CYBRARY

IoT Foundations of Trust

Whitepaper

Authored by Matthew Clark
Cybrary Fellow

Whitepaper

IoT Foundations of Trust

This whitepaper will discuss the technical foundation of trust necessary to establish proper security within an Internet of Things (IoT) ecosystem. It will identify the trust attributes necessary to create a root of trust within an IoT device. It will also outline how secure elements, cryptography, immutable code, secure boot, and isolated environments are used to achieve the attributes of trust. Lastly, the paper will outline the common implementations of hardware roots of trust including the Trusted Execution Environment, Trusted Platform Module, Hardware Security Module, Device Identifier Composition Engine, and Smartcards.

Foundations of Trust

Foundations are an important principle, not only in architectural design but in security as well. A southern folk children's song called The Wise Man and the Foolish Man illustrates the importance of stable foundations. The lyrics describe the manner in which two men built their homes, with the first man building his structure on a firm foundation made of rock while the second man choosing to build on an unstable foundation made of sand. At first, both structures appear to be equally matched; that is until the rain comes and washes away the sand from under the second house.

IoT devices have unique characteristics that make them susceptible to security and privacy risk. These include a price-sensitive market, long device lifetimes, and the fact that most IoT devices exist outside a defined security parameter. Unfortunately, many manufacturers have been willing to trade production cost, consumer convenience, and time to market with the implementation cost, design complexity, and additional time required to properly implement secure IoT product ecosystems.

Recent headlines of privacy stealing cameras, IoT zombie botnets, and smart car hacking underscore the importance of implementing security at the earliest stages of the product lifecycle. In two recent surveys, consumers expressed their expectation that security should be built into IoT products.

IoT Foundations of Trust

A January 2020 ADT survey found that 92% of consumers feel companies should take measures to protect customer data and information (ADT, 2019), and a December 2019 survey found that 87% of IoT consumers expect their device to be secured by the manufacturer (Business Wire, 2019). Both of these are strong indicators that consumers expect manufacturers to address the unique security requirements of IoT products before they purchase them.

However, while consumers expect greater security and privacy, it is not always easy for manufacturers to add it because consumers have rarely been willing to pay for it. On this topic, Brian Krebs stated, "Years of experience has shown that consumers are not interested in paying a huge premium for security when a comparable product with the features they want is available much more cheaply" (Krebs, 2018). This might explain the disconnect in a February 2020 PwC survey, which found that 90% of business leaders recognize that gaining customer trust will lead to a competitive advantage; however, less than half consider security and privacy to be a top priority (Mendoza, 2020).

A strong foundation for IoT devices is important given long device lifespans. In many cases, an IoT device's lifetime is based on the battery lifetime, which can last between three to ten years (D'mello, 2019). In industrial settings, where equipment is expected to last twenty to thirty years, the lifetime of sensors and actuators is even more pronounced (Hanna, 2016). These long device lifetimes increase the likelihood that vulnerabilities will eventually be discovered, if not within the product or sensor itself, then within the greater ecosystem where the product is deployed. Without proper lifecycle management planning, these risks will go unmitigated.

Many IoT devices exist outside a defined security perimeter and may require special threat assessments in order to determine how to best address security requirements. In industrial settings, devices may be physically difficult to reach because they are embedded in manufacturing equipment or installed in geographically remote locations. In commercial settings, some products, such as point of sale devices, can be so ubiquitous that it makes manual software updates timely and arduous to schedule. In consumer settings, devices may be installed on insecure or unstable networks, requiring security best practices to be configured as default factory settings.

IoT Foundations of Trust

Root of Trust

These unique characteristics of IoT devices require proper foundations on which to build trustful IoT ecosystems where devices, mobile and cloud applications, and data can be secured. In his article *Building security, privacy and trust in IoT deployments*, Ashwin Krishman stated, "The T in IoT doesn't stand for trust, but it's a critical component of any IoT deployment." Ashwin is correct. Establishing trust is essential to developing device security and instituting privacy.

From a conceptual standpoint, trust implies several key attributes:

- Identity and authentication – a device can be verified to be what it claims to be
- Integrity – a device, including its firmware, software, and data have not been modified
- Authorization – only proper access is granted
- Confidentiality – secrets can remain secret
- Accounting – output is within expected parameters

The Latin root word ver means truth. This root word can be seen throughout the English language in different words such as very, verily, verify, verdict, and verification. The fact that so many different words are required to communicate a simple concept underscores the difficulty of achieving that goal. Many of the same challenges exist in the technical world.

To solve these challenges, security architects and engineers attempt to create a strong foundation known as a "root of trust". A root of trust acts as an undeniable source of truth – a starting condition that can always be trusted. Roots of trust are technical implementations of the conceptual attributes of trust above and can be implemented as hardware or software.

Roots of trust generally are found in hardware as software solutions are not robust enough. There are solutions on the market today that claim to be able to establish a proper root of trust using software solutions; however, those claims are not in scope for this paper. The National Institute of Standards and Technology (NIST) addresses hardware and software roots of trust in NIST SP-164. "Hardware [Roots of Trust] are preferred over software [Roots of Trust] due to their immutability, smaller attack surfaces, and more reliable behavior. They can provide a higher degree of assurance that they can be relied upon to perform their trusted function or functions" (Chen et al., 2012).

IoT Foundations of Trust

A root of trust is typically defined as having the following core technical components:

- Secure Element
- Cryptographic functions
- Immutable code
- Secure Boot
- Isolated Environments

Secure Elements

A secure element is an autonomous microprocessor chip that supports security services such as certificate storage, signing, verification, encryption, and hashing. These tamper-resistant silicon chips act as companion devices to standard processors and provide important defense-in-depth technology that is used to implement trust attributes. For example, private keys that are stored in secure elements can be hardware locked so that they are inaccessible from both physical probing and logical attack.

There are two basic types of secure hardware chip architectures: external and integrated. External chip architecture pairs a microprocessor with a separate secure element. An integrated chip architecture includes normal processing activities along with secure processing in the same silicon chip. Both individual product security requirements and cost often drive the architecture choice.

External

One of the main advantages of external secure elements is their wide interoperability with other chips. Although external secure elements can be paired with many different microprocessors, implementation is not always a simple process. Each microprocessor will require custom driver integration with the secure element in order to interact properly with the environment. This custom integration should be planned for in the overall IoT project in terms of time, resources, and cost.

IoT Foundations of Trust

When using these secure elements, there are security requirements that need to be considered during the design stage. The primary security consideration is the external communication channel between the processor and the secure element. As the diagram below indicates, this channel oftentimes runs unsecured between both physical elements. This vulnerability can be exploited by a threat actor through either physical or side-channel attacks to reveal cryptographic secrets or intellectual property (Sabev, 2017).

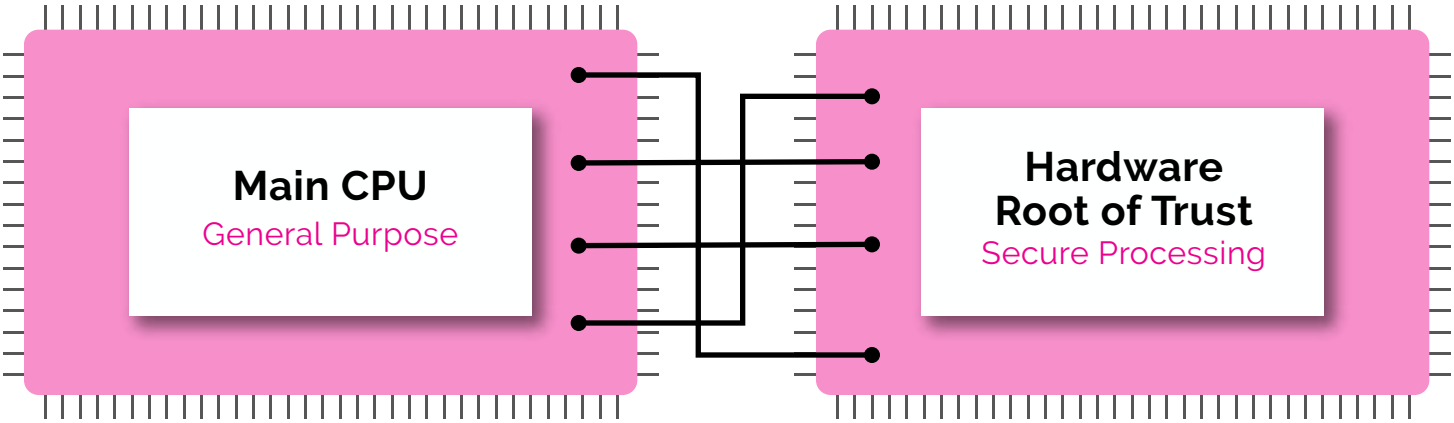

Figure 1: Basic secure element design

As with any secure element design, careful consideration will need to be given to the provisioning process if contract manufacturers are used. A secure process will need to be designed to provision the chip with the necessary cryptographic keys, identities, and secrets without exposing them during the manufacturing process. Additionally, steps will need to be taken to protect intellectual property, such as firmware and application code.

Integrated

System on a Chip (SoC) architecture has been around since at least 1970 when Hamilton Pulsar unveiled the first "Wrist Computer" watch on the Johnny Carson Show (Computer History Museum, 2020). SoC implementations have evolved over the years from wristwatches to embedded systems and mobile computing.

This technology can be combined with artificial intelligence, machine learning, and neural network technology to create edge computing, which has broad implications in IoT ecosystem architecture (Garibay, 2017).

Modern SOC designs combine many components of traditional computing hardware and software on a single chip. In her article entitled What is an SoC? A Basic Definition, Scharon Harding, Senior Editor at Tom's Hardware, explained the three basic types of System on a Chip as being those that use a microcontroller, which constitutes a chip with the CPU, RAM, ROM, and potentially other components, those that use a microprocessor, which is a chip with a CPU only, and those that are designed for specific applications, which may or may not use a microprocessor or microcontroller (Harding, 2019).

SoC implementations do not necessarily have security built-in. For secure implementations, a SoC can be combined with a secure element, such as a Hardware Security Module (HSM), to establish a secure root of trust. This enables each device state to be secured from being powered down to device initiation, throughout the boot process, and during normal operations.

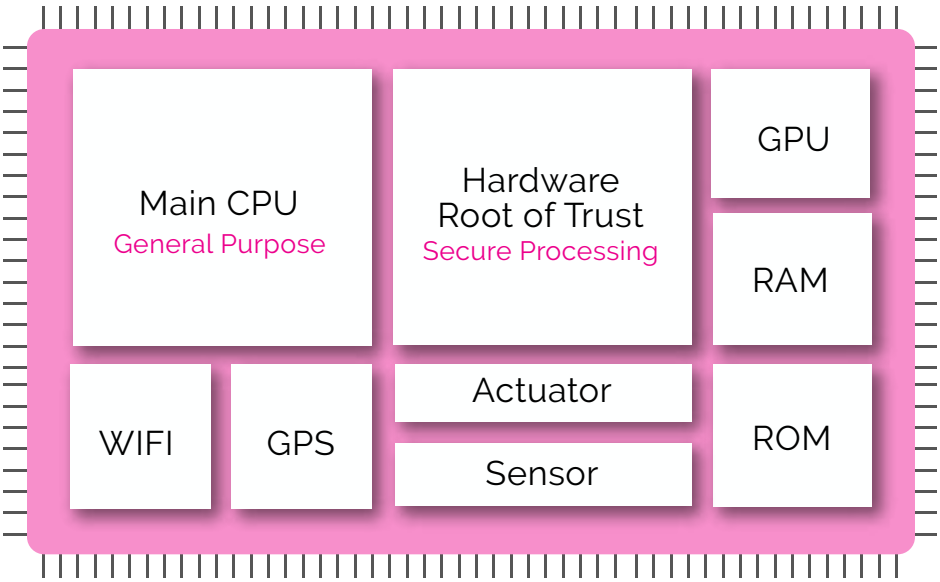

Figure 2: System on a Chip

Cryptography

Cryptography is used to achieve many of the trust attributes. Although cryptography is typically associated with protecting confidentiality by withholding secrets, it also enables a wider array of security services. Through asymmetric encryption, cryptography can be used to establish identity and provide authentication services for firmware and updates. With symmetric encryption, cryptography can be used for secure communication. Finally, with hashing algorithms, cryptography can be used to prove integrity of bootloaders, kernels, hypervisors, operating systems, and applications prior to execution.

Hardware accelerators are often used to enhance the performance of cryptographic functions. These coprocessors are more efficient than general-purpose processors. This paper will later outline how these are used in hardware implementations of roots of trust such as TTE, TPM, HSM, DICE, and Smartcards

Cryptographic keys and certificates are generally programmed into the IoT device during the manufacturing process to create trust anchors. NIST defines a trust anchor as "A public or symmetric key that is trusted because it is directly built into hardware or software, or securely provisioned via out-of-band means, rather than because it is vouched for by another trusted entity (e.g. in a public key certificate)" (NIST, 2015). Much like ships use an anchor to keep from drifting at sea, IoT devices can use trust anchors to keep their identity from being lost in the endless ocean of the Internet.

While simple root of trust implementations can have a symmetric key embedded in a microprocessor, most implementations utilize Public Key Infrastructure (PKI). PKI is a technology used to establish strong identity and device authentication. It relies on asymmetric encryption to create trust anchors.

Device manufacturers have several options available for using PKI, from setting up a proprietary environment to leasing PKI services from an established vendor. A risk assessment should be conducted before deciding which model to use; however, cost usually dictates that companies will use an established Certificate Authority instead of standing up one of their own. Regardless of choice, consideration should be given to rotating certificates and keys, especially if the device is expected to have a long lifetime.

Identity

Identity is one of the key trust attributes. It is important that each device in the IoT ecosystem be uniquely recognized; however, establishing device identity for a sea of ubiquitous devices has not always been easy. PKI is changing that through the use of digital certificates, the new defacto standard for IoT device identity. Digital certificates provide a unique electronic identity that can be easily authenticated. If certificates are compromised, they can be revoked by a central authority, thus preventing device identity theft.

IoT device identities can be created either during the manufacturing process or in the field. If device identity is created in the manufacturing process, this typically occurs when crytographic keys are injected into a onetime programmable secure element. Care needs to be taken to ensure this process is secure, including the secure handling of keys, secrets, and intellectual property.

If the supply chain cannot be trusted to securely inject keys, or the process is cost-prohibitive due to limited production runs, then cloud-based field provisioning may be a suitable option. Cloud-based provisioning allows for remote identity creation after the device has entered the field. This relies on the creation of a minimal bootstrap during production, which can be used by the cloud service, to properly authenticate and authorize the device.

Authentication | *Symmetric Key*

With symmetric keys, the same key that is used to encrypt is also used to decrypt. Symmetric encryption is very fast compared to asymmetric alternatives. Encryption speed is the main advantage of using this type of encryption, especially when large amounts of data need to be processed. The use of symmetric encryption significantly reduces the cost of IoT authentication and streamlines the device provisioning process; although, not without some hurdles.

There are several problems with symmetric encryption. One of the main problems with using symmetric cryptography in an IoT ecosystem is key distribution, or rather, the lack of a defined process to securely distribute keys. Without an internal mechanism to distribute its keys securely, symmetric encryption relies on an out of band key distribution model.

This creates a vulnerability that the key can be stolen either during transit to other devices in the ecosystem or during the device provisioning process that occurs offsite at the contract manufacturer.

Another problem is key management. As the number of IoT devices increase in an ecosystem, the number of symmetric keys also increase. This is generally represented mathematically with the following equation: n(n-1)/2 where n equals the number of devices in the ecosystem. For example, if ten devices are added to the ecosystem that all need to communicate with each other, then 10(10-1)/2 would equal 45 different keys that would need to be distributed and managed.

The last issue with symmetric encryption is that it only supports confidentiality. While this is a trust attribute, it is a single attribute. Asymmetric encryption provides authenticity and nonrepudiation, along with confidentiality. With symmetric encryption, there is no certain way to tie a cryptographic operation back to a single entity, since by definition, more than one entity has ownership of the key.

Reliance on symmetric encryption could indirectly lead to insecure practices. For example, to resolve key distribution issues, a company may decide to embed a non-unique symmetric key directly in the silicon of each device, relying on the hardware security to protect it. While this obviously reduces complexity during the device provisioning process, it leaves the entire IoT ecosystem vulnerable because the strength of the entire ecosystem is only as strong as its weakest point. A successful attack and compromise of the shared key would lead to the exposure of the entire ecosystem.

Authentication | *Public Key Infrastructure*

PKI is a system of hardware roots of trusts, software, and policies to create, manage, distribute, and revoke X.509 digital certificates. PKI is used for a myriad of reasons, including encrypting web traffic from e-commerce and emails as well as securing Virtual Private Network (VPN) connections. Over the last several years, PKI has risen to become a popular mechanism for securing IoT environments.

PKI is based on asymmetric cryptography, which uses two mathematically linked keys for encryption and decryption. One key is private, and the value is only known by the key owner. The other key is public, and no effort is made to hide the key's value. Items encrypted by one key can only be decrypted by the other key in the linked key pair.

Because asymmetric encryption is slower than symmetric encryption, it is not generally used to encrypt large amounts of data. If a large amount of data needs to be encrypted and sent between two devices, usually an asymmetric algorithm will be used to securely exchange a symmetric key that both devices know. Once the exchange has completed, the encryption will occur using a symmetric algorithm.

Digital certificates are used to establish trust between two entities. This occurs because both entities trust an independent third party who vouches for the identity of one of the entities. This trusted source is called a Certificate Authority (CA). Before issuing a new certificate, the CA performs a strong verification of the identity of the requesting entity who will own the certificate. The certificate contains identifying information about the entity, such as their name and email, along with their new public key. The new certificate is then digitally signed with the CA's private key to prove authenticity.

PKI provides IoT devices in the ecosystem with a mechanism for both mutual authentication and secure updates. IoT devices can authenticate servers within the IoT ecosystem to ensure that updates and other key services are authentic. Servers and services can use IoT certificates to ensure devices are authorized to communicate, request and receive services, and to update data.

PKI use within IoT is not without challenges. Many devices manufacturers have to rely on external vendors to serve as Certificate Authorities. Since many IoT devices have long lifetimes, certificate lifecycle management will need to be considered during the product design stage. If a device manufacturer chooses to stand up their own CA to provide PKI services, then that environment will need to be provisioned and maintained outside of a specific IoT product lifecycle.

Immutable Code

Immutable code is used to achieve trust attributes as well. Injecting boot code into the device during the manufacturing process creates a durable software anchor of trust, which is firmly rooted in hardware. Since the initial code cannot be changed after device provisioning, the software can be used as a known good starting place. Immutable code is also used to verify other steps in the boot process and to create a transitive chain of trust.

Immutability creates a state that cannot be easily changed. Immutable code occurs when code is burned into the silicon chip or stored in a manner in which only an appropriate cryptographic key can be used to change the code. Code protected by immutability properties is typically run directly after a device is powered on or a reset occurs and is used as a component of secure boot. If the software used to initialize the device during the first stage of booting is vulnerable to manipulation, then no other stage can be trusted.

Immutable code allows the device to be reset to a known good configuration. If the device is compromised due to malware then being able to reset the device to a condition that can be trusted is very valuable. This is especially true since many IoT devices are deployed into hostile environments.

Not all immutability is desirable, and when applied to the wrong control, it can have disastrous effects. For example, hardcoding immutable credentials into the device is worse than not having any credentials because it gives a false sense of security. Making all device code and applications immutable would inhibit the secure update process and prevent vulnerabilities from being patched.

Secure Boot

Secure boot is an important core component of establishing a root of trust. Secure boot helps an IoT device become resistant to attack by ensuring the system will only boot using trusted firmware. It also protects the confidentiality of intellectual property such as firmware code and embedded software.

IoT Foundations of Trust

Developed properly, secure boot establishes a mechanism for trusted remediation. In cases of device compromise or failure, the device can reset itself by booting to a known good state. This mechanism relies on both immutable starting code stored in a secure location and the ability to verify the trustworthiness of the next code before it is executed. A strong foundation of trust is vital to establishing a transitive chain of trust, where the trustworthiness of a latter section of the system builds upon the trustworthiness of the preceding section.

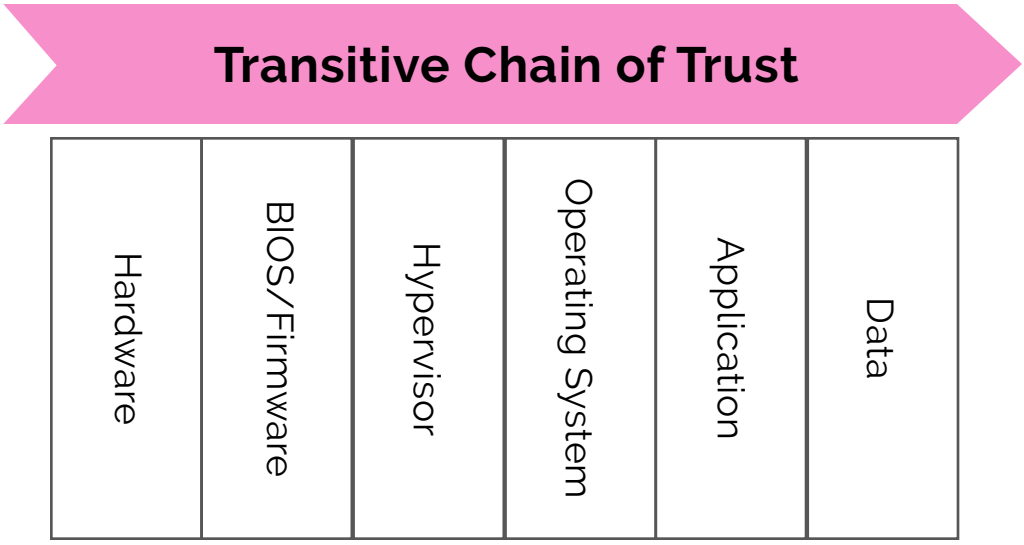

Figure 3: Transitive chain of trust built from trusted hardware and extending throughout the device (Qin, Chang, Shen, & Gao).

Isolated Environments

Isolation is also an important technical concept for achieving trust attributes and establishing a root of trust. Isolating data, code, and execution environments helps to establish device integrity and confidentiality, and ensures that only authorized transactions occur. Isolation can occur within a single processor, such as a Trusted Execution Environment (TEE) implementation. It can also occur with a dedicated security hardware module such as an HSM embedded in a microcontroller or a SoC, or as a Trusted Platform Module (TPM) in which the security hardware module is paired with a microchip.

Isolation is also essential in building a layered defense. Establishing multiple roots of trust allows different entities, including chip providers, Original Equipment Manufacturers (OEM), service providers, and end users to manage their own security domains. This process uses isolation and strong hardware security to enforce security boundaries around entities while only allowing approved actions (Levine, Strong Security, 2019).

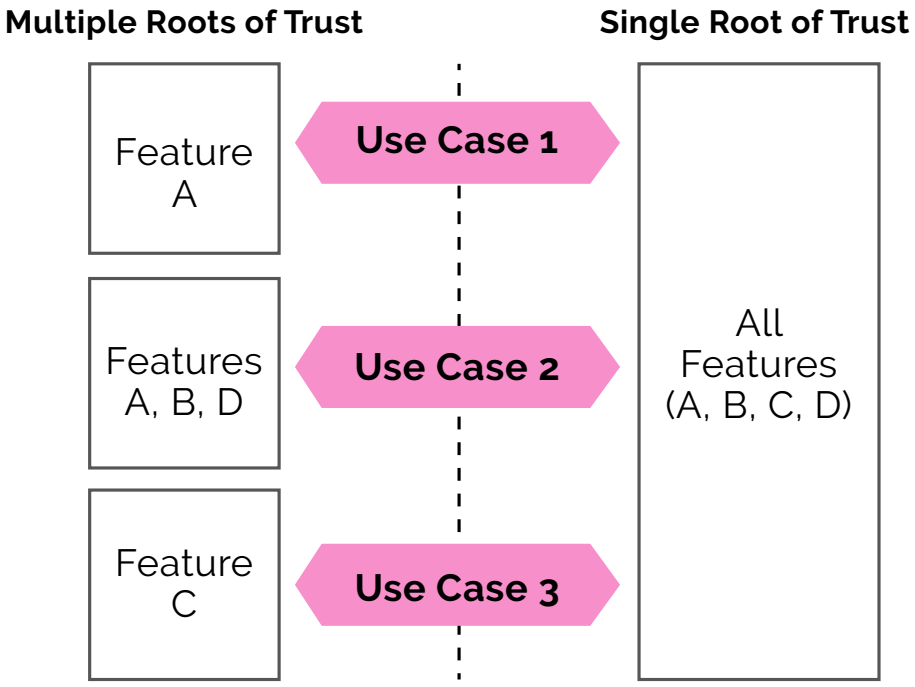

Figure 4: Multiple Roots of Trust [Based on the following drawing: (Levine, Will the Real Root, 2019)]

Implementations of Hardware Root of Trust

There are two general classes of hardware roots of trust: fixed-function and programmable (Levine, Will the Real Root, 2019). Fixed-function hardware roots of trust are characterized as being state-based with limited basic cryptographic functionality such as encrypting, decrypting, and signature verification. Later updates or changes to these functions is not possible. Programmable roots of trust are characterized as being able to perform higher-level cryptographic functions in addition to the more basic ones provided by their more limited fixed-function counterpart. This is due to the inclusion of a silicon chip, which allows the system to be changed over time (Rambus Press, 2019).

IoT Foundations of Trust

Programmable roots of trust allow for maximum flexibility when services and operations need to be performed in a trusted execution environment. Even if a product is released with a minimal amount of services, systems should be created so that additional functionality can be built-in. These changes may include extending cryptographic functions, performing secure software updates, provisioning additional identities, or changing cloud configuration data.

There are different hardware implementations of root of trust. This paper will identify the most commonly used ones in IoT ecosystems and compare and contrast them. This includes the Trusted Execution Environment, Trusted Platform Module, Hardware Security Module, Device Identifier Composition Engine, and Smartcards.

Trusted Execution Environment

A TEE is an isolated processing environment that separates secure resources from non-secure resources. A well-designed TEE will consist of the following security principles: secure boot, operating system isolation, application isolation, controlled access to hardware, and a tamper - resistant secure element capable of protecting cryptographic secrets and executing code securely (Secure Technology Alliance, 2018). Together, these principles enable a TEE to create an environment where trusted applications and data are separated from their untrusted counterparts.

Software running on a non-secure processor allows other applications or individuals with root access to investigate or manipulate the code and data running on the device. TEE addresses this vulnerability by enforcing confidentiality and integrity during program execution. The TEE divides the device into two trust zones, a “Normal World” and a “Secure World”. The Normal World represents the non-secure hardware and untrusted software on the device while the Secure World represents the protected hardware and trusted software. This isolation boundary prevents trusted resources from intermingling with untrusted or other trusted resources (Prado, 2020).

Trusted applications and data are isolated from untrusted applications outside the TEE and other trusted applications inside the TEE. This means that secrets and private data that is processed or stored in one security domain cannot be accessed from another security domain. This isolation also creates device stability because one application cannot impact the performance of another one (Simon & Stumann, 2019).

ARM Trusted Zone

ARM Trusted Zone is an implementation of TEE as an optional hardware security extension available for ARM processors. TrustedZone implements a secure world and non-secure world for code, memory, and peripherals and includes secure boot to establish a root of trust. There is currently wide support for this across many ARM processors.

TEE Secure Boot

TEE secure boot ensures the device boots to a trusted state. The process begins with the Boot ROM, which is considered a trusted starting point because it is provisioned with the OEM boot code and installed by the manufacturer. The Boot ROM verifies the bootloader image and signature before loading. If correct, the bootloader is allowed to execute. The bootloader then executes its code, and before turning over control of the device to the Trusted OS, the bootloader verifies the Trusted OS image and signature. If correct, the Trusted OS is allowed to execute. The Trusted OS then launches Trusted Apps within the secure world and the Rich OS bootloader in the insecure environment within the normal world (Felton, 2019).

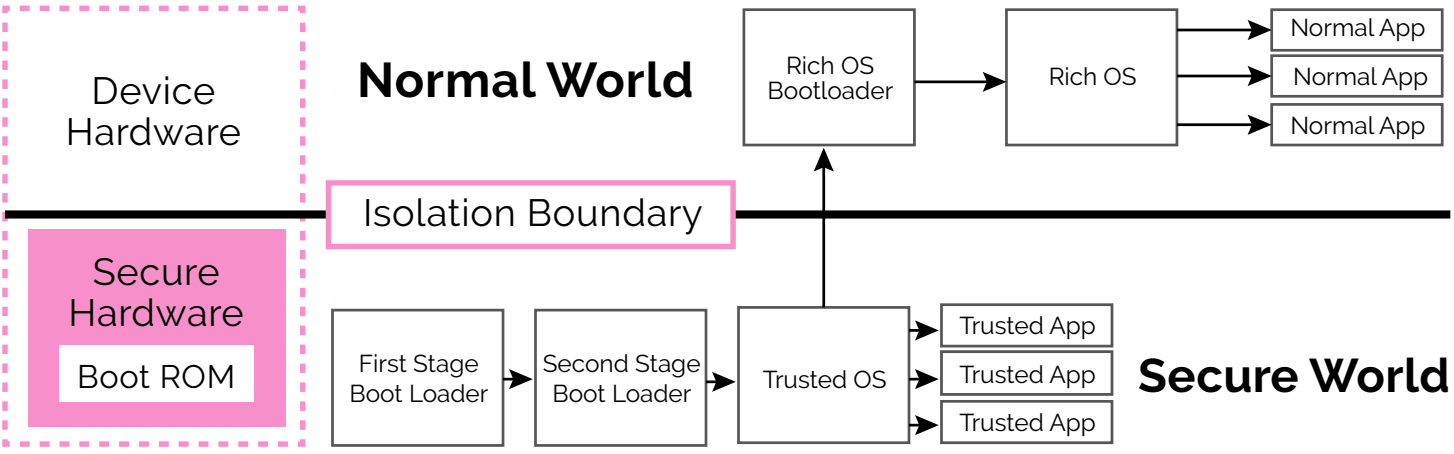

Figure 5: TEE Secure Boot Process [Based on the following drawing: (Felton, 2019)]

The example above illustrates a TEE with an operating system and user-installed applications. While some IoT devices may have this level of complexity designed in them, many IoT devices do not have a fully developed operating system or allow for user-installed applications. Device OEMs may also use the TEE as a hardened execution environment for the firmware to interface directly with hardware. It is important to note that a TEE does not create as strong of a root of trust as found in specialized hardware-based roots of trust such as TPM, HSM, or Smartcards (Simon & Stumann, 2019).

Trusted Platform Module (TPM)

The Trusted Computing Group created the Trusted Platform Module (TPM) and released it for use in 2003. TPM specifications are more restrictive than TEE (Simon & Stumann, 2019) because they are designed for specific security processing workloads. TPMs are widely used and have been deployed on nearly every computer in the last decade and support a variety of applications from disk encryption to VPNs.

TPM chips have to be physically added to the system during manufacturing. Unlike HSMs, TPMs cannot be retroactively added to systems, nor can they be removed at a later time, because the chip is usually soldered to the motherboard. This design requirement underscores the importance of designing security in from the beginning of the product lifecycle.

One of the vulnerabilities of secure elements is the unsecured communication path between the processor and the secure element. TPM is a secure element implementation that establishes trust and enables secure communications. This solves some of the communication vulnerabilities of a basic secure element since TPM adheres to a set of security protocols that govern secure communication between the components.

One of the design considerations for TPM is that it is passive. It is not designed to run in the background constantly monitoring the security of the system. To utilize the capabilities of the secure chip, interaction has to occur between the TPM and the processor (Singer, 2019). In other words, the TPM functionality has to be actively called in order to be utilized.

TPM uses endorsement keys and storage root keys. An endorsement key is a unique RSA key burned into the silicon chip during production. This key is used to encrypt data. The storage key is an RSA key created by the TPM when configuring ownership and is used to protect other key material generated by the TPM. While the endorsement key cannot be removed, the storage key can be.

The main purpose of the TPM is to protect the host platform. It offers secure storage for cryptographic keys, protected execution of cryptographic functions, and attestation of both the TPM and the host platform. It does this by measuring software to determine validity (Singer, 2019).

TPM does not offer a secure boot process; however, it relies on secure boot provided by the system. To secure the boot process, the initial boot process "needs to perform a minimum level of initialization to verify the next piece of boot code before handing off to that code" (The Chromium Projects, 2020). This initial code is stored in protected memory.

Measured Boot

TPM supports a process called measured boot. Also referred to as a trusted boot, this process measures each of the components in the next stage of the boot process to verify the component's integrity. The overall purpose of a measured boot is to detect unauthorized software or configurations.

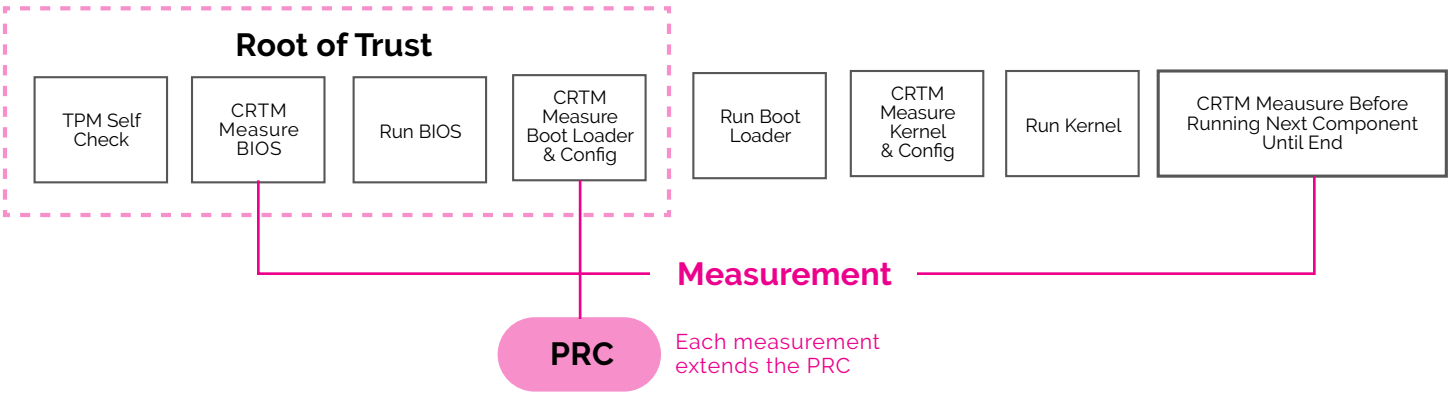

Figure 6: TPM Boot Process [Based on following drawings: (Fedorkow, 2015), (Boneh et al., 2020), (Stamm et al., 2008)]

The TPM measured boot process begins when the device is booted. The first action TPM takes is to conduct a self-test. When completed, the Core Root of Trust Measurement (CRTM) will measure the BIOS and send the hash value to the memory location 0 in the Platform Configuration Register (PCR) bank. The PCR acts as a retainer for measurements which represent a safe boot process. The PCR will then verify that the measurements are correct. If the measurements do not match, then a rollback procedure is called to return the module to the Last Known Good and move to the next measurement step. This process continues with the TPM measuring the boot loader next and then the next component and so forth, with each preceding component in the boot chain describing the following component until the device has successfully booted (Russinovich et al., 2012).

Unlike a secure boot process, a measured boot does not inhibit the usage of the system. The TPM process tracks the last known good state of the system. Advanced implementations can roll back modules that have aberrations from previous measurements. An IoT device's ability to "self-heal" by rolling back to the last known good state is a vital attribute, especially when most IoT devices are deployed outside of a defined security perimeter (Digi-Key, 2015).

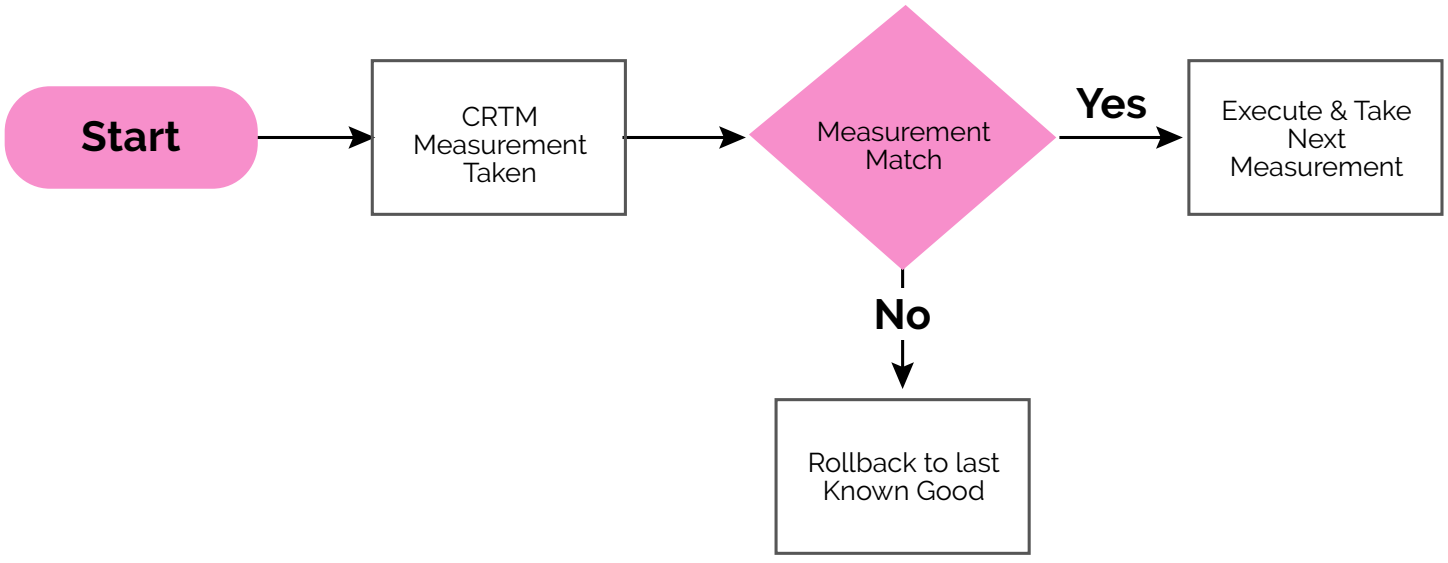

Figure 7: Measured Boot Decision Process [Based on the following drawing: (Trusted Computing Group, 2014)]

Hardware Security Module (HSM)

The term HSM is used to describe "secure hardware that does not adhere to a specific security protocol" (Microsoft, 2017); however, true HSMs are standards-based and are formally validated against the FIPS 140-2 standard. Like TPMs and smartcards, HSMs provide cryptographic services and secure storage of cryptographic material. They handle the full range of cryptographic key management from provisioning, managing, and storing keys to key archival and deletion. These devices also maintain their secrets in a hardened tamper-resistant physical and logical vault.

Unlike TPM, both HSM and TEE chips can be programmed for general purpose use; however, HSM implementations are usually orders of magnitude more expensive than TEE chips (Simon & Stumann, 2019). HSMs are physical devices that offload cryptographic functionality from general-purpose processors. They can be a single-chip or multi-chip implementation. Multi-chip HSM implementations can be standalone and portable such as a PCIe card or a USB dongle, or they can be integrated as part of a System on a Chip.

HSMs serve many different purposes that are not always apparent. For example, when individuals browse encrypted websites, the HTTPS protocol that protects the communication between the client and server works because the Certificate Authority uses an HSM as a root of trust for storing its private key (CA/Browser Forum, 2020). HSMs are also used as payment card systems, banking systems, and cryptocurrency wallets.

HSMs provide a means to ensure identity is properly injected into a device. Identity is fundamental to IoT security and a building block for security within the larger IoT ecosystem. Being able to identify a smart lightbulb from a smart thermostat is understandably important; however, authenticating and authorizing two individual smart lightbulbs of the same model is more valuable. Finally, being able to identify smart lightbulb running authorized updates from a smart lightbulb running malware may be the most beneficial.

HSMs also help protect against counterfeiting. Many times a developer will have to deliver their intellectual property to a contract manufacturer to have the code injected into IoT device.

HSMs ensure the integrity and confidentiality of IoT firmware and application code during the manufacturing process by protecting secrets stored in physically remote third-party-owned hardware. This prevents critical intellectual property from being reverse-engineered into competing products.

Cloud HSM

Cloud Hardware Security Modules are HSM appliances that achieve trust attributes by implementing a cloud-based root of trust. Cloud providers offer products that are scalable and fully managed, with services such as cryptographic operations and secure storage for cryptographic keys. These services are designed to meet regulatory requirements and contractual agreements such as FIPS 140-2 Level 3, which is required for electronic payments, document signing, and operating as a public Certificate Authority.

Secure Boot

To fully protect a system, the device has to be protected in all states, including when the device is powered off, during the boot process, while receiving updates, and when running in normal operational mode. HSMs support secure boot in order to create a mechanism where only trusted code is loaded and executed during the initial system boot process while also preventing the disclosure of embedded code. Secure boot is an important defense in depth control because it assumes the device will eventually be compromised and provides a mechanism to reset the device back to a known good state in a manner that can also provide proper attestation after device reset (Povey, 2017).

HSM secure boot uses asymmetric encryption and PKI to implement digital signatures in order to establish chains of trust throughout the device boot cycle. Digital signatures use private keys, which are kept secret and never distributed, and public keys, which are freely disseminated. Both public and private keys are mathematically linked; however, one key cannot be calculated by studying the contents of the other key.

IoT Foundations of Trust

To set up secure boot properly, the OEM will use its private key to create a digital signature of the software that will be used by the device during the boot process (Young, 2018). During the manufacturing process, these signatures, along with the first stage bootloader code, are stored on device hardware in non-writable protected memory. This process creates a root of trust for the device.

During boot, the device loads the trusted code stored during the manufacturing process. The cryptographic system then uses the public key in BIOS to compare the signature of the second stage bootloader with the second stage bootloader code (Fedorkow, 2015). If the comparison matches, then the system determines the second stage bootloader can be trusted, and the boot process moves forward to the next stage. This signing and verification continue throughout the boot process from the firmware and bootloader to the kernel and modules, creating a secure chain of trust (Trudel-Lapierre, 2017).

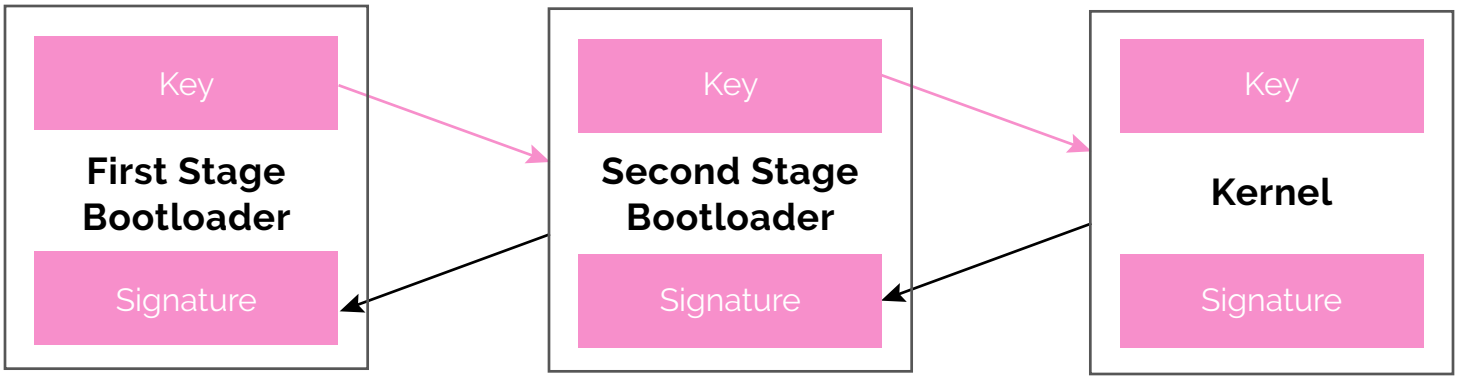

Figure 8: Secure Boot Chain of Trust [Based on the following drawing: (West, 2017)]

Both measured boot and secure boot rely on an established root of trust to build a chain of trust. As described in the previous section, secure boot works by using cryptographic keys to check the signature of the firmware, boot loader, kernel, and modules. If the signature is invalid, the boot process halts. Since the OEM's private key is used to create the signature, the secure boot has limited usability beyond these first stages in the boot process. For example, it would be onerous to expect the OEM to sign system configuration files (Fedorkow, 2015).

Unlike secure boot, measured boot does not inhibit the boot process. As the name implies, measured boot creates a process to measure and store the hash of an object in a chain of trust. Compared to secure boot, this measurement process provides more granularity in what objects can be evaluated, for example, allowing for measurement of configuration files (Fedorkow, 2015).

Device Identifier Composition Engine

TPM and HSM are not practical solutions for many IoT devices, embedded systems, and sensors. To this end, the Trusted Computing Group created the Device Identifier Composition Engine (DICE). DICE is a combination of hardware and software that creates a lightweight root of trust (Mattoon, 2018) at a near zero-dollar cost. It is especially beneficial for adding security services to resource-constrained sensors and devices found in the Industrial Internet of Things (IIoT).

DICE is an immutable process injected into the IoT device during the manufacturing process. It allows for hardware-based identification and authentication, and since DICE hardware requirements are minimal, many existing processors with embedded cryptographic functions can be used. DICE uses the terminology Unique Device Secret (UDS) to describe the cryptographic key embedded in the processor (Trusted Computing Group, 2018). The picture below illustrates the layered secure boot process used by DICE.

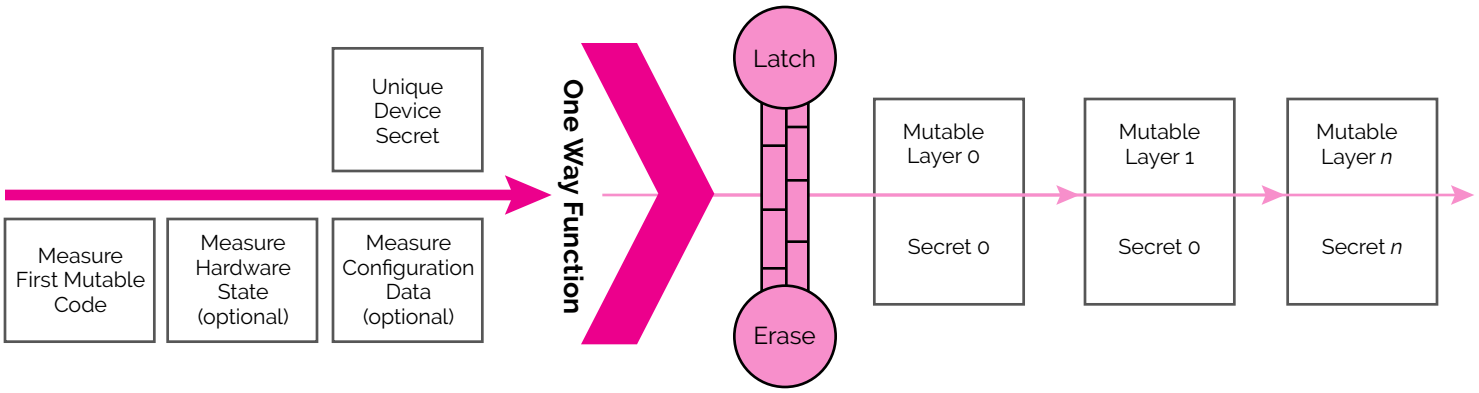

Figure 9: DICE Lightweight Secure Boot [Based on the following drawing: (Trusted Computing Group, 2018)]

DICE is started at power-on or device reset and has exclusive read access to the UDS. The first step in the process begins by taking a measure of the first mutable code. This computed measure and the UDS are passed through a one-way function to create a Compound Device Identifier (CDI). A hardware latching mechanism is engaged to prevent read access to the UDS. The UDS is then securely erased from memory. The CDI is then passed to the next layer of the secure boot process along with control of the boot process. Each subsequent layer receives a secret from the preceding layer and generates a new secret for the next layer (Trusted Computing Group, 2018).

DICE employs three techniques to limit access to the UDS. First, hardware uses a power-on latch to disable read-access to the UDS prior to executing firmware. Second, cryptographic one-way functions are used to protect the UDS. Third, the CDI is based on the first mutable code used (England et al., 2015).

The first technique protects the confidentiality of the UDE by ensuring it cannot be accessed after it is used to create the CDI. The second technique prevents the UDE from being scraped from memory or derived by reverse-engineering the CDI (assuming the underlying cryptography is sound). The third technique prevents malware from revealing the device keys since the act of injecting the malware code will change the CDI. The third technique also ensures that valid updates to address vulnerabilities, which would reveal the UDI, automatically force device rekeying (England et al., 2015).

Smartcards

Smartcards are single-chip cryptographic solutions, generally optimized for performance (Rosenau, 2018), and used when strong authentication is required. In an IoT environment, Smartcards can be used to authenticate a user to a device or application within the IoT ecosystem. This would add another layer of defense to the overall security and privacy of the system.

These devices work by connecting to a reader through physical contact or remotely via radiofrequency. Data is stored in non-volatile memory of the Smartcard, which is activated when an external power source is supplied. Smartcard architecture uses either a secure memory integrated chip or a secure microcontroller.

Secure memory chips are less secure than secure microcontrollers. Secure memory chips share some of the same characteristics as TPM in regards to cryptography and secure storage (Rosenau, 2018). Symmetric encryption is used to protect data in transit from the card to the reader. While some secure memory chips protect against unauthorized read, most are used in applications with minimal data protection requirements (Smart Card Alliance, 2008).

Secure microcontrollers offer more functionality because they combine the chip with additional hardware, such as RAM, ROM, and I/O units, along with operating code. Secure microcontrollers are dissimilar to TPMs in that they have an independent operating system and can execute dynamic applications that enable them to interact intelligently with a reader (Rosenau, 2018). They also support advanced cryptographic functions that allow this type of smartcard to utilize symmetric and asymmetric cryptography (Smart Card Alliance, 2008).

Summary

Proper security within an IoT ecosystem is achieved by creating a foundation and environment where trust attributes can be established. This occurs through proper use of secure elements, cryptography, immutable code, secure boot, and isolated environments. IoT manufacturers can deliver secure devices across industrial, commercial, and consumer markets by designing security into products and appropriately using common implementations of hardware roots of trust, including Trusted Execution Environment, Trusted Platform Module, Hardware Security Module, Device Identifier Composition Engine, and Smartcards.

References

ADT. (2019, January 28). ADT Announces New Consumer Privacy Initiative. Adt.Com. https://investor.adt.com/press-releases/press-release-details/2019/ADT-Announces-New-Consumer-Privacy-Initiative/default.aspx

Boneh, D., Bursztein, E., & Mitchell, J. (2020). Trusted Computing Architecture. CS155. https://crypto.stanford.edu/cs155old/cs155-spring11/lectures/08-TCG.pdf.

Business Wire. (2019, December 10). Karamba Security Survey: Consumers Hold Vendors Accountable for Their Devices' Cybersecurity. Businesswire.Com. https://www.businesswire.com/news/home/20191210005192/en/-Karamba-Security-Survey-Consumers-Hold-Vendors-Accountable

CA/Browser Forum. (2020). CA/Browser Forum Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates. https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.6.9.pdf#page=48

Chen, L., Franklin, J., & Regenscheid, A. (2012). Guidelines on Hardware- Rooted Security in Mobile Devices (Draft) Recommendations of the National Institute of Standards and Technology. In NIST Special Publication 800-164. https://csrc.nist.gov/CSRC/media/Publications/sp/800-164/draft/documents/sp800_164_draft.pdf

Computer History Museum. (2020). 1974: Digital Watch is First System-On-Chip Integrated Circuit | The Silicon Engine | Computer History Museum. Computerhistory.Org. https://www.computerhistory.org/siliconengine/digital-watch-is-first-system-on-chip-integrated-circuit/

Digi-Key. (2015, April 16). Hardware Security for a Worldly-Wise Internet of Things. Digikey.Com; Digi-Key's European Editors. https://www.digikey.com/en/articles/hardware-security-for-a-worldly-wise-internet-of-things

D'mello, A. (2019, February 11). The real-life applications of IoT and why battery life is critical - IoT Now - How to run an IoT enabled business. IoT Now. https://www.iot-now.com/2019/02/11/92898-real-life-applications-iot-battery-life-critical/

England, P., Mattoon, D., Spiger, R., Thom, S., Marochko, A., Peinado, M., & Kane, K. (2015, January 1). DICE: Device Identifier Composition Engine - Microsoft Research. Microsoft Research. https://www.microsoft.com/en-us/research/project/dice-device-identifier-composition-engine/

Fedorkow, G. (2015, September 17). What's the Difference between Secure Boot and Measured Boot? Juniper. https://forums.juniper.net/t5/Security/What-s-the-Difference-between-Secure-Boot-and-Measured-Boot/ba-p/281251

Felton, D. (2019, July 5). Trusted Execution Environment (TEE) - What is it? Trustonic. https://www.trustonic.com/news/technology/what-is-a-trusted-execution-environment-tee/

References

Garibay, T. (2017). Implementing Machine Learning & Neural Network Chip Architectures Using Network-on-Chip Interconnect IP Implementing Machine Learning & Neural Network Chip Architectures Using Network-on-Chip Interconnect IP. In Arteris.
https://www.arteris.com/hubfs/docs/ArterisIP%20-%20implementing%20machine%20learning%20and%20neural%20network%20chip%20architectures%20-%20notes%20pages.pdf?utm_campaign=ai%20machine%20learning&utm_source=email&utm_content=Download%20Implementing%20ML%20and%20AI%20Architectures

Hanna, S. (2016, July 26). Hardware is the Foundation of IoT Security. Globalsign.Com.
https://www.globalsign.com/en-sg/blog/iot-security-hardware/

Harding, S. (2019, September 11). What Is an SoC? A Basic Definition. Tom's Hardware; Tom's Hardware.
https://www.tomshardware.com/reviews/glossary-soc-system-on-chip-definition,5890.html

Krebs, B. (2018, October 18). Supply Chain Security is the Whole Enchilada, But Who's Willing to Pay for It? — Krebs on Security. Krebsonsecurity.Com.
https://krebsonsecurity.com/2018/10/supply-chain-security-is-the-whole-enchilada-but-whos-willing-to-pay-for-it/

Levine, B. (2019, June 10). Strong security requires multiple roots of trust - Embedded.com. Embedded.
https://www.embedded.com/strong-security-requires-multiple-roots-of-trust/

Levine, B. (2019, June 20). Will the Real Root of Trust Stand Up? EEWeb Community.
https://www.eeweb.com/profile/blevine/articles/will-the-real-root-of-trust-stand-up

Mattoon, D. (2018, August 31). Securing Industrial IoT Sensors, part 3: Foundational Trust for IoT. Embedded-Computing.Com; Securing Industrial IoT Sensors, part 3: Foundational Trust for IoT.
https://www.embedded-computing.com/guest-blogs/securing-industrial-iot-sensors-part-3-foundational-trust-for-iot

Mendoza, N. F. (2020, February 20). Data privacy: What consumers want businesses to know. TechRepublic;
https://www.techrepublic.com/article/data-privacy-what-consumers-want-businesses-to-know/

Microsoft. (2017). The Right Secure Hardware for your IoT Deployment (pp. 1–9). Redmond, WA.
NIST. (2015). Trust Anchor - Glossary. National Institute of Standards and Technology.
https://csrc.nist.gov/glossary/term/trust_anchor

Povey, H. (2017). Enabling a Secure IoT From Design to Deployment. Iotsecurityfoundation.Org.
https://www.iotsecurityfoundation.org/talks-iotsf-conference-2017

References

Qin, X., Chang, C., Shen, C., & Gao, L. (n.d.). A Novel Trusted Terminal Computer Model Based on Embedded System. Information Technology Journa, 9, 730–739. doi: 10.3923/itj.2010.730.739

Rambus Press. (2019, September 10). Hardware Root of Trust - Everything you need to know | Rambus. Rambus. https://www.rambus.com/blogs/hardware-root-of-trust/

Rosenau, D. (2018). The Benefits of a Hardware Security Module in Industrial IoT Applications. FP InovoLabs GmbH. Retrieved from https://cdn0.scrvt.com/fa130d14754799-caab3c404eab497d13/a39a6e45ec63ba9e/b6b0995668f3/FP_Whitepaper_Hardware-Security-Module_IoT40_EN_11052018.pdf

Russinovich, M. E., Solomon, D. A., & Ionescu, A. (2012). Bitlocker Boot Process. In Windows Internals Part 2 (6th ed., pp. 170–171). Microsoft Press.

Sabev, A. (2017, August 2). Pros & Cons of Secure Elements. Intrinsic ID. https://www.intrinsic-id.com/pros-cons-secure-elements/

Secure Technology Alliance. (2018). Trusted Execution Environment (Tee) 101: A Primer. Retrieved from https://www.securetechalliance.org/wp-content/uploads/TEE-101-White-Paper-FINAL2-April-2018.pdf

Secure Thingz. (2017). The Keys to Establishing a Chain of Trust. In Secure Thingz IoT Security. https://www.securethingz.com/siteassets/blocks/learn/articles/the_keys_to_a_chain_of_trust_secure_thingz.pdf

Simon, A., & Stumann, L. (2019, December 2). Current Trusted Execution Environment landscape. Redhat; now + Next. https://next.redhat.com/2019/12/02/current-trusted-execution-environment-landscape/

Singer, A. (2019, February 7). Trusted Platform Modules: 8 Surprises for IoT Security. IoT World Today. https://www.iotworldtoday.com/2019/02/07/trusted-platform-modules-8-surprises-for-iot-security/

Smart Card Alliance. (2008). What Makes a Smart Card Secure? A Smart Card Alliance Contactless and Mobile Payments Council White Paper. https://www.securetechalliance.org/resources/lib/Smart_Card_Security_WP_20081013.pdf

Stamm, S., Sheppard, N. P., & Safavi-Naini, R. (2008). Implementing Trusted Terminals with A and SITDRM. Electronic Notes in Theoretical Computer Science, (197), 73–85.

The Chromium Projects. (2020). Firmware Boot and Recovery. Chromium.Org. https://www.chromium.org/chromium-os/chromiumos-design-docs/-firmware-boot-and-recovery?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1

Trudel-Lapierre, M. (2017, August 11). How to sign things for Secure Boot. Ubuntu. https://ubuntu.com/blog/how-to-sign-things-for-secure-boot

References

Trusted Computing Group. (2014). Using the TPM to Solve Today's Most Urgent Cybersecurity Problems. In trustedcomputinggroup.org.
https://trustedcomputinggroup.org/wp-content/uploads/17.pdf

Trusted Computing Group. (2018). TCG Hardware Requirements for a Device Identifier Composition Engine Family "2.0" Level 00 Revision 78 TCG Published.
https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf

Villasenor, J. (2013). Compromised By Design? Securing the Defense Electronics Supply Chain Executive Summary. In Brookings.edu.
https://www.brookings.edu/wp-content/uploads/2016/06/Villasenor_HW_Security_Nov7.pdf

West, D. (2017, June 27). Hardware or Software Security: Which is right for my IoT Device? IoT Central.
https://www.iotcentral.io/blog/hardware-or-software-security-which-is-right-for-my-iot-device

Young, N. (2018, January 2). What is Secure Boot? Trentonsystems.
https://www.trentonsystems.com/blog/what-is-secure-boot