Module Introduction

Purpose
• The intent of this module is to provide you with an overview of the security features of the i.MX31.

Objectives
• Describe the High Assurance Boot.
• Describe the Security Controller.
• Describe the Secure JTAG Controller.
• Describe the IC Identification Module and e-fuses.
• Describe the Run-Time Integrity Checker.
• Describe the Tamper Detection System.

Content
• 12 pages
• 2 questions

Learning Time
• 20 minutes

The intent of this module is to provide you with an overview of the security features of the i.MX31, including High Assurance Boot, the Security Controller, the Secure JTAG Controller, the IC Identification Module, the Run-Time Integrity Checker, and the Tamper Detection System. It should be noted that, unless specifically mentioned, all information in this module applies to both the i.MX31 and the i.MX31L.
The i.MX31 provides a secure platform and protects information and data from unauthorized access in the form of inspection (read), modification (write) or execution (use). The i.MX31 security architecture is a combined software and hardware solution.

The i.MX31 implements the following security features: a Memory Management Unit; High Assurance Boot, or HAB; a Security Controller, or SCC (including secure RAM and security monitor); a Run-Time Integrity Checker, or RTIC (including an SHA-1 accelerator); a true random number generator accelerator; a Secure JTAG Controller (with optional JTAG disabling); Universal Unique Identification; and finally, Tamper Detection.

The i.MX31 security features are designed with many concerns in mind. I.MX31 security protects against network infiltration, as well as services and device theft. In the United Kingdom alone, it is estimated that one million phones are stolen yearly. Should a device be stolen, the security features of the i.MX31 are designed to protect personal data. The i.MX31 is designed to handle the following security concerns: cloning, configuration protection, and theft of services; operating system and vendor software authentication; secure mobile communications and transactions; secure video conference call; M-commerce; digital rights management; and defense against physical tampering.
Here is a high-level block diagram of the i.MX31 security architecture. Let’s look at a few of its features.

The iROM secure boot is a secure boot mechanism. It ensures that the image code is a trusted and authenticated operating system.

The true RTIC is a device that verifies the authenticity of software and data in real-time.

The Memory Management Unit (MMU) is responsible for process separation. It also generates a “Supervisor mode” signal that indicates whether the process accessing a specific peripheral is a Supervisor mode process or a User mode process. The security hardware in the i.MX31 can only be accessed by Supervisor mode processes.

The i.MX31 contains a secure RAM, which provides secure storage facilities.

Above the secure RAM is the security monitor, the brain of the system. It is connected to all of the security alarms of system. The security monitor informs the secure RAM how to act in cases of security violations.

The security monitor is connected to the RTIC alarm. The software alarm indicates if, for example, the anti-virus application has detected a virus. The security monitor is also connected to the secure JTAG controller, which protects the system from JTAG manipulation. Finally, the security monitor is connected to the tamper detection external circuitry, which indicates when there has been an attempt to remove the device’s cover.
The High Assurance Boot, or HAB, is a software process that ensures trusted execution of the operating system and application code. After the operating system is powered on or reset, the HAB gains control and performs a health check of all security hardware.

Once it has performed the check, the HAB validates that the code image, stored in external memory, originated from a trusted authority (its authenticity). It also verifies that the code stored in external memory is in its original form and has not been modified (its integrity). The boot process is flexible because it is controlled by authenticated scripts that reside in external memory.

A hash of the original code block is calculated and encrypted with the private key creating a signature. The signature is then bundled with the code and prepared for storage in flash. This procedure is used for each protected block of code or data.

The HAB uses digital signatures to verify the authenticity of the code and data resident in off-chip memory. The signatures are created using hashing and public key cryptography. Digital signatures are simply Secure Hash Algorithm-1 (SHA-1) hash values that are encrypted with a RSA private key crypto system. Verification by the HAB uses the public key, also known as the Super Root Key (SRK), which is stored in on-chip non-volatile memory. To enhance the robustness of the HAB security, multiple SRKs are stored in internal ROM, and the applicable one for a given manufacturer is selected using electrically blowable fuses. Once the fuses are blown, they are protected from further modification. i.MX31 supports both 1024 bit SRKs and 2048 bit SRKs for enhanced security products.

The signature, extracted from the bundled code block and the SRK, is used to decrypt it into the original hash value. A new hash of the code block is re-calculated then compared with the decrypted hash value.

There are two possible results of the code comparison. If the two hash values are equal, the code is verified to be correct, unaltered and authentic. If the two hash values do not match, it is certain that the code has been modified, either intentionally or unintentionally, and the operating system will not run.

HAB has a mechanism for version control that ensures that old revoked code image releases will not be used. If a security flaw is discovered in the code and an update is issued, it is not possible to use the old version to take advantage of the security flaw. The HAB can also be set to control the version of the loaded operating system. This is called “code revocation.”
Question

What security checks does the HAB perform at power on/reset?
Select all that apply, and then click Done.

- It scans the in-coming code for viruses.
- It validates that the code image is authentic, that is, that it originated from a trusted authority.
- It checks the private and public keys to ensure they have not been damaged since the previous power on/reset.
- It verifies that the code stored in external memory is in its original form and has not been modified.

This question tests your understanding of the security function of the HAB.

Correct.

The HAB validates the authenticity of the code image (that it is from a trusted authority) and the integrity of the code (that it has not been modified). The HAB can also be set to control the version of the loaded operating system.
The Security Controller, or SCC, is composed of two hardware blocks: the Secure RAM and the Security Monitor. The Secure RAM provides a way of securely storing sensitive information both in on-chip RAM, and in off-chip non-volatile memory. The Security Monitor block is used to determine when and how Secure RAM resources will be used in the system.

The SCC provides a series of tools to the product developer to allow sensitive data stored in the Secure RAM, such as keys, certificates, and account information, to be protected for storage off the main processor. It is set up so that information is encrypted using a hardware block that only that specific processor can then decrypt.

If the user wants to power down the device and store data externally, the SCC Key Encryption Module (KEM) can encrypt the data. It uses an encryption engine inside the SCC and stores the data in external memory. The data is encrypted using a unique key for each chip. The key is set by burning the appropriate fuses. This key cannot be read by any means, such as software, scanning, or any other physical ways. Thus, the data is secure.

The Security Monitor receives all the alarms from the system. When it detects that there is a security violation, it issues an alarm to the Secure RAM and blocks the Secure RAM. All Secure RAM content is erased and the system cannot use the Secure RAM to get any encryption data from its stolen memory. In this way, if there are problems in the system, secure data is protected.
Now we’ll look at the Secure JTAG controller. Remember the JTAG hacker we saw in the i.MX31 security diagram? JTAG manipulation is one of the known ways that hackers use to gain control over a system. Most devices on the market today do not have any protection against debug port manipulation. Even if a device has the best of software and hardware, it can still be completely vulnerable if the debug port is left open. The debug port is like the “back door.” It can be used to read data, gain control over the operating system, or instruct the processor to jump to a certain location in memory and execute code.

While the debug port leaves a system vulnerable, it is necessary for development and quality checks on production levels. The Secure JTAG Controller provides a way of regulating JTAG access. It allows access to the JTAG port protection mechanism by using different levels of security.

The Secure JTAG Controller has four possible levels of protection. The customer determines the level by blowing (or burning) dedicated e-fuses. The device is at Level 3, “Debug enabled, low security”, at the time of purchase. The customer’s requirements will determine what the level of security should be. The fuse burning is an irreversible operation. Once a security level is set, it is not possible to move it to a lower security level.

The lowest level is “no JTAG security” when the JTAG port is open for all access. This level is configured for samples that are used in ADS and not production. At this level, there is no JTAG alarm, and so the Secure RAM is also open to the JTAG port.

Level 3 is low security. At this level, the JTAG port is open, but if someone connects to the JTAG port, it will issue an alarm in the system.

Level 2 is high security. At this level, access to the JTAG is blocked to all except for authorized entities who can access JTAG if they use the right key. Only debugging tools with the appropriate key will be able to access the JTAG port. It is the responsibility of the customer to burn the key. Customers have three options: they can use their own key, have a unique key for each device, or have the same key for several devices.

The challenge key will be based on 64-bit hardware unique ID. When access to the JTAG is required, the device generates a challenge value transmitted to a secure server. If it is an authorized entity seeking access, the secure server provides the right response. This will be brought back to the device and the JTAG port will be opened. Customers can use this level if they want good security but, at the same time, some JTAG access. A good example of this would be for field return tests.

Level 1 is maximum security. If the customer will not need any access to the JTAG by the debug port, maximum security is recommended. In Level 1, the JTAG port is permanently disabled.
The IC Identification Module (IIM) provides the primary user-visible mechanism that is used to interface with on-chip fuse elements. The fuses are used as unique chip identifiers, cryptographic keys, to mask revision numbers, and for various control signals that require permanent non-volatility.

The IIM and its electrical fuse boxes allow customers to set the required security levels for specific products without the need for different system-on-a-chip (SoC) versions. The IIM is able to override fuse values in software without affecting the fuse element. Override capability can be permanently disabled. The IIM is also able to provide controlled access to the e-fuses, including write-protect and scan-protect (read and program). IIM fuses may be programmed by software, directly by JTAG or indirectly by JTAG, using the ARM® processor.
After the HAB authenticates code on power on/reset, there is no further authentication of code in the external memory. To improve the security of the i.MX31, it has a Run-Time Integrity Checker, or RTIC. Its main purpose is to authenticate external code in run-time. It both verifies the memory contents during system boot and checks memory integrity during application run-time execution.

The core processing element of the RTIC is the hashing engine, which performs an SHA-1 operation on the memory contents. The RTIC acts as an SHA-1 accelerator during the HAB. After the HAB, the RTIC can be configured to Run-time mode. Once it is configured to Run-time mode, it cannot be accessed by software and is a completely independent module.

The RTIC has up to 4 signatures stored internally. These signatures are constantly checked against signatures for the four external memory zones. If the memory contents fail to match the RTIC hash signature at runtime, an error in the Security Monitor is triggered.

The RTIC connects directly to external memory on the bus master. It has the lowest bus priority so it will not interfere with software. Each time the i.MX31 is in an idle state and no one is using the bus, the RTIC will initiate access and read data from the external memory.

In order to protect the device from a “denial of services” attack in which the bus is overloaded and the RTIC is prevented from accessing external memory, the RTIC contains an internal watchdog timer. The timeout value of the watchdog timer can be set before switching the RTIC to Run-time mode. Once the timeout period is up, the RTIC will issue a security alarm.
Let’s review the functions of the various components you have learned about in this module.

Correct.

The HAB is responsible for security checks at boot up; the SCC uses the Secure RAM and Security Monitor to protect sensitive data that is stored in on-chip RAM and in off-chip non-volatile memory; the Secure JTAG Controller regulates JTAG access through various security modes; the IIM and its electrical fuse boxes interface with on-chip fuse elements to perform various security functions; and finally, after the HAB has done its job, the RTIC verifies external memory contents during run-time.
The final feature of the i.MX31 that we will look at is the tamper detection system. It is a small switch that provides evidence of any physical attempt to remove the device’s cover.

The tamper switch is comprised of hardware tamper detector circuitry and is connected to the i.MX31 processor’s GPIO pin.

When a tamper attempt is detected and the switch is triggered, the GPIO pin will issue an alarm in the SCC’s Security Monitor, and the GPIO pin will generate an ARM interrupt. The SCC will then erase the Secure RAM and will transit into Failure mode. This is useful in protecting memory because it is easy to monitor what is happening in—and retrieve data from—external memory.
Module Summary

i.MX31 security features:

- HAB
- SCC
- Secure JTAG Controller
- IIM and e-fuses
- RTIC
- Tamper Detection System

In this module, you learned about various security features within the i.MX31, including HAB, the SCC, the Secure JTAG Controller, the IIM, the RTIC, and the Tamper Detection System.