Delivering high-mix, high-volume secure manufacturing in the distribution channel

Steve Pancoast (*Author*)
Secure Thingz, Inc.
IoT and Embedded Systems Security
San Jose, California, USA

Rajeev Gulati (*Author*)
Data I/O Corp.
Software, Semiconductor, and Systems Technology
Redmond, Washington, USA

Abstract**—** *This paper examines the cryptographic foundational elements required to establish roots-of-trust in silicon to design, manufacture and deliver secure devices. Recent advancements in security and programming technology, when designed in, streamline the manufacturing process, scale and deliver trusted devices to partners and OEMs cost-effectively. Other topics for discussion include impacts on manufacturing and downstream provisioning processes, as well as new technology in security provisioning and data programming that OEMs of any size can implement.*

Keywords— security, secure manufacturing, OEM, Internet of Things, IoT, root of trust, microcontroller, MCUembedded security; device provisioning; supply chain of trust; IP protection; IoT device; certificate signing; cryptographic key pairing, authentication, and decryption; hardware security module (HSM); public key infrastructure

I. INTRODUCTION

As billions of new IoT products come online every year, the opportunity to co-opt these products for nefarious purposes grows exponentially. In addition, the supply chain for these IoT products may be susceptible to threats such as cloning and intellectual property (IP) theft. The effects cannot be underestimated: Unauthorized attacks can significantly impact an OEM's revenue, profits, brand and reputation. Because of this, pressure is building for each and every IoT device to include security features that prevent the device from being used by an unauthorized agent. Zero security is no longer an option – it is a must-have.

Techniques used to combat the above threats include secure provisioning and programming of the products, along with operational security measures such as establishing trusted mutual authentication between the IoT device and a remote server, securing communication to and from the IoT device and securing the firmware running on the device itself. These capabilities can be enabled by features in secure semiconductor devices such as Secure Elements (SEs) and Secure Microcontrollers (Secure MCUs) as long as they are properly and securely provisioned.

This paper describes common security issues facing OEMs when developing and manufacturing IoT devices, including establishing a supply chain of trust, creating a root of trust and a solution for the secure provisioning and programming of Secure Elements and Secure MCUs. The paper also details the component architecture of the Data I/O SentriX™ system used for secure provisioning of Secure Elements and Secure MCUs, and it provides an example of how devices can be provisioned for mutual authentication during manufacturing.

II. CHAIN OF TRUST

A. Supply Chain of Trust

In creating secure products, the product developer (the OEM) should adopt a "zero / low trust" approach across the supply chain to minimize vulnerabilities and IP loss or theft. The OEM should continually authenticate and individualize deliverables across the supply chain as far as possible, and this involves establishing this chain of trust across the entire product lifecycle, including the end customer who will need a way to securely apply software updates for its product.

Figure 1 – Supply Chain of Trust

A typical supply chain of trust is shown in Figure 1. The chain of trust should start with silicon vendors (for the Secure Element or Secure MCU) and continue with programming solution providers and contract manufacturers all the way through to the OEM, who develops the end products, and even the end customer who needs to securely update the product in the field. Think of the chain of trust as a process flow - any step in the process builds upon the security of the previous step. Once the context of the overall supply chain is understood, focus can be placed on the Programming Center and the Provisioning System that resides there.

B. Root of Trust

The security of an IoT product starts by having a secure "root of trust" (RoT) that must be securely provisioned into the product which is usually contained in a SE or a Secure MCU itself. The root of trust typically consists of four key items, three of which are shown in Figure 2.

Figure 2 – Components of a Root of Trust

1. **A unique product asymmetric key pair** that is provisioned into the product and is secure and immutable. The private part of the key pair must be protected and provisioned/programmed into the SE or secure MCU so that it is never exposed, but can be used for authentication purposes (see #3).

2. **A unique identity that is secure and can be validated** (typically a product certificate). Every connected product should have a unique identity certificate. The most common implementation of this uses signed product certificates that can be verified by a certificate authority (CA). Unlike web browsers that connect to multiple sites, most IoT devices connect back to just the OEM's own site, so the CA can be the OEM itself (i.e. a self-signed CA) or a third-party CA can also be used. The key principle is that there needs to be a verifiable certificate chain from each product back to a trusted CA.

3. **A secure way to authenticate the identity of the product** (i.e. tie the product to the certificate). The SE or MCU provides a cryptographic method to authenticate that the public key (from the product certificate that was previously validated) matches the corresponding private key in the SE or MCU.

4. **A secure and immutable boot path (for MCU solutions).** In addition to the items above, a secure MCU must also provide a secure boot mechanism where the integrity of the initial boot software is cryptographically verified before executing it. This process continues successively where the boot software verifies the integrity (signature) of the subsequent software before is executed, etc.

The RoT represents the base level of security information that must be protected in the secure device against readout and tampering, etc. This is usually done with a variety of hardware and software protection methods in the device. Beyond the RoT, many layers of operational security software are required for different classes of IoT products, but each of these solutions typically rely on some initial security starting point that is empirically trusted. So it is critical that the RoT be secure and properly protected at manufacturing.

C. Secure Provisioning and Programming

To implement the chain of trust process, the OEM should define roles for its own ecosystem partners and suppliers in the product's lifecycle, including partners in the development and manufacturing of the product. The OEM should take ultimate ownership for the security of its own products and protect its own Intellectual Property. One of the key areas often overlooked by OEMs is the secure provisioning and programming of the RoT and product software either at a Programming Center or contract manufacturer. In addition, it is important for the OEM to address how software updates can be securely deployed for its own products.

The challenges of a secure manufacturing solution should not be understated. Secure devices (SEs and MCUs) must be produced securely anywhere in the world with an OEM's keys (RoT) and product software protected. OEMs must restrict access to secrets, but most OEMS need to trust third parties for high volume production. With an automated security provisioning and data programming solution, Programming Centers and contract manufacturers are able to handle more customers with less audit overhead because the provisioning and programming are managed cryptographically. The OEM's secrets are protected inside a hardware security module (HSM), and over-production is eliminated.

For Secure MCUs, the security problem is more complicated. Beside the need to securely provision the RoT into the MCU, there is a need to securely program the OEM's application software / firmware into the MCU to protect against IP theft. The OEM should also provide a solution to securely update the software in its products after production. Secure Thingz has created a solution where the software can be securely programmed / updated as it is "mastered" with a secure system at the OEM and sent to the programming facility. Mastered software images are encrypted and protected against any modifications and can only be decrypted and installed by the targeted device or family of devices.

III. PROVISIONING SYSTEM ARCHITECTURE

A provisioning system component architecture is a turnkey solution that enables an OEM to securely provision component devices like Secure Elements and Secure MCUs. A common usage model for the provisioning system, shown in Figure 3, involves its setup at a Secure Programming Center. Secure Programming Centers provide important provisioning services to OEMs at various volume levels ranging from first article to hundreds of thousands of units. Automated security

provisioning and data programming systems may also be similarly setup at OEM-controlled factories or factories owned by contract manufacturers.

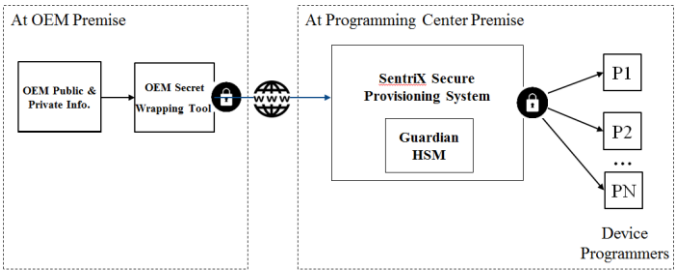

Figure 3 – Secure Provisioning System Architecture

While the secure provisioning of devices may be outsourced by an OEM to a Secure Programming Center, OEMs need to provide both public and secret information that is necessary to provision their devices. As an example, an OEM may need to provide a Secure Programming Center with private signing keys, certificates, certificate templates, production counts and other important information. For the purpose of this paper, and as shown in Figure 3, such material will be referred to as "OEM Public and Private Information".

OEMs create and manage the OEM Public and Private information on their premise in a highly secure environment. Such information is company secret and of very high value to an OEM, yet this information needs to be transmitted to a Secure Programming Center so that devices can be provisioned. The OEM Secret Wrapping Tool is a subsystem that enables OEM Public and Private Information to be cryptographically signed and encrypted, or "wrapped". This allows the information to be protected and be securely transmitted to a **specific** provisioning system at a Programming Center over an unsecure Internet connection. The unwrapping of the OEM's secret information only occurs inside a secure, tamper-resistant HSM where it is stored and protected. This process of wrapping takes place at an OEM Premise by the OEM and the wrapped file is then sent to a Secure Programming Center.

The Secure Programming Center is responsible for provisioning devices for the OEM customer at its premise. This process starts with creation of a Security Product using the specific system programming software. During product creation, the wrapped OEM Public and Private Information is imported into the programming system and cryptographically bound to a Unique Product ID and a Unique OEM ID within the HSM. Thus a unique product representation (OEM ID and Product ID) is created.

IV. Mutual Authentication use case

Now that the component architecture of the system has been reviewed, consider a real world use case of mutual authentication to see how the provisioning system is used.

Mutual authentication requires both parties to have a provisioned root of trust as outlined in Section II. Two devices attempting to communicate with each other will use this root of trust to cryptographically authenticate before exchanging data. A brief study of how mutual authentication works will establish the device provisioning requirements. One of the devices could be an update server and the other device an IoT product, but many variants are common.

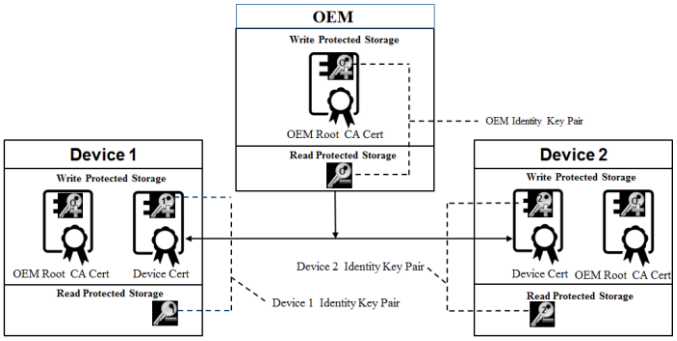

Figure 4 – Mutual Authentication Between Two Devices

The application and devices are developed by OEM A, who is thus the owner of Device 1, Device 2 and the application. Thus OEM A is also responsible for provisioning both devices for mutual authentication.

This process generally requires creating a Public Key Infrastructure (PKI) system among the various devices in the application. Identities for Device 1 and Device 2 are created by associating an Identity Key Pair with each device. As shown in Figure 4, the private part of the Identity Key Pair for each device is stored in read and write protected storage on the device (and thus never exposed to the outside world). The public part of the key pair is stored in a Device Certificate in write-protected storage and represents the Public Identity of the device (and is available to share with the outside world).

Assigning ownership of Device 1 and Device 2 to OEM A requires that an Identity Key Pair for OEM A be created as shown in Figure 4. To assign ownership of Device 1 and Device 2 to OEM A, the Device Certificate for each device is signed with the Private Key from the OEM Identity Key Pair. The Public Identity of the OEM is represented by the OEM Root CA certificate, which contains the Public Key of the OEM. In this example, we are assuming that the OEM is also the Root Certificate Authority (where the certificate signature chain of trust needs to terminate), thus the OEM Root CA certificate is signed by the Private Key of the OEM. Note that in most cases, the Root CA is used to sign intermediate CA certificates and these are, in turn, used to sign device certificates. However, for this example, the Root CA will be used directly for simplicity.

The OEM Root CA certificate is also associated with Device 1 and Device 2 so that during the device authentication process, the chain of trust starting with the Device Certificate for Device 1 (or Device 2) can be dynamically verified to terminate at same Root CA certificate as the OEM Root CA certificate stored on the device.

Once the above PKI system is set up, the mutual authentication algorithm works as follows:

1. Device 1 requests Device 2 to send its Certificate.
2. Device 2 sends its Device Certificate to Device 1.
3. Device 1 authenticates Device 2 by validating Device 2's Device Certificate. This is done in two steps.
    a. Cryptographic validation that Device 2 certificate is from OEM A – this involves ensuring that the Device Certificate is signed by Private Key of the OEM Identity Key Pair and by verifying the signature chain of trust starting with the Device Certificate of Device 2 and terminating at the OEM Root CA Certificate. Each device has a local copy of the OEM certificate that it trusts.
    b. Authentication of the identity of Device 2 using the Device 2 Public Key from the Device 2 certificate and performing a challenge response algorithm with Device 2.

    Since Device 2 is the only entity that knows the Device 2 private key, it is the only device that can successfully respond to the challenge, thus proving that the Public Key belongs to Device 2. The details of the challenge response mechanism are not covered in this paper.
4. If validation of the Device Certificate by Device 1 is successful, Device 2 sends affirmative authentication response to Device 2.
5. Device 2 executes a complementary sequence to authenticate Device 1.

V. PROVISIONING IMPLEMENTATION

Now that mutual authentication requirements have been discussed, provisioning Secure Element devices using the specified provisioning system is outlined. From the discussion related to the provisioning architecture and understanding of the mutual authentication use case, the following requirements will need to be supported for provisioning devices:

1. The OEM creates the following at the OEM Premise:
    a. An OEM Identity Key Pair
    b. An OEM Root CA Certificate
    c. A Device Certificate Template
    d. Production Count for number of devices to be produced at the Secure Programming Center
2. The OEM securely transmits the following OEM information to the Secure Programming Center:
    a. OEM Device Certificate Signature Key (which is the Private Key from the OEM Identity Key pair)
    b. OEM Root CA Certificate
    c. A Device Certificate Template
    d. A Production Count
    e. Unique serial numbers to be programmed into devices (optional and not shown)

The above information flow is shown in Figure 5.

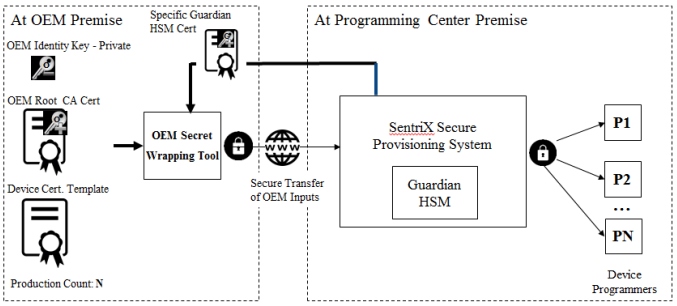

Figure 5 – Secure Transfer of OEM Information to Provisioning System

The OEM Identity Key is secret, must be protected and is wrapped before transfer to the specific provisioning system. The OEM Secret Wrapping Tool targets a specific HSM system at the Secure Programming Center, thus a Specific Guardian HSM identity certificate from the target programming is required. OEM Public Information is not secret and technically does not need to be wrapped; however, this is often a convenient method of transfer.

Once securely stored inside the provisioning system, the OEM Information is used to create a Job Package, which initiates the provisioning cycle for a batch of Devices. The simplified provisioning flow for a single device is as follows:

1. Generate a Device Identity Key Pair for each device.
2. Create a Device certificate using the Public Key of the Identity Key Pair.
3. Sign the Device certificate with OEM Device Certificate Signature Key.
4. Program the Device Certificate into the Device Write Protected storage.
5. Program the Root CA Cert into the Device Write Protected storage.
6. Lock the Device.

Once this provisioning flow has been executed, devices are protected from modification and prepared for mutual authentication in the field.

VI. SUMMARY

This paper has described common security issues facing IoT OEMs. In order to secure IoT devices, OEMs must establish a supply chain of trust, create a root of trust in each device and ensure the secure provisioning of secure elements and secure MCUs for those devices. A component architecture of a secure provisioning system was discussed and an example of mutual authentication was used to demonstrate the necessary device provisioning steps. This fundamental provisioning architecture can also be used for more advanced usage models, including secure boot of an MCU and the authentication and encryption of firmware. These advanced usage models can be implemented with a Secure Element or a Secure MCU.

VII. TRADEMARKS

VIII. REFERENCES

[1] Microsoft, "Securing Public Key Infrastructure (PKI)," May 2014. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn786443(v=ws.11)

[2] Jones, Scott, "Secure Authenticators answer the call to solve IoT device embedded security needs", Embedded World 2017, unpublished.