malchow, et. Al. [76]. PLC Guard works by intercepting traffic between an engineering workstation when code is being loaded onto the PLC. The code that is being provided to the PLC is then displayed for the engineer to perform an integrity check to ensure that the code is not compromised. In this case, a root of trust in the PLC would be able to verify the digest and signature of the code autonomously which alleviates work and responsibility from the engineer as well as the potential for human error.

In conclusion, there is a heavy effort for research and development in supply chain security, industrial control system and NPP security, and supply chain security. There is a need for more work in the intersection of these efforts.

This page left blank

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Lon Dawson	8851	ladawso@sandia.gov
Michael Rowland	8851	mtrowla@sandia.gov
Benjamin Karch	8851	brkarch@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

Email—External (encrypt for OUO)

Name	Company Email Address	Company Name
Katya LeBlanc	Katya.LeBlanc@inl.gov	INL
Shannon Eggers	Shannon.Eggers@inl.gov	INL

This page left blank

This page left blank