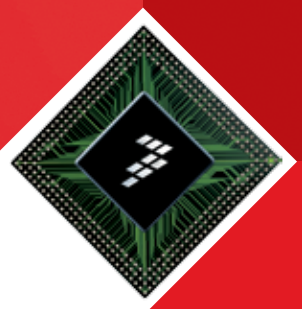
White Paper

Secure Boot

For QorIQ Communications Processors



freescale.com

freescale™

Introduction

Security continues to be an increasingly important concern in the design of modern systems. Threats against networks and network-connected devices are real and growing. With an estimated $40 billion (USD)* of data loss per year, service providers and end users are becoming painfully aware of the consequences of unsecured networks and databases.

As the market leader in embedded communications processors, Freescale understands the foundational role our processors can play in securing the traffic passing through networks as well as the potential for processors to harden network nodes and connected devices against attack, effectively making them trusted systems. A trusted system is a system that does what its builder (OEM) and users expect it to do, and specifically does not do other things which the OEM and/or users would consider attacks.

The starting point for a trusted system is assurance that it boots and executes only authentic code. Consequently, secure boot is a cornerstone of the QorIQ platform's trust architecture, which also includes secure runtime, secure debug, tamper detection and device-specific secret key usage. The QorIQ P1010 processor, along with the P2040, P2041, P3041, P4080, P4040, P5020 and P5010, implements the trust architecture, providing system developers with the hardware anchor points needed to develop a trusted system. The secure boot information contained in this white paper, though specifically referencing the P1010 processor, applies to the other products listed here as well.

Objectives of Secure Boot

Secure boot is a process through which the P1010 determines whether the system's image is trusted. The P1010 doesn't know the intent of the code, and has no objective way of knowing whether the code will attempt to do malicious things. Therefore, in the context of secure boot, trusted = authentic. System developers digitally sign their code to allow the P1010 to distinguish authentic trusted code from untrusted non-authentic code.

The ability to distinguish between trusted and untrusted code enables the following capabilities:

- Prevent the CPU from running untrusted code rather than authentic OEM signed code
- Detect and reject modified security configuration values and device secrets
- Allow trusted code to use a device-specific, one-time programmable master key (OTPMK) when the trust architecture says the P1010 is in a secure state
- Prevent extraction of sensitive values from the device by any means, short of de-processing

Note that while some developers have a critical need to know their systems are hardened against attack, this isn't a universal requirement. Consequently, the secure boot and trust architecture are disabled by default. Developers not implementing trust features can ignore their existence.

Also note that developers who choose to leverage the trust architecture are not dependent on Freescale to provision devices or sign code. P1010 devices are sold ready for provisioning with very little impact to board design or the developer's manufacturing flow, and code signing is performed using tools provided as part of the P1010 software development kit (SDK).

Code Signing

The starting point for a trusted platform is the creation (by the developer) of a bug-free and malware-free code base. Once the developer "trusts" the code, the developer digitally signs the code so that accidental or deliberate modifications to the code base will be detected during the secure boot cycle.

As shown in the figure at right, the OEM calculates a hash over the system code (executable instructions and configuration information). It is possible (even advantageous) for portions of the code to be encrypted—this prevents attackers from stealing the code from flash.

The OEM generates an RSA public and private key pair. It is the responsibility of the OEM to tightly control access to the RSA private signature key.

If this key is ever exposed, attackers will be able to generate alternate images that will pass secure boot. If this key is ever lost, the OEM will be unable to update the image.

The hash is signed using an RSA private signature key. This encrypted hash is known as a digital signature, and the digital signature is appended to the code, with both being written to flash (or other system non-volatile memory). Another hash is calculated over the public key which the P1010 will use to validate the system's code. This hash value (the super root key hash) is programmed into a fuse block within the P1010.

Secure Boot Sequence

At a high level, secure boot entails the P1010 using the RSA public key (super root key) to decrypt the signed hash while simultaneously recalculating the SHA-256 hash over the system code. The P1010 compares the decrypted original hash to the freshly calculated hash and, if the values match, the code is considered authentic. In reality, however, this process involves a few more steps.

Pre-Boot Phase

When the device is first powered on, reset control logic blocks all device activity (including scan and debug activity) until fuse values can be accurately sensed. The most important fuse value at this stage of operation is the intent to secure (ITS) bit. When an OEM sets ITS in the security fuse processor, they intend for the system to operate in a secure and trusted manner or not at all. The setting of ITS determines the default settings of a range of configuration registers within the device, essentially locking down interfaces, memory permissions and MMU configurations until trusted software is executing.

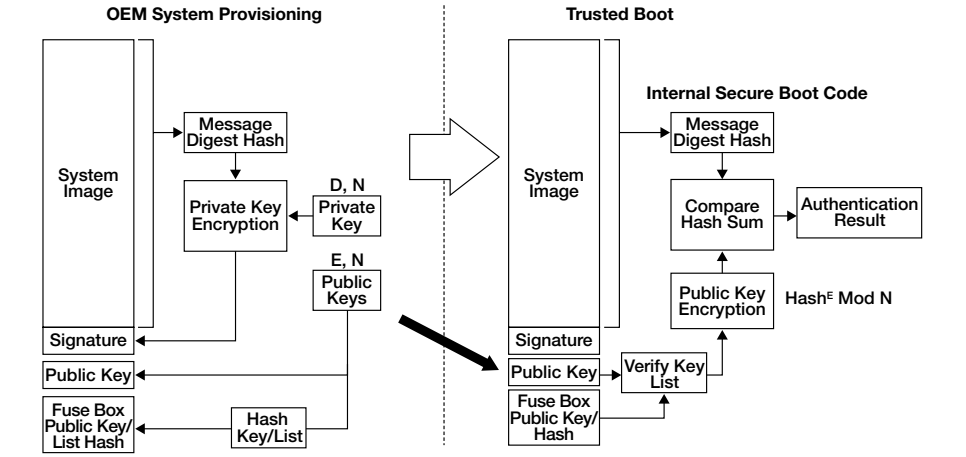
When the ITS bit is set, the system jumps to an internal boot ROM (IBR) for booting. The contents of this ROM include the internal secure boot code (ISBC), which checks the cfg_rom_loc value to determine the location of the code the developer wants to run, assuming secure boot passes.

ISBC Boot Phase

The internal secure boot code (ISBC) is Freescale-developed code that performs device security health checks and verification of the digital signature over the developer's code in external non volatile (NV) RAM. The developer's code could be a monolithic image including bootloader, OS and applications, or secure boot could involve a chain of validations, where the ISBC validates the bootloader, the bootloader validates the OS and the OS validates the applications. The Freescale SDKs for trust architecture devices include an example of a chain of validations, starting with the ISBC validating a modified version of u-boot referred to as trusted u-boot. In some documentation, trusted u-boot is also referred to as the external secure boot code (ESBC).

The ISBC's main function is to verify the authenticity of the ESBC, which performs more extensive device configuration and code authentication similar to that performed by the ISBC. The ISBC accesses main memory to obtain the ESBC code from a predefined address. The ISBC relies on a header file called the command sequence file (CSF) header to determine that it has found a potentially valid image, and to know the size and other characteristics of the image to be validated. This CSF header is added to the developer's image by the Freescale code signing tool. If the ISBC successfully validates the image (ESBC in this example), it puts the P1010 into a secure state and jumps to an entry point within the ESBC. If the digital signature validation fails, the P1010 goes into a FAIL state and refuses to boot.

Code Integrity Via the Trusted Boot Process



ESBC Phase

Unlike the ISBC, which is in an internal ROM and therefore unchangeable, the ESBC (trusted u-boot) is Freescale-supplied reference code, and can be changed by OEMs. Consequently, the description here is based on the Freescale reference implementation—other actions are possible.

Trusted u-boot performs typical u-boot configuration functions, such as mapping physical memory, initializing the network interfaces and data path infrastructure, and loading next-stage software such as the OS (trusted u-boot client) into main memory. The trusted u-boot client has the same format CSF header prepended to it as trusted u-boot have, which allows trusted u-boot to perform signature validation over the trusted u-boot client. The public key used for this validation can be the same as used by the ISBC, or it can be a new public key from the trusted u-boot client's CSF header. If the signature passes, trusted u-boot jumps to the entry point within the client and begins execution. At this point, the developer's authentic device configurations, OS and applications can be considered to be running.

If trusted u-boot fails to validate the trusted u-boot client, the P1010 goes into a FAIL state and the CPU spins until the device is reset.

Conclusion

With the explosive growth in network connected devices, the need for security assurance from embedded systems has grown considerably. Networking, access and industrial embedded systems are multi-billion dollar industries, and the economic impact of the unavailability of these embedded systems (or, of being practically unavailable due to lack of trust in their operation) is many times larger. Open source code is increasingly treated like modules to be downloaded and plugged into holes in an OEM's own software offerings, despite the unknown origins and without thorough analysis for backdoors.

The Freescale QorIQ processor portfolio has deliberately designed trusted subsystems into its devices that allow users to meet assured computing goals without comprising the performance requirements of the mission. The QorIQ platform's trust architecture, available in the P1010, P2040, P2041, P3041, P4040, P4080, P5010 and P5020, provides OEMs with the hardware anchor points they need to develop a trusted system. Additional hardware support during boot time assures that boot and runtime code is trusted before execution and prevents unauthorized debug access to secure state. The trust architecture and secure boot features within the QorIQ processors provide the OEM the tools necessary to achieve assured computing within the system requirements of size, weight and power.

How to Reach Us:

Home Page:
freescale.com

QorIQ Portfolio Information:
freescale.com/QorIQ

e-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For more information, visit
freescale.com/QorIQ and freescale.com/security

Document Number: QORIQSECBOOTWP REV 1