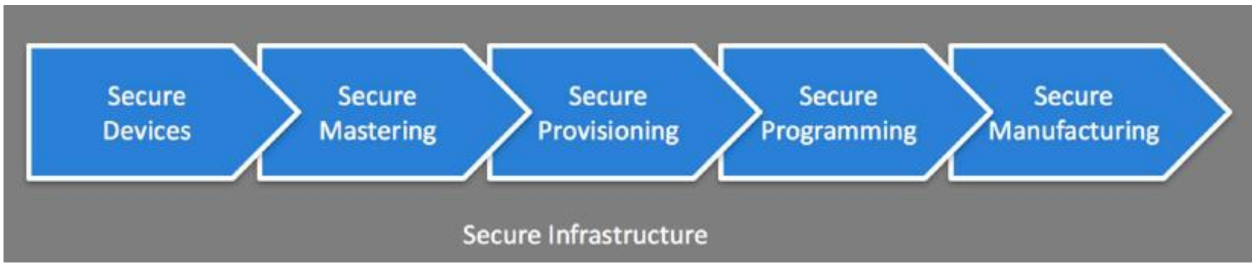
IoT Security: *The Keys to Establishing a Chain of Trust*

As the Internet of Things (IoT) proliferates, billions of cloud-connected devices are expected to be designed, manufactured and deployed over the next decade. No one wants a product or application that is prone to hacking or theft, so security is more important than ever. Yet the challenges are growing, requiring ever-increasing efforts to prevent hacking of connected devices and protect critical data and intellectual property. The industry is rapidly converging on best practices for keeping next-generation IoT devices safe by demanding a robust "chain of trust" across the entire product lifecycle.

Delivering security-oriented, embedded systems is a major challenge today. A "supply chain of trust" includes silicon vendors (for the microcontroller device), embedded software companies, programming solutions providers, and OEMs who develop the end IoT products. For truly secure products, the only real solution is to develop a "zero trust" approach across the supply chain to minimize vulnerabilities and continually authenticate and individualize deliverables as far as possible. The chain of trust can be thought of as a process flow, where any step in the process depends upon the security of the previous step. A simple flow of trust is required to engender security starting with the microcontroller (MCU) device, to how the OEM provisions the device then advancing to how the product is programmed and manufactured.



Secure Thingz is a leading provider of advanced security solutions focused on the IoT. Our goal is to make security simpler to implement, ensure it can be instilled from the very beginning, and ultimately, enhance product lifecycle management.

At Secure Thingz, we believe there are three critical areas to consider when applying trust across the supply chain to create a chain of trust:

- Secure Development
- Secure Mastering
- Secure Production

The foundation for a secure supply chain

Security needs to be architected into devices from the moment of inception, and there needs to be an ability to provide secure updates to the device across its lifecycle. The model we have adopted is based on establishing a strong “Root of Trust” (RoT) in the processor system and delivering an encrypted application image to the processor, where it is decrypted in place under specific locking conditions, thereby inhibiting both intellectual property theft and over-production or cloning.

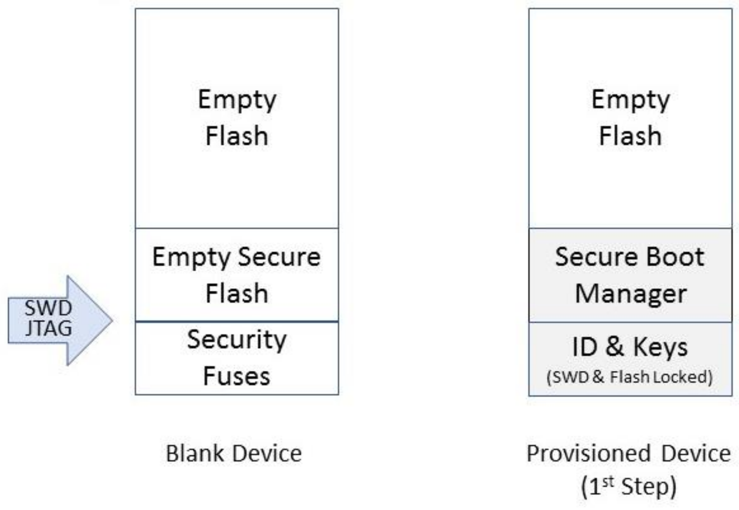
Secure development

Initially, devices need to be architected and implemented with the correct security frameworks, encompassing the root of trust, management of secrets and secure services offered to the operating system and application. In an ideal world, primary keys and certificates are provisioned at the moment of device creation ensuring that bad actors cannot syphon off devices and that every device has a known providence. Alternatively, intrinsic identifiers, such as physically unclonable functions, may be used to provide an internal unique seed.

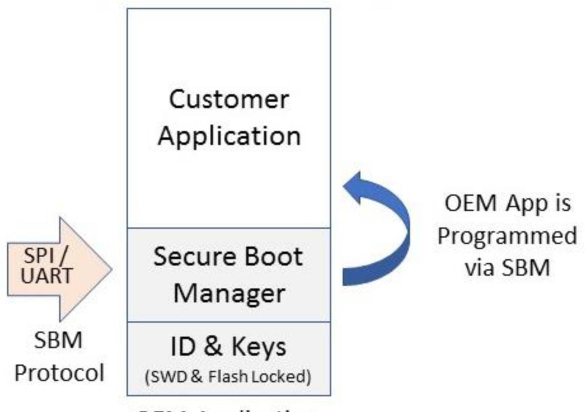
While secure authentication devices, such as Subscriber Identification Modules (SIMs) and Machine Identification Module (MIMs), exist today, their very strengths as “hard bound” devices are also their weaknesses, as they lack the flexibility to execute applications. Instead, a new generation of flexible and low-power microcontrollers are evolving into the secure foundations required to host secrets effectively while also running robust control applications.

The new generation of end devices requires a complex balancing act from the device platform, including the implementation of a robust root of trust to hold critical private key material securely, and a systematic approach to architecture that delivers confidentiality, integrity, and availability while maintaining application flexibility.

The solution begins with a Secure Boot Manager (SBM), which is injected into MCUs at birth, alongside the provisioning of secure keys and certificates that provides a robust root of trust. Early in the development phase, the OEM programs the SBM into the microcontroller using its preferred method (SWD, JTAG, other, etc.) and provisions the microcontroller’s certificates, keys and security lock bits (although this is in itself a major area of development and innovation for many organizations). At this point, the SBM is immutable and all subsequent “application programming” for the processor must be delivered via the secure process utilizing the boot manager to ensure the application code is signed and encrypted correctly.



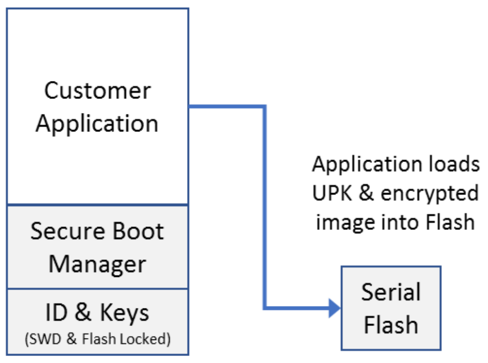
The programmer feeds the Update Protection Key (UPK) and encrypted application image to the SBM, which decrypts and verifies the UPK, then decrypts the application image and programs the flash. Finally, it updates the application signature table in secure memory.



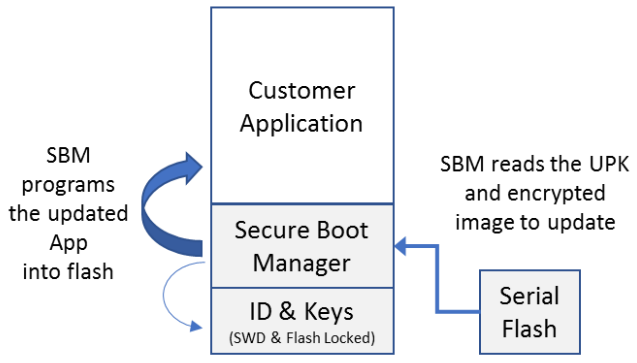
OEM Application Programming (Subsequent steps)

Every time the processor transitions through a device reset process, the SBM calculates the hash signature of the application and compares it against the values stored in the signature table to ensure nothing in the application memory has been tampered with, before running the application.

The SBM also enables a secure software update process. The customer application will download the potential software update to a separate memory location and will make a software update API call to the SBM. The SBM will reset the MCU, and after reset, it will process the requested update by first verifying it, then programming the flash with the software update.



SW Update Process (step 1)


SBM programs the updated App into flash
Customer Application
Secure Boot Manager
ID & Keys
(SWD & Flash Locked)
SBM reads the UPK and encrypted image to update
Serial Flash

SW Update Process (step 2)

This framework is the start of a more robust solution to deliver trusted IoT devices that can be securely manufactured and updated across the product's lifecycle. Runtime security services are provided by the boot manager to process software updates, and enable certificate access, validation and authentication for customer applications. This combined with a secure software encryption and delivery mechanism enables software to securely be manufactured and/or updated over the web for the lifecycle of the product.

Secure mastering

Secure Development is the first step in creating a chain of trust, and we discussed the role of the Secure Boot Manager (SBM) in the MCU device. The next step involves Secure Mastering of the software application image so it will be accepted by the SBM with the ultimate goal of ensuring Secure Production.

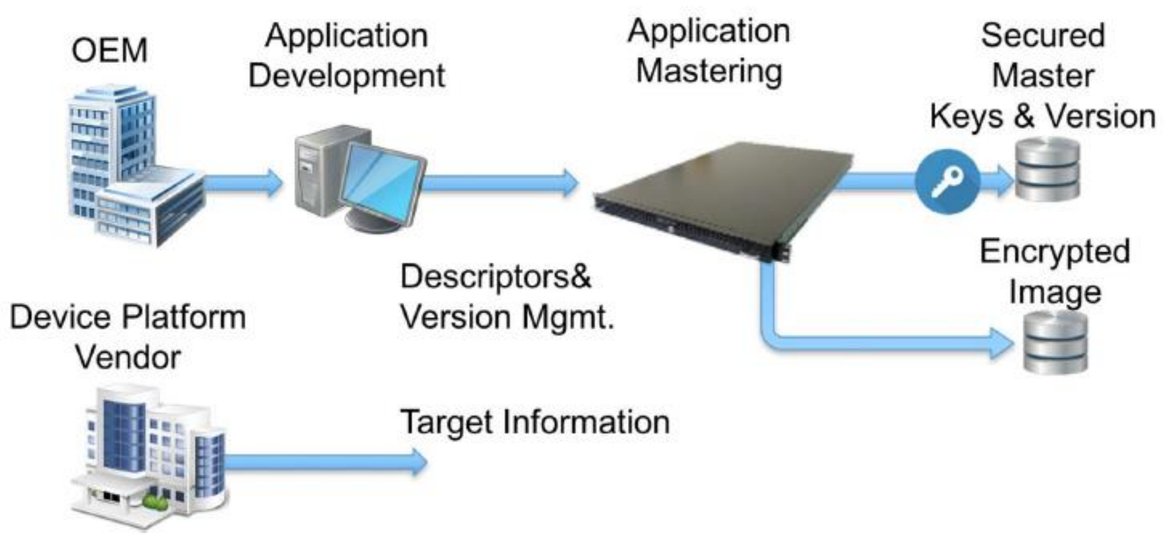
Secure products need to be architected and implemented with the correct OEM security framework which encompasses the root of trust, management of secrets and secure services offered, and the operating system and applications. Ideally, the OEM provisions the MCU with a Secure Boot Manager along with the OEM keys that are used to load all firmware and software into the MCU, immediately ensuring that the product has a unique identity and that all software loaded into the MCU has a known providence. Any firmware or software image that is to be loaded into the MCU must go through a "mastering" process where the image is signed and encrypted.

Inhibiting intellectual property theft, reducing over-production and cloning, and undermining counterfeiting are the key goals of the secure manufacturing process. To achieve secure manufacturing, it is therefore fundamental to encrypt the application software as early as possible and only decrypt the software in-situ in the device itself, securing as many touch points as possible. While simple in theory, there are multiple aspects that must be secured, including the device itself, how we master the application, how we handle and share the keys securely, and fundamentally how the application is loaded in to the device.

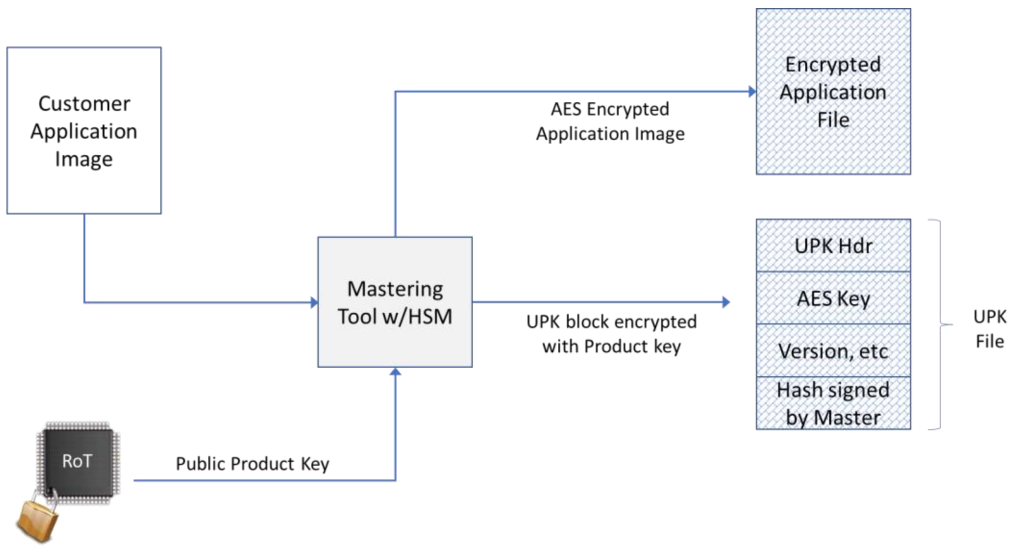
The product OEM develops the application software and may want to target it for a specific product or family of products and specifies some type of personalization or versioning for the software. To assist with the development of security, the software IDE can help establish security certificates and key hierarchies that are used to provision the SBM and MCU device. All OEM personnel developing the product software should be authorized. To enable secure manufacturing, only a limited number of personnel should have the ability to configure and master the software image that will released to production.

The mastering process itself accepts the application software image and encrypts the image with a secure Advanced Encryption Standard (AES) key, then creates a UPK containing the AES key along with version info that is signed with a unique cryptographic mastering key and encrypts the image with other product specific device keys. The mastering key and other keys must be handled securely to prevent third parties from having access to it, and it is optimal to incorporate a solution that integrates a robust hardware security module (HSM).

The mastering may utilize specific device identifiers, such as UUID or device type identifiers, if engendered by the silicon vendors. However, for initial implementations, it is presumed that these need to be provisioned at the secure programming center.



The output of the mastering process is a pair of encrypted files which are loaded into a secure manufacturing process.   These secure files can only be decrypted and verified by the Secure Boot Manager that was provisioned with the corresponding master key and product keys.


Customer Application Image
AES Encrypted Application Image
Encrypted Application File
Mastering Tool w/HSM
UPK block encrypted with Product key
UPK Hdr
AES Key
Version, etc
Hash signed by Master
UPK File
RoT
Public Product Key

Secure production completes the Chain of Trust

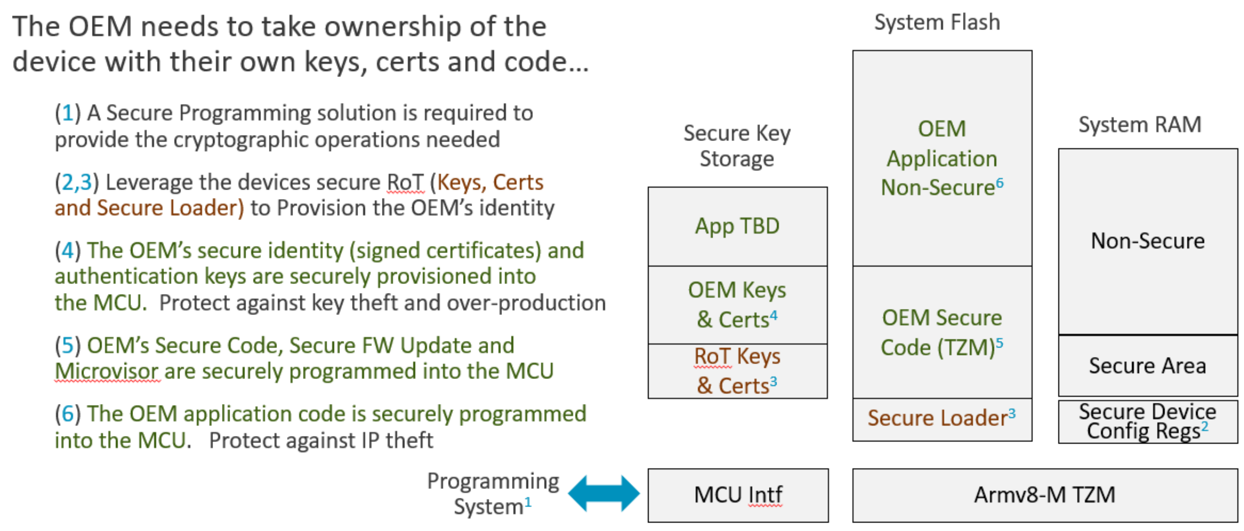
The steps to Secure Production include Secure Provisioning and Secure Programming/Manufacturing.

Secure provisioning

Given an OEM-centric provisioning model, the programming center needs to implement a model for securely accepting and handling the OEM's keys and certificates and provision these into the device that engenders the secure services on the device. To achieve this, a zero-trust secure channel must be implemented to enable secrets to be passed from the OEM through a non-secure channel, without delegating any knowledge of the secrets themselves outside of the secure provisioning process.

At a simple level, the secure provisioning of the device is exemplified as a secure IPSec connection between two secure enclaves, the OEM mastering and key appliance held by the OEM, and the

Secure Manufacturing appliance integrated within the programmer itself. Via this mechanism, it is possible to reduce the potential attack surface to a minimum, removing liability and risk from the programming center, and limiting the threat surface for the OEM.


The OEM needs to take ownership of the device with their own keys, certs and code…

(1) A Secure Programming solution is required to provide the cryptographic operations needed

(2,3) Leverage the devices secure RoT (Keys, Certs and Secure Loader) to Provision the OEM's identity

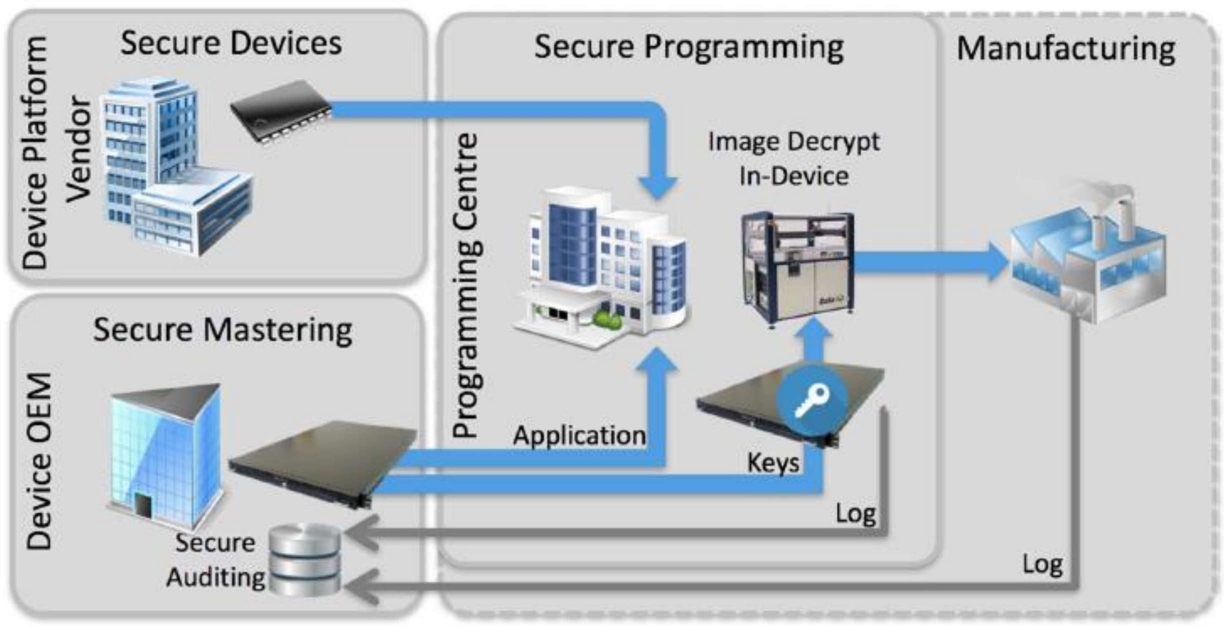
(4) The OEM's secure identity (signed certificates) and authentication keys are securely provisioned into the MCU. Protect against key theft and over-production

(5) OEM's Secure Code, Secure FW Update and Microvisor are securely programmed into the MCU

(6) The OEM application code is securely programmed into the MCU. Protect against IP theft

Secure programming/Secure manufacturing

Once the OEM's keys, certificates and SBM have been programmed to the device, the programming center also needs to accept the encrypted image from the OEM. Then the programming center needs to accept the production information for the system, including which devices may be targeted and how many devices are enabled. To enable the exchange and to maintain security, a secure channel is created between the OEM's mastering system, and the Secure Manufacturing appliance integrated into the programmer itself.



"Zero-trust" secure programming fundamentally requires that the image is protected from all stakeholders through encryption, and that no stakeholders can act upon the image through its deployment to the device. To enable this, the programmer must firstly identify that the device is valid; it must then download the application image to the memory of the device; the on-chip SBM must then confirm the download image and decrypt the image and program it to flash.

The splitting of the Secure Provisioning and the Secure Programming enables flexibility in how provisioning is accomplished, while enabling forward movement on secure programming. For example, the provisioning of the MCU devices may initially be carried out within the programming center, but may migrate to the silicon vendor foundry over time, increasing security further; however, the OEM programming can be maintained as is, ensuring continuity, minimizing costs and reducing any impacts on delicate manufacturing processes.

Summary

The threats to connected devices are only increasing, while more and more cloud-connected devices come online.

Product security is not something that can be just added at the end; it is something that the OEM must architect from the very beginning, starting with the software development process and continuing with the production of the product.

The only solution is to implement a "zero trust" design philosophy across the supply chain to minimise vulnerabilities and continually authenticate and individualize deliverables, from the start of development through to the deployment of millions of devices into our systems. Secure Development, Secure Mastering, and Secure Production are critical parts of ensuring a secure infrastructure and a chain of trust.
