MPC5748G Gateway, HSM and Secure OTA Update

FTF-AUT-F0347

Alasdair Robertson | Auto MCU Applications EKB

APR. 2014
Agenda

• MPC5748G Device Overview

• Automotive Security Overview
  – Why do we need security in automotive?
  – Security concepts & terminology (encryption, MAC)

• The HSM
  – Introduction to the Hardware Security Module (HSM)
    ▪ What is it, how does it differ from the CSE?
  – Using the HSM
  – Example use cases Including OTA updates

• Demo / Q&A
MPC5748G Device Overview
32-bit High End Body MCU Roadmap

MPC5748x family of MCUs offers:

- **Advanced communication peripherals** – Ethernet AVB support, USB, SDHC, FlexRay™, MOST, higher quantity of LINs, CANs, CAN FD support, etc.
- **Improved performance** - multi-core MCUs options, increased MHz
- **Large flash and RAM** to support increased message handling/code requirements
- **New low power unit** with improved functionality in low power modes
- Support of **functional safety** – ISO26262 process, targeting ASIL-B
- Enhanced **hardware security module**
- **Family concept** – Scalable hardware and software approach within Calypso family and migration path from the widely used Bolero family
Highly Integrated Body Control/Gateway MCU

- **Performance through multicore**
  - Up to three e200 cores built on Power Architecture® technology, with up to 160 MHz performance, allows easy division of tasks in an integrated BCM/gateway system

- **Most diverse set of networking communication protocols**
  - Ethernet with AVB support, FlexRay, MLB, USB, up to eight CAN with CAN Flexible Data Rate (FD) up to 18 LIN, SDIO interface, I²S all supported on a single-chip solution

- **Flexible memory options**
  - Up to 6 MB flash and 768 KB of embedded SRAM provide suitable storage to maintain the local BCM/gateway application functionality, handle message buffering, and also store additional flash images for other nodes in the vehicle
Driving Low Power Consumption

- **New low-power unit**
  - Allows for increased functionality in a lower power state, reducing current consumption by over 30% for a typical cyclic wake-up application over previous generation devices
  - Provides a mechanism to bypass entire platform while supporting a smaller set of peripherals (1x CAN, LIN, SPI, ADC, timer, etc.) thereby providing very low power execution modes

- **Analog comparator**
  - Typical periodic monitoring routines can be fully handled in Standby mode, offering a significant improvement in power consumption

- **Pretended networking support**
  - Enabled through advanced filtering, wakeup capabilities and CAN availability in low power modes
Addressing Functional Safety and Security

• Designed with the ISO26262 process in mind
• Freescale’s Safe Assure functional safety program:
  - Safety Process: Integrating functional safety into dev process
  - Safety Hardware: Built in self tests, error code correction, etc
  - Safety Software: Autosar MCAL, OS, core self tests, etc
  - Safety Support: Training, documentation and tech support

• Designed to support next generation security needs
  - Security gatekeeper, immobilizers, component protection, protection of data sets
• Hardware Security Module (HSM)
  - Meets SHE and EVITA medium spec requirements
  - Dedicated, programmable security core
  - Secure flash and SRAM
  - Hardware Cryptographic module – AES-128, random number generator etc.
  - Helps protect security keys, secure boot up, tamper detection, advanced debug support, etc.
MPC5748C/G Development Device and High End Gateway/BCM Solution

Applications:
- High end gateway and body modules

Key Characteristics:
- 2x e200z4 + 1x z2 cores, FPU on z4 cores
- 160 MHz max for z4s and 80 MHz on z2
- HSM Security Module option supports both SHE and EVITA low/medium standard
- Media local bus supports MOST communication
- 2 x USB 2.0 (1 OTG and 1 Host module) support interfacing to 3G modem and infotainment domain
- 2x Ethernet 10/100 Mbps RMII, MII, +1588, AVB
- Ethernet switch
- CAN module optionally supports CAN FD
- SDHC provides standard SDIO interface
- Low power unit provides reduced CAN, LIN, SPI, ADC functionality in low power mode
- Designed to ISO26262 process for use in ASIL B
- -40º to +125ºC (ambient)
- 3.0V to 5.5V

Packages:
- 176 LQFP, 256 BGA, 324 BGA

*Mixture of internal and external channels. Features available depend on package and device version
Automotive Security Overview
Why Do We Need Security in Automotive?
Introduction: What’s Driving Automotive Security - 1

- **Safety:**
  - 2010 publications highlight how vehicle networks could be hacked from inside the cabin. With the connected car, the attacker could be anywhere.
  - Potentially hackable software is also used in safety critical systems (e.g. brakes, steering, etc.).
  - Safety-relevant systems and their software must be secure.

- **Reliability / Integrity:**
  - Counterfeit parts and ECUs can impact vehicle reliability (and safety).
  - After-market “add-on” devices could interact with the car network, impacting vehicle reliability and safety.
  - Modifying one node on the car network could potentially cause malfunction or breakdown of complete system.
Introduction: What’s Driving Automotive Security - 2

- **Protection of (financial) assets**
  - **For the OEM:**
    - Ensure ECUs and other spare parts only work if genuine
    - Protection of high $ value features that are enabled via software switch
    - Potentially achieve lower value insurance rates through theft protection, making car more attractive to consumers
  - **For the owner:**
    - Vehicle remains safe and reliable, retaining $ value
    - Mileage manipulation considered impossible

- **Privacy/Confidentiality**
  - Car data, driver location, personal driver data and preferences must remain confidential
  - Research example at University of South Carolina: TPMS signals hacked and used to eavesdrop on car movement
Introduction: Potential Security Threat Examples

- Hack of immobilizer / remote entry
  - Firmware modification
  - Unauthorized ECU replacement
  - Access to proprietary software

- Modification of vehicle network (CAN, LIN, FlexRay, Ethernet, etc.)
  - Malicious application
  - Mileage manipulation
  - Reading of personal data

- Eavesdropping
- Service denial
- Modification of comms

Base station / Infrastructure/ Other cars
Security Concepts & Terminology
Concepts: **ECB (Electronic Code Block) Encryption**

- **Simple 128-bit encryption (128-bit key for 128-bit data)**
  - There is no dependency between encrypted data blocks
  - The same data encrypted with the same key will always give the same answer

- **So what’s the problem?**
  - Data is in independent 128-bit blocks so easier to hack over time
  - Repetitive data streams can be de-coded

- **Example below where encrypted image can still be made out**
  - (Especially in data that divides by 128 eg 32-bit colour image)
Concepts: CBC (Cipher Block Chaining) Encryption

- Advanced 128-bit encryption
  - Previous encrypted data value is fed into next data flow (XOR’d)
  - Same data encrypted twice with same data will result in different answer
  - 1st encrypted 128-bit data block (data1 in diagram) needs IV (initial value) which is also needed for decryption
Concepts: **CMAC (Cipher) Message Authentication Code**

- Used to verify data integrity (data has not been modified / is from correct ECU)
- Run an AES encryption algorithm (128-bit key) over a block of data to generate a 128-bit CMAC
  - Calculated over 128-bits of data at a time using the defined key
  - The output (CMAC) from the previous calculation is fed into the next (XOR’d)
  - This is repeated until all the data has been processed at which point the CMAC is generated (if the last block of data is < 128-bits, only part of the key is used)
Concepts: Random Number Generation

• Use-case
  - Key generation
  - Prevents re-play attacks

• Pseudo Random Number Generation (PRNG)
  - Reproducible value generated by a deterministic algorithm
  - Digital IP
  - Fast

• TRUE Random Number Generation (TRNG)
  - Value generated via measurement of physical effects (e.g. thermal noise)
  - Includes analog elements (e.g. simple A/D-converter)
  - Slow
# Security Examples

## Concepts:

### Security Goals

<table>
<thead>
<tr>
<th>Security Goal</th>
<th>Description</th>
<th>Examples</th>
<th>Security Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integrity</strong></td>
<td>Detection of unauthorized modification</td>
<td>Firmware, application, vehicle network</td>
<td></td>
</tr>
<tr>
<td><strong>Authenticity</strong></td>
<td>Ensure origin of data</td>
<td>Firmware, application, immobilizer</td>
<td></td>
</tr>
<tr>
<td><strong>Confidentiality</strong></td>
<td>Protect against disclosure of (arbitrary) data</td>
<td>Motor management algorithm</td>
<td></td>
</tr>
<tr>
<td><strong>Privacy</strong></td>
<td>Protect against disclosure of personal data</td>
<td>Location, travel, phonebook, contacts</td>
<td></td>
</tr>
</tbody>
</table>

### Challenge-Response

- Host (ECU) issues random number to client (keyfob)
- Client encrypts data and returns to host (client must know correct key to encrypt data correctly)
- Host validates returned data and either allows or denies access.
Introducing the HSM
## Security Standards

|                  | HIS (SHE)  
|------------------|-------------------------------------------------|
| **Who**         | Audi, BMW, Escrypt  
|                  | (Published as a standard by HIS)  
| **Coverage**    | Single Security Level  
| **Encryption**  | AES-128  
| **Secure Boot** | Yes  
| **Firmware**    | ROM  
| **Secret Keys** | No  
| **FSL Implementation** | CSE  
| **Unique ID**   | Yes  

|                  | EVITA  
|------------------|-------------------------------------------------|
| **Who**         | Co-founded by EU in partnership with companies such as BMW, Continental, Escrypt  
| **Coverage**    | Spec covers 3 security levels (low, **med**, high)  
| **Encryption**  | AES-128  
| **Secure Boot** | Yes  
| **Firmware**    | Flash (2 x 64K code, 16K data)  
| **Secret Keys** | 2 (programmed at test)  
| **FSL Implementation** | HSM  
| **Unique ID**   | Yes  

http://portal.automotive-his.de
http://www.evita-project.org
**HSM vs CSE**

- **CSE** (Cryptographic Services Engine)
  - Freescale’s implementation of HIS (SHE) spec
  - CSE firmware is ROM based (no custom firmware possible)
  - Contains pre-defined keys with set functions. Managed by the CSE hardware

- **HSM** (Hardware Security Module)
  - Freescale’s implementation of the EVITA spec
  - HSM firmware is in flash and is totally user defineable
  - Keys are defined and managed by the firmware
    - Number is limited only by available space in secure flash
    - Only fixed (pre programmed) keys are Secret keys and Device ID
  - Intention is that freescale will make CSE firmware available
    - Exact details still to be confirmed
    - Demo is running preliminary version of this firmware

*All use case examples in this presentation assume the HSM with “CSE Like” functionality*
The Hardware Security Module (HSM) on MPC5748G

Features:
- e200z0h core (80 MHz)
- 4Kbytes Instruction cache
- Secure Debugger Interface
  - Via PASS module and lifecycle
- C3 AES-128 Hardware Cryptographic Module with Random Number Generator, DMA

Memory:
- SRAM 32 Kbytes
- Flash:
  - Code: 2 x 64 Kbytes, 1 x 16Kbytes
  - Data: 2 x 16 Kbytes

Firmware:
- User programmable, resides in dedicated flash
C3: AES-128 Features

Features
- DMA support for automated data fetching
- Random number generation
- Cipher Modes:
  ▪ ECB, CBC, OFB, CFB, CTR, XTS, CMAC, GCM and OFB for PRNG

Hardware Implementation
- Approx 70 times faster at decrypting a CBC encoded image than an e200z4 at 160MHz (no cache)
- For e200z0 software, relative hardware performance is approx 140 times faster!
MPC5748G Additional Security Measures

PASS (Password and Device Security Module)
- Provides password-based read and write protection for each flash block
- Provides password access to Debug Interface based on lifecycle and flash censorship state

TDM (Tamper Detect Module)
- Allows flash blocks to be set as OTP
- Provides a mechanism to prevent flash areas being erased without first writing a record to the TDM
  - Does not prevent flash erase but provides a record / history of erase.
  - Up to 6 Tamper detect regions can be configured

Lifecycle:
- Enable / disable security features based on current “State”
- Controls UTEST OTP, Debug access, passwords etc

---

External Use | 25
Using the HSM
Secure Boot Overview – Check Boot Block Authenticity

HSM firmware runs at reset and could be programmed to perform a secure boot over user-defined flash area.

Step 1: HSM uses “Boot KEY” to calculate MAC value over user-defined boot block in flash.

Step 2: MAC is compared to stored BOOT MAC. If identical, secure boot OK and Boot Keys available.

Option for BAF to suspend boot until HSM secure boot has completed.

- Could implement “Chain of trust” to decrease initial boot time
- HSM can issue optional system reset on secure boot failure
HSM Keys Example (Based on CSE Firmware)

Remember that HSM keys are defined by firmware, CSE keys are hard defined

- General Points about Keys:
  - Keys cannot be viewed (except the Device ID) but keys can be validated against known key value
  - There are keys for specific functions like secure boot, CMAC generation and encryption
  - Keys can be write protected so they can never be changed or erased
  - The UID and Secret keys are hard coded in the HSM

<table>
<thead>
<tr>
<th>Key Name</th>
<th>Description</th>
<th>Constraints</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>Provides a unique ID for every device</td>
<td>Factory programmed by Freescale Cannot be erased or re-programmed</td>
<td>Always</td>
</tr>
<tr>
<td>Master Key</td>
<td>Used for system recovery</td>
<td>User-programmable If known, can erase any other non WP key</td>
<td>Depending on flags</td>
</tr>
<tr>
<td>Boot Key</td>
<td>Key used for Boot MAC calculation</td>
<td>User programmable</td>
<td>Depending on flags</td>
</tr>
<tr>
<td>Boot MAC</td>
<td></td>
<td>Automatically set on 1st secure boot</td>
<td>Depending on flags</td>
</tr>
<tr>
<td>User Keys</td>
<td></td>
<td></td>
<td>Depending on flags</td>
</tr>
<tr>
<td>Secret Keys</td>
<td>Used for local module control</td>
<td>Factory-programmed by Freescale. Key value cannot be read</td>
<td>Always</td>
</tr>
</tbody>
</table>
HSM Keys 2 – CSE Implementation

• Each key has the following associated attributes which may be used to limit the use of a specific key:
  - **Write Protect (WP)** – Can be used to make a key so it can be updated or erased. Use with caution. Will render key unable to be updated
  - **Boot Protect (BP)** – A key can be disabled if the BOOT_MAC calculation did not match what was previously stored in the BOOT_MAC key slot
  - **Debugger Protection (DP)** – A key can be disabled if a debugger has been or is currently attached
  - **Wildcard Updates (WC)** – A key can be protected from Wildcard Updates (UID’=0).
    - If enabled, the UID is required for key updates.
  - **Key Usage (KU)** – A key is assigned to be use for either encryption/decryption (KU=0) or for MAC generation/verification (KU=1)
    - A **counter** is stored with each key in secure flash and this must be incremented on every update (this helps prevent replay attacks)
    - A **checksum** is stored with each key
Use Case Examples
Chain of Trust

Validate the integrity / authenticity of blocks of code after secure boot (reduce SB time)

Pre-requisites:
- Successful secure boot must have occurred

Method:
1. HSM performs successful secure boot (unlocking keys)
2. Perform a CMAC on next block of data to “validate” using one of the previously stored keys and compare this with a stored CMAC
3. If calculated CMAC = stored CMAC, the area is validated.
Immobilizer

Allow only a valid key to start / unlock the car (challenge / response)

Assumptions:
- Secure boot has completed and the required keys are available to the HSM
- The car key and ECU have been programmed with the same key available to both

To validate the car key:
- The ECU sends a random number to the key
- The car key encrypts this using the defined KEY and returns this encoded value back to the ECU
- The ECU verifies the returned value
- The car can only be unlocked / started on a match
- Avoids replay attacks / key synchronisation issues
Secure Communication - Sensor

- Random number: protects against replay attacks.
- Encryption: protects against eavesdropping.
- Random number and encryption: ensures data integrity and authenticity.

Step 1: Central ECU obtains random number and sends it to sensors ECU (e.g., after power-on of car)
Step 2: Sensor ECU reads sensor value and asks HSM module to encrypt it and the received random number (using key #x)
Step 3: Sensor ECU sends encrypted message to central ECU.
Step 4: Central ECU asks HSM module to decrypt received message (using key #x).
Step 5: Central ECU checks sent random number vs. received/decrypted random number.
Secure Communication – Secure OTA Update

- The Decrypted firmware could then update a distributed module in the vehicle (being re-encrypted if necessary to maximise security in transmission in the vehicle network)
- For enhanced security, Unique ID can be applied to packet and Random number can be used against replay attacks

juergen.frank@freescale.com
So why the MPC5748G?

**Gateway features**
- Large amount of comms peripherals ideal for high end gateway
- Ethernet, USB, SDIO could enable wireless connectivity

**Security**
- HSM, PASS, TDM and Lifecycle together provide highly secure platform

**Multicore**
- Possibility to add additional features (Gateway and Body Control) in one ECU saving module cost

**Low Power**
- LPM unit with dedicated low power core
The Demo!