Keywords: secure boot, embedded security, asymmetric cryptography, secure microcontroller, embedded system security

APPLICATION NOTE 5696

Protect Your Designs from Malware with the DeepCover MAXQ1050 Secure Microcontroller

By: Christophe Tremlet, Security Segment Manager
Aug 06, 2013

Abstract: Malware injection has become a critical threat to embedded systems. Implementing an asymmetric cryptography-based secure boot is the best protection against this class of attacks. This application note describes the key principles of such a secure boot and explains how to implement it with the DeepCover® MAXQ1050 secure microcontroller.

Embedded system security is a growing concern. There are new attacks on embedded systems every day, including on systems involved with health or safety. One type of attack is malware injection, the insertion of malicious code into a webpage. Once an attacker has succeeded in making a device run a fraudulent piece of software, this unauthorized software can:

- **Send confidential data externally.** If used in the medical industry, malware injection could cause devices (such as a portable ECG) to inadvertently transmit personal health information. In perhaps a more wide-reaching effort, malicious software could make an encryption key to accessible to the public.
- **Force the device to operate incorrectly.** A famous example of this is the Stuxnet virus, which after infecting programmable logic controllers (PLCs), forced centrifuges to run at different speeds than expected.
- **Induce an unpredictable device behavior.** This includes behaviors that could threaten human life.

A properly secured boot process allows only authorized software to run on a given device. It thus prevents malware injection, even during update phases. To bring a high level of trust, a secure boot must rely on proven cryptographic algorithms. This, however, creates several challenges:

- The most appropriate algorithms are asymmetric ones, which require intensive computing power.
- The keys associated with these algorithms must be protected.
- The implementation must be flawless.

In many systems, these requirements can be challenging to implement. However, adding a secure microcontroller as a coprocessor like the MAXQ1050 can efficiently support a secure boot implementation while guaranteeing a very high level of security.

Authentication and Digital Signature of the Firmware

To ensure that the target embedded device runs only authorized software, we need to provide a way to authenticate said firmware. This means making sure software is genuine and was written or approved by an authorized entity. Adding a digital signature to a firmware (similar to putting a seal or a manual signature at the bottom of a letter) enables this.

The software loaded during the manufacturing phase should be digitally signed, and should also apply to each firmware update. This way, the digital signature enables trust during the device's entire lifetime.

A strong digital signature must be computed by a cryptographic algorithm. To bring the highest level of security, the algorithms need to be public and well proven. Here, we will consider asymmetric cryptographic algorithms, namely the Elliptic Curve Digital Signature Algorithm (ECDSA) and RSA associated to SHA.

Asymmetric Cryptography Applied to Secure Boot

The key principle of a secure boot based on an asymmetric cryptography is that the software developer holds the private key used for signing, while the embedded device stores the public key for verification. The main advantage of this (compared to symmetric crypto based secure boot) is that the confidential element (i.e., the private key) is never stored in the end product. Hence, when using

ECDSA or RSA, there is no way an attacker can retrieve the private key, even through the most sophisticated invasive attack. The secret is just not stored anywhere in the field device! All the attacker can get from the device is the public key, and with asymmetric crypto, it is practically impossible to retrieve the private key when only knowing the public key. This is a fundamental benefit of asymmetric cryptography.

Figure 1 presents the flow of a secure boot based on asymmetric cryptography. The firmware authenticity is verified through ECDSA or RSA, which provide an extremely high level of trust, providing that the key length is adequate (typically 2048 bits for RSA or 224 bits for ECDSA). ECDSA and RSA are efficiently and securely sustained by SHA-256. It would be impractical to digitally sign a full piece of firmware, so instead we compute a digest ("hash value") that is guaranteed to be unique and unforgeable by the fundamental properties of the SHA-256 hash algorithm. This digest is then signed through ECDSA or RSA. The same flows are applicable to firmware updates, except that firmware loading will not apply in the manufacturing facility. One will also notice that the private key never needs to leave the software design center.

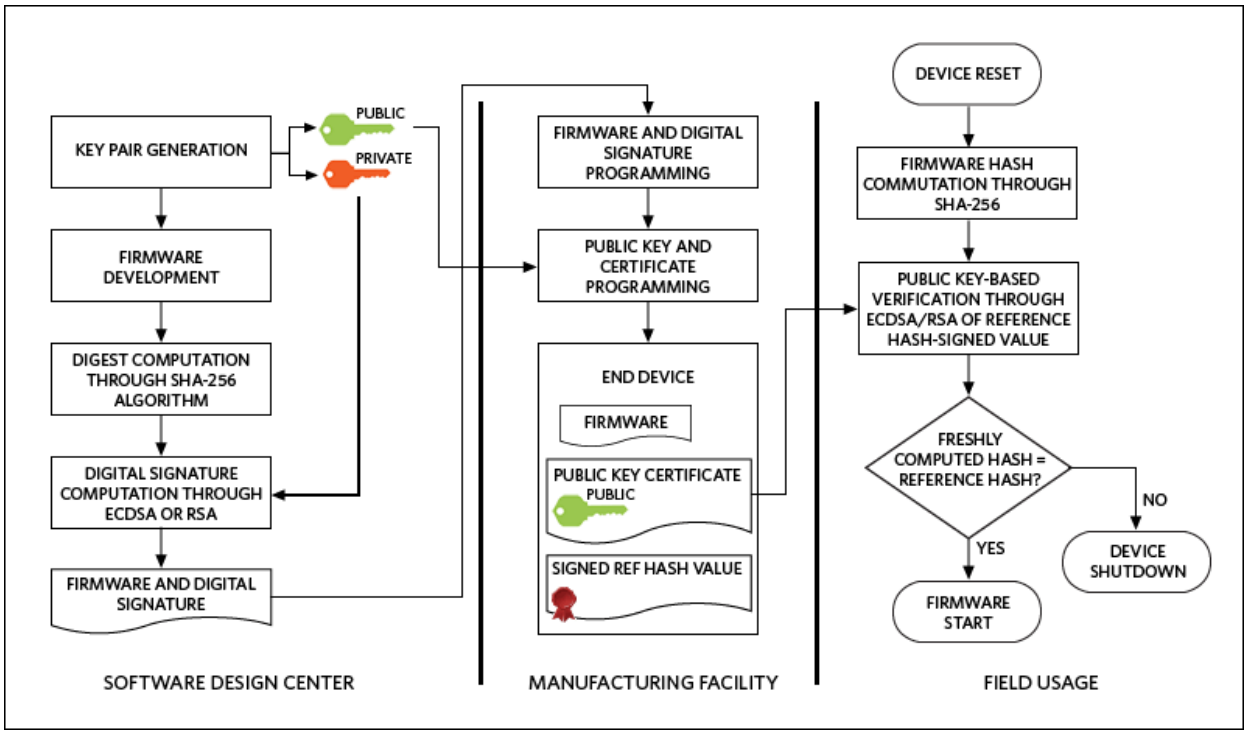

Figure 1. A secure boot flow across device life-cycle phases.

While asymmetric cryptographic offers key benefits, it does require intensive computing resources. Computing a SHA-x on a large piece of software is time-consuming when done through software. The RSA or ECDSA signature also requires resources, especially if the main system microcontroller does not have a multiplier.

Another challenge is the integrity of the public key and its certificate. A public key, as one can imagine, does not need to be confidential. The public key can be disclosed to anyone because it only allows verification. However, an attacker can try to substitute the original public key with a personal public key. If this substitution is successful, the device would authenticate the piece of software signed by the attacker's private key. To avoid this scenario the public key, *integrity* must be protected. In other words, we must guarantee it is not modified nor replaced.

Why Use the MAXQ1050 for Secure Boot Implementation?

The DeepCover MAXQ1050 secure microcontroller fulfills these requirements. **Figure 2** presents a typical block diagram for the implementation of a MAXQ1050-enabled secure boot for an embedded device.

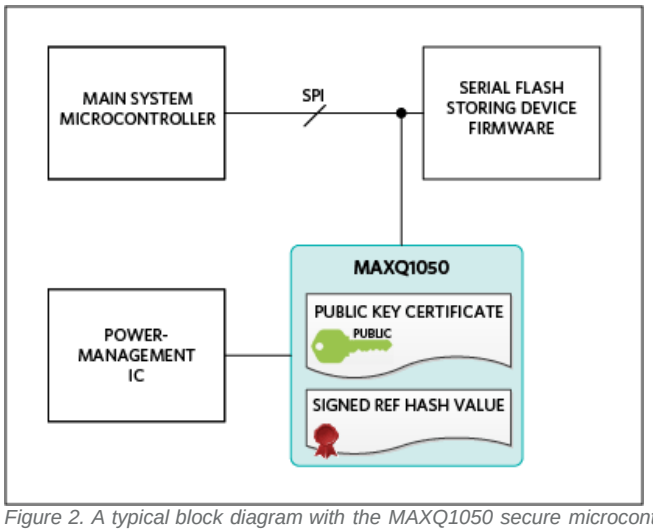

Figure 2. A typical block diagram with the MAXQ1050 secure microcontroller used as a secure coprocessor.

Thanks to its embedded secure hash engine, the MAXQ1050 accelerates the computation of the firmware hash, which improves the device boot time. The MAXQ1050 can also perform a fast ECDSA or RSA signature verification, due to its MAA hardware accelerator for modular arithmetic operations.

The MAXQ1050 executes the steps shown in Figure 1's "Field Usage" section. On top of the hash computation of the firmware and digital signature verification, the MAXQ1050 also informs the main system microcontroller and/or the power management IC (PMIC) of the status of the firmware authentication. Implementation examples include the following:

- The MAXQ1050 (through a GPIO) can allow the PMIC to power the microcontroller.

  OR

- A GPIO of the MAXQ1050 can be connected to the reset pin of the main microcontroller. The reset is only released once the firmware authenticity has been verified.

  OR

- The two options above followed by a digital sequence sent by the MAXQ1050 on several GPIOs.

Integrity of the public key, associated certificate, and reference hash value are guaranteed, thanks to the secure storage capabilities of the product.

Targeting the Right Level of Security

The expected level of security for a product is often difficult to define—reaching the highest possible level generally results in high development and manufacturing costs. Hence a trade-off has to be found. This trade-off is often based on the potential for attacks and the damage resulting from a successful attack. To determine the appropriate level of security, many risk factors need to be considered, including:

- The attacker's tools: the attacker can have a much stronger impact if he has access to sophisticated equipment, such as a scanning electronic microscope (SEM) or a focus ion beam (FIB).
- The attacker profile: an attacker can either be someone acting alone or a member of a mafia/criminal organization with strong financial support.
- The attacker's motivation: an attacker may expect either financial benefit or only recognition for his skills.
- The financial impact of a successful software attack.

In this application note, we will consider three levels of attack potential:

- **Basic:** the attacker is able to attack the system by all software means, including malware injection. However, he/she is unable or not tooled to modify any physical characteristic of the system.
- **Moderate:** in addition to software attacks, the attacker can perform physical attacks, such as probing a printed circuit board (PCB)

track to read a signal, forcing the level of a digital pin, or removing an IC from the PCB.

- **High:** the attacker is able to perform highly invasive attacks, such as microprobing the signals on the bonding wires of an IC or microprobing the IC itself.

For each level of attack potential, Maxim Integrated can implement the adequate level of protection:

Basic potential: the implementation proposed in Figure 2 provides an adequate level of protection without further specific recommendation.

Moderate potential: the basic implementation provides protection against some attacks (for example, the attack where an agent replaces the serial flash containing the system with a fake that contains the attacker's software). This is because our secure boot sequence detects any fake software. To increase the resistance of the system to hardware attacks without increasing the development or manufacturing costs, we recommend taking some layout precautions. Here are some examples:

- Route the tracks of the signal connecting the MAXQ1050 to the PMIC or main system microcontroller in the inner layers of the PCB to prevent easy access to the signal. This way, an attacker cannot easily connect them to a probe and force them.
- Use dynamic signals (i.e., pulses or sequences of pulses) to notify the main microcontroller that the boot is successful. This prevents an attacker from connecting the signal pins to a fixed level such as ground or V_{DD}.
- Use multiple pins bearing different dynamic signals to inform the main controller that the boot is successful. Route the tracks in such a way that it would be difficult for the attacker to control both pins at the same time.

High potential: bringing protection against the most sophisticated attacks would consist in an implementation that is compliant with FIPS 140-2 level 3 or 4. This implementation should detect any physical tamper attempt and react immediately by destroying any sensitive information, rendering the system inoperable. Because making the device operable again would require going through a maintenance phase, this level of protection should be implemented only when security overtakes system availability.

Thanks to its self-destruct inputs and instantly erasable NV SRAM, the MAXQ1050 secure microcontroller can support these high-level security requirements. Since implementing this level of security is beyond the scope of this application note, please contact us for support.

Related Parts	
MAXQ1050	DeepCover Secure Microcontroller with USB and Hardware Cryptography

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5696: http://www.maximintegrated.com/an5696
APPLICATION NOTE 5696, AN5696, AN 5696, APP5696, Appnote5696, Appnote 5696