Mechanisms for Trusted Code Execution

Bob Waskiewicz
Introduction
Root of Trust

- Root of Trust
  - Whole process depends on the integrity of the first module
  - First module protected by hardware: Trust Anchor
Application Code

Secure Execution

• Application code should be written and executed in a secure manner
  • Start with security coding ‘Best Practices’ including software layering

• Use of appropriate mechanisms
  • Use derived keys
  • Use process partitioning as appropriate

• Chain of Trust
  • At start-up, each ‘stage’ verifies the integrity and/or authenticity of the next stage
  • Chaining of HASH / integrity checked
    • Only if check is OK, control is transferred to the next stage, otherwise system is halted
Secure Storage
Data at Rest

- Access to data must be carefully controlled and restricted to authorized persons, machines, and processes
  - Data at rest – protected
    - Encrypt as necessary; All keys, master keys, session keys used for secure communications, passwords etc.
    - Encrypt sensitive application data
    - Encrypt sensitive customer personal data
  - Destroy keys that are no longer valid
• IoT ecosystem relies on confidential and trusted communications
  • Data integrity
  • Encryption end to end

• Device authentication
  • Using Public Key Infrastructure (PKI) solutions
IoT Attacks
Local Versus Remote

Non Invasive
- Logical attacks
- Side channel attacks

Product Invasive
- Uncontrolled state device
- Fault Injection

Silicon Invasive
- Probing
- Reverse Engineering

ATTACKS COST and EXPERTIZE
COMPONENT and SYSTEM SECURITY LEVEL
ARM Cortex® / STM32
Security Features
IoT Devices

Three Types of CPU Based Devices

• MPU: Microprocessors
  • High-performance CPUs (>GHz) using lots of memory (non-volatile and volatile) >GB
  • Large Storage
  • Typically running complex file operating systems/General computing devices

• MCU: Microcontrollers
  • Lower performing CPUs (~MHz) with self contained internal memories (< few MB)
  • Typically running native code or RTOS
  • Focused Application Devices

• Secure MCU: Secure Microcontrollers
  • Sub-class of MCUs, limited peripheral set
  • Design for security and contain specific hardware counter attack measures
  • Use certifiable processes from inception to manufacturing
  • Limited to specific security applications (SmartCard, SIM, eSE, Authentication)
• Processor
  • 80 MHz ARM® Cortex® M4

• Memory Support
  • Internal 1MB Flash
  • Internal 128K SRAM
  • QuadSPI
  • External Memory Controller

• Connectivity
  • I2C, SPI, USB

• Analog

• Control
Cortex®-M Cores
M0/M0+, M3, M4, and M7
ST Has Licensed ALL Cortex®-M Cores

Source: ARM
Cortex®-M0 / M0+ Microarchitecture

- **Thumb-2 Technology**
- **Integrated configurable NVIC**
- **Microarchitecture**
  - 3-stage pipeline with branch speculation
  - 1x AHB-Lite Bus Interfaces
- **Configurable for ultra low power**
  - Deep Sleep Mode, Wakeup Interrupt Controller
- **Flexible configurations for wider applicability**
  - Configurable Interrupt Controller
  - Optional Debug and Trace
  - M0, No Memory Protection Unit
  - M0+, Optional Memory Protection Unit

ARMv6M Architecture
Cortex®-M3 Microarchitecture

ARMv7ME Architecture

- Thumb-2 Technology
- Integrated configurable NVIC
- Microarchitecture
  - 3x AHB-Lite Bus Interfaces
- Configurable for ultra low power
  - Deep Sleep Mode, Wakeup Interrupt Controller
- Flexible configurations for wider applicability
  - Configurable Interrupt Controller
  - Optional Debug and Trace
  - Optional Memory Protection Unit
Cortex®-M4 Microarchitecture

- Thumb-2 Technology
- DSP and SIMD extensions
- Optional single precision FPU
- Integrated configurable NVIC

Microarchitecture
- 3-stage pipeline with branch speculation
- 3x AHB-Lite Bus Interfaces

Configurable for ultra low power
- Deep Sleep Mode, Wakeup Interrupt Controller
- Power down features for Floating Point Unit

Flexible configurations for wider applicability
- Configurable Interrupt Controller
- Optional Debug and Trace
- Optional Memory Protection Unit
Cortex®-M7 Microarchitecture

- Thumb-2 Technology
- DSP and SIMD extensions
- Optional Double precision FPU
- Dual-issue superscalar architecture
- Microarchitecture
  - 6-stage pipeline with branch speculation
  - AXI-M Bus Interface with cache memory
- Configurable for ultra low power
  - Deep Sleep Mode, Wakeup Interrupt Controller
  - Power down features for Floating Point Unit
- Flexible configurations for wider applicability
  - Configurable Interrupt Controller
  - Optional Debug and Trace
  - Optional Memory Protection Unit
## Today - STM32 Portfolio

More than 30 product lines

<table>
<thead>
<tr>
<th>Category</th>
<th>CoreMark</th>
<th>Frequency (MHz)</th>
<th>MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High-performance</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cortex-M0</td>
<td>398</td>
<td>120</td>
<td>150</td>
</tr>
<tr>
<td>Cortex-M0+</td>
<td>106</td>
<td>48</td>
<td>38</td>
</tr>
<tr>
<td>Cortex-M3</td>
<td>177</td>
<td>72</td>
<td>61</td>
</tr>
<tr>
<td>Cortex-M4</td>
<td>245</td>
<td>72</td>
<td>90</td>
</tr>
<tr>
<td>Cortex-M7</td>
<td>273</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td><strong>Mainstream</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cortex-M0</td>
<td>1082</td>
<td>216</td>
<td>462</td>
</tr>
<tr>
<td>Cortex-M0+</td>
<td>93</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>Cortex-M3</td>
<td>93</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>Cortex-M4</td>
<td>245</td>
<td>72</td>
<td>90</td>
</tr>
<tr>
<td>Cortex-M7</td>
<td>273</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td><strong>Ultra-low-power</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cortex-M0</td>
<td>75</td>
<td>32</td>
<td>26</td>
</tr>
<tr>
<td>Cortex-M0+</td>
<td>93</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>Cortex-M3</td>
<td>273</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>Cortex-M4</td>
<td>245</td>
<td>72</td>
<td>90</td>
</tr>
<tr>
<td>Cortex-M7</td>
<td>1082</td>
<td>216</td>
<td>462</td>
</tr>
</tbody>
</table>

**Longevity Commitment:** 10 Years
# STM32 Family vs. Security Features

<table>
<thead>
<tr>
<th>ST Family</th>
<th>Debug Access Port</th>
<th>RESET Register</th>
<th>FLASH WRP</th>
<th>FLASH Mass ERASE</th>
<th>Tamper Pins</th>
<th>CRC Hardware</th>
<th>96-Bit Unique ID</th>
<th>Crypto Library Support</th>
<th>Memory Protection Unit(MPU)</th>
<th>FLASH RDP</th>
<th>TRNG</th>
<th>AES Hardware Accelerator</th>
<th>FLASH PCROP</th>
<th>HASH Hardware Accelerator</th>
<th>Firewall</th>
<th>SRAM RDP</th>
<th>FLASH ECC</th>
<th>Sys Clock (MHz)</th>
<th>ARM Cortex®</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32 F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>72</td>
<td>M3</td>
</tr>
<tr>
<td>STM32 F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>72</td>
<td>M4</td>
</tr>
<tr>
<td>STM32 F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48</td>
<td>M0</td>
</tr>
<tr>
<td>STM32 L1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
<td>M3</td>
</tr>
<tr>
<td>STM32 F2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>120</td>
<td>M3</td>
</tr>
<tr>
<td>STM32 F4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>180</td>
<td>M4</td>
</tr>
<tr>
<td>STM32 F7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>216</td>
<td>M7</td>
</tr>
<tr>
<td>STM32 L0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
<td>M0+</td>
</tr>
<tr>
<td>STM32 L4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80</td>
<td>M4</td>
</tr>
</tbody>
</table>

*ARM TRM, RMxxxx, AN4246, AN3371, RMxxxx, UM1924, AN4838, AN178, AN4246, AN4230, U1924, AN4246, AN4701, AN4758, UM1924, AN4729, AN4729

ARM Memory Protection Unit

Overview

- Optional feature available on Cortex™-M cores
- Enforce privilege rules on read / write / execute only or no-access
- Memory areas defined by regional (8 Regions) parameters for memory isolation
- Upon violation, core generates a hard-fault or core “lock-up”
ARM Memory Protection Unit

Why Use an MPU?

- Prevent processes from accessing memory that has not been allocated to them
- Protect applications from a number of potential error such as the detection of stack overflows
- Protect from invalid execution by RTOS tasks and protect data from corruption
- Protect system peripherals from unintended modification
Embedded Systems Conference

ARM Cortex® Debug

Debug Access Port (DAP)

• Serial Wire Debug or IEEE JTAG Debug
• Embedded break / watch capabilities for easy Flashed application debugging
• Includes a Serial Wire Viewer for low bandwidth data trace
• Includes an Embedded Trace Module for system core clock debugging

• DAP is ALIVE ALL the time → After RESET, when core enters low power mode, and when non-core security features are enabled
  • The BKPT assembly instruction will cause the core to enter Debug state

More pins available for the application
• Creates a specific “trust area” of code with own memory isolated from all other code areas

• Has a single gateway interface to enter the Firewall. Any access other than the proscribed gateway interface results in a system reset

• Ideal for protecting algorithmic IP separate from the rest of the internal application, and performing security sensitive operations (i.e. HASHing)

• Intrusive detection into a protected area generates a MCU reset
  • Includes DMA and / or Interrupt intrusions

• Configured at Start and remains active until the next system reset
STM32 Memory Features

Overview

• Readout Protection (RDP)
  • Level 0: no readout protection
  • Level 1: memory readout protection
  • Level 2: chip readout protection

• Proprietary code Read Out Protection (PcROP)
  • Specific configurable area
  • 1 each per Flash sector

• Write protection (WRP)
  • 1 each per Flash / SRAM* sector

• Error Correction Code (ECC)

Flash code is protected when accessed through the JTAG interface or when the Boot is different from Flash memory

Flash code is only executable, not readable

Flash code is protected from unwanted write/erase operations

Robust memory integrity and safety
STM32 Memory Features

Readout Protection (RDP)

• Readout Protection **Level 0** (no protection, factory default)
  - All operations (R/W/Erase) are permitted on Flash memory, SRAM, and Backup Domain

• Readout Protection **Level 1**
  - If the selected boot mode is User Flash and if no debugger access is detected (no JTAG):
    - All operations (R/W/Erase) are permitted on the Flash memory, SRAM, and Backup registers
    - If the selected boot mode is not user Flash, or if a debugger access is detected (JTAG):
      - ALL operations (R/W/Erase) to Flash memory, SRAM, and Backup registers are blocked and a hard fault interrupt is generated.
STM32 Memory Features

Readout Protection (RDP)

• Readout Protection **Level 2** (JTAG fuse blown)
  • All protections provided by Level 1 are active
  • Boot from RAM or System memory is no longer possible (only from User Flash memory)
  • The physical JTAG interface is disabled
    • Factory Failure Analysis Report is limited, thus ensuring there is no factory backdoor
  • If the selected boot mode is User Flash memory
    • All operations (R / W / Erase) are permitted on the Flash memory, backup registers and SRAM
  • Level 2 can **NOT** be reversed
STM32 Memory Features

Readout Protection (RDP)

Level 0

Level 1

Level 2

* Only on STM32L4
** Only on STM32L0

Debug tools / Boot from SRAM / Boot from system memory
STM32 Memory Features

RDP Transition Scheme

- **Level 0**
  - Option byte mods are allowed
  - Can transition to Level 1 or Level 2

- **Level 1**
  - Option byte mods are allowed.
  - Can transition to Level 0 or Level 2
    - Level 0 → Mass erase of user Flash, backup registers and newer device SRAM sector

- **Level 2**
  - Option bytes are frozen
  - No transition possible
## STM32 Memory Features

### Access status vs. readout protection level

<table>
<thead>
<tr>
<th>Area</th>
<th>Protection Level (RDP)</th>
<th>Access rights when Boot = User Flash</th>
<th>Access rights when Boot ≠ User Flash Or Debug Access detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main memory</td>
<td>1</td>
<td>R/W/E</td>
<td>No Access</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>R/W/E</td>
<td>-</td>
</tr>
<tr>
<td>System memory</td>
<td>1</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>Option bytes</td>
<td>1</td>
<td>R/W/E</td>
<td>R/W/E</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>Backup registers</td>
<td>1</td>
<td>R/W</td>
<td>No Access</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>R/W</td>
<td>-</td>
</tr>
<tr>
<td>SRAM2*</td>
<td>1</td>
<td>R/W</td>
<td>No Access</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>R/W</td>
<td>-</td>
</tr>
</tbody>
</table>

W: Write  
R: Read  
E: Erase
STM32 Memory Features

Proprietary code Read Out Protection (PcROP)

- Third-parties can develop and sell specific software IPs for STM32 MCUs
  - Prevents malicious software or a debugger from reading sensitive code

- Customers may use these software IPs for development with / in their own application code
  - The PCROP Flash memory area is executable only
    - R / W / Erase operations are not permitted

Protect confidentiality of software IP code whatever the RDP level
STM32 Memory Features

Firewall vs. PCROP

- **PCROP**: Prevents snooping execution, however, the results of execution (variables, core registers) are not protected.

- **PCROP**: RAM not protected

- **PCROP**: Set at POR via option byte configuration, no need to initialize.

- **Firewall**: Dynamic execution protection, (open / close), Flash and SRAM, **AFTER** initialization.
STM32 Memory Features

Flash and SRAM Write Protection (WRP)

- The Flash write-protected area is defined on a per sector basis via the STM32 option bytes setting.
- In newer STM32 devices, the WRP area is defined by “start” and “end” addresses.
- In the STM32L4 devices, a SRAM section is write protectable.
STM32 Memory Features

Robust memory integrity and safety

• **ECC** (Error Code Correction): 8 bits long for a 64-bit word
  • Single error correction: ECCC bit, interrupt generation
  • Double error detection: ECCD bit, NMI generation
  • Failure address and bank saved in FLASH_ECCR register

• Programming granularity is 64 bits (really 72 bits incl. 8-bit ECC)
STM32 96-bit Unique ID

Features

• Unique Device Identifier installed at the ST factory
  • Provides a reference number unique for any STM32
  • It will not repeat for many years

• The Unique ID is suited for:
  • Generating a serial number via an algorithm
  • Combining with cryptographic primitives to increase security before programming STM32 Flash, key derivation.
  • Used as part of device authentication during secure boot process
STM32 Cyclical Redundancy Check

Overview

- Used to get a CRC code from 8, 16, or 32-bit data word
- Verify data integrity
  - Generate a software code signature
- Can be used direct by core or via DMA
STM32 Advanced Encryption Standard

- Hardware acceleration that transforms original plaintext to unreadable ciphertext
  - Supports
    - Several standard operation modes and key sizes
    - Supports several standard AES chaining modes
    - Supports data swapping
    - Supports DMA
  - Reduces CPU time:
    - typical 100 - 200 sysclk cycles
STM32 Advanced Encryption Standard

Block Diagram

NIST FIPS 197 compliant implementation of AES

AES Accelerator

- AES operation mode:
  - Encryption
  - Decryption
  - Key derivation
  - Key derivation + decryption
  - Key: 128-256-bit

- AES chaining mode:
  - ECB
  - CBC
  - CTR
  - GCM
  - GMAC
  - CMAC

Data In

DMA request for incoming data transfer

Data swapping

Data swapping

DMA request for outgoing data transfer

Data out
STM32 HASH Processor

Overview

• Hardware acceleration that transforms original plaintext to an unreadable Message Digest
  • Supports
    • Supports several HASH standards
    • Supports data swapping
    • Supports DMA
  • Reduces CPU time:
    • typical 50-66 sysclk cycles

Compliant with:
FIPS Pub 180-2
Secure HASH Standards (SHA-1*, SHA-224, SHA-256)
IETF RFC 1321 (MD5*)
STM32 Crypto Library

Software ONLY

- STM32 Firmware Crypto Library V3.1.0
  - All STM32 series supported: STM32F0, STM32F1, STM32F2, STM32F3, STM32F4, STM32F7, STM32L0, STM32L1 and STM32L4
  - All algorithms are based on firmware implementation without using any hardware acceleration
  - The STM32 Firmware Crypto Library is distributed by ST as an object code library, accessed by the user application through an API
  - The library is compiled for Cortex® M0, M0+, M3, M4, and M7 cores
STM32 Crypto Library

Hardware Acceleration

• STM32 Hardware Acceleration Crypto Library V3.1.0
  • Support all STM32 series with hardware acceleration (AES and / or HASH): STM32F2, STM32F4, STM32F7, STM32L0, STM32L1 and STM32L4
  • Support the algorithms based on firmware implementation with hardware acceleration (Hybrid)
  • The STM32 Hardware Acceleration Crypto library is distributed by ST as an object code library, accessed by the user application through an API
  • The library is compiled for Cortex® M0, M0+, M3, M4, and M7 cores
X-CUBE-CRYPTOLIB library is ready for use in security-conscious STM32-based applications

- Helps customers prove the security of their new products quickly and cost-effectively
- Ready for use STM32-based applications including IoT
- Removes the burden of algorithm validation
- Allows OEMs to fasten their security certification process
- Includes all the major algorithms for encryption, hashing, message authentication, and digital signing
STM32 True Random Number Generator

Features

• 32-bit Random Number Generator based on a noise source
  • A 32-bit random number can be generated at an average frequency of AHB / xx

• Three Flags:-
  • Valid random data is ready
  • An abnormal sequence occurs on the seed
  • A frequency error is detected when using a PLL48 RNG clock source

• One interrupt
  • To indicate an error (an abnormal seed sequence or a frequency error)
STM32 True Random Number Generator

Block Diagram

RNG

- 32-bit random data register
- LFSR (Linear Feedback Shift register)
- Analog seed

Error management
- Clock checker
- Fault detector

Flags
- DRDY
- SECS
- SBS
- CECS
- CBIS

Interrupt Enable bit
- IM

RNG interrupt to NVIC
STM32 Reset

Features

Safe and flexible reset management without external components

- Manages three types of reset:
  - System reset
  - Power reset
  - Backup domain reset

- Peripherals have individual reset control bits in the RCC_CSR register
STM32 Reset

Source

- No external components are needed due to internal filter and power monitoring
• Backup Domain Contains
  • A Calendar RTC
  • xx Data Bytes, Backup SRAM
  • Separate 32kHz oscillator for RTC

• Tamper Detection Pins
  • RESETs all RTC backup registers and Backup SRAM
  • Time stamp event
• External Anti Tamper Features
  • Pattern control (Timer control)
    • External connection between I/O pair – pattern out / in pins
  • Voltage control
    • DAC output / ADC input + ADC watchdog
  • Temperature anti tamper
    • Use internal temperature sensor (5-10 °C accuracy)
  • Under / Over-voltage tampering
    • Analog WDG on BandGap voltage (supply voltage measurement)
STM32 Secure Firmware Update Overview
AN4023
STM32 Boot Loader Options

Native (ICP) and IAP Methods

1. JTAG Programmer
2. USB Device
3. Bluetooth LE Radio
# STM32 Boot Modes

Hardware and Software Controlled (Security Level 0/1)

<table>
<thead>
<tr>
<th>Boot mode selection</th>
<th>Boot mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BOOT1/nBOOT1</strong> (*option bit)</td>
<td><strong>BOOT0</strong> (pin)</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- When Boot mode → User Flash memory & **BFB2** option bit ENABLED:
  - Boot is done from Flash memory Bank 1 or Bank 2.
# Bootloader

## Native Boot Peripherals (Security Level 0/1)

<table>
<thead>
<tr>
<th>Protocol</th>
<th>STM32L4 I/Os and Comments</th>
<th>Comments</th>
</tr>
</thead>
</table>
| **USART** | USART1 on pins PA9 / PA10  
                USART2 on pins PA2 / PA3  
                USART3 on pins PC10 / PC11 |  
| **USB**   | USB DFU interface on pins PA11 / PA12 | Boot-loader checks if HSE present : USB clock is HSE  
                        If no Boot-loader checks if LSE present : USB clock is  
                        MSI auto-trimmed with LSE |
| **CAN**   | CAN1 on pins PB8 / PB9 |  |
| **SPI**   | SPI1 on pins PA4 / PA5 / PA6 / PA7  
                SPI2 on pins PB12 / PB13 / PB14 / PB15 |  |
| **I2C**   | I2C1 on pins PB6 / PB7  
                I2C2 on pins PB10 / PB11  
                I2C3 on pins PC0 / PC1 | I²C slave address is 0x86 |
Custom Boot Loader

Benefits

• An alternative to ICP load mechanisms giving additional flexibility

• Tailored to the application

• Can use non-published load methods

• Ability to use other interfaces rather than the native load interfaces

• Must be done when using STM32 Level 2 Security
AN4023 Secure Firmware Update

Overview

• Secure Loader Installation (1\textsuperscript{st} Trust Event)
  • Secure Firmware Insertion (ST Factory / OEM / CM / Distributor)
  • Public / OEM Key installation
  • Lock down the device (Level 2)

• Provisioning (2\textsuperscript{nd} Trust Event)
  • Firmware Activation/installation (OEM / CM / Distributor)
  • Done as part of a Hardware Secure Module (HSM)

• Secure Boot

• Secure Loader
Secure Provisioning
Signed Firmware → One-Way Verification

1. Host Public Key

2. Hash

3. Sign

4. Application Firmware Package

5. IAP verifies FW Image w/signature

- Host Private Key
- Application FW Image
- FW Signature
- FW Hash
- IAP Code
- Host Public Key
- Application FW Image
- FW Signature
Secure Provisioning

Signed Firmware → Two-Way Verification

1. Host signs FW image
2. Application Firmware Package
   - Application FW Image
   - FW Signature
3. IAP verifies FW Image w/signature. Signs image + dev id
   - Host Public Key
   - IAP Private Key
   - Application FW Image
   - FW Signature
4. FW+Dev ID Signature
5. Host verifies FW + Dev ID signature

Backroom Tools
- Host Private Key
- IAP Public Key

Device ID
- IAP Code
- Host Public Key
- IAP Private Key
- Application FW Image
- FW Signature
STM32 Secure Boot

Best Coding Practices

- Secure Boot Application
  - Authenticate the STM32 device
  - Enable the IWDG (note LP modes with MCU_DBG features)
  - RESET Recovery Check
  - Disable the ARM DAP Configuration
  - Initialize the Firewall and / or MPU
    - HASH The Loader firmware
    - Initialize SRAM (zero)
  - At each step a GO / NO-GO decision is made by Secure Boot Application.
STM32 AN4023 Secure Loader

Flow Chart

START

Is there a fraud?

Yes →

Set protection level to 0: Erase the flash memory

No →

Is the device secured?

Yes →

Is the firmware valid?

Yes →

Is there an upgrade request (is the key pressed)?

No →

Execute the existing firmware

Yes →

Load secure firmware

Is the device personalized?

Yes →

Personalize the chip

No →

No →

Is there a fraud?

No →

Is the device secured?

Yes →

Is the firmware valid?

Yes →

Is there an upgrade request (is the key pressed)?

No →

Execute the existing firmware

Yes →

Load secure firmware

No →

Personalize the chip
STM32 Secure Loader Architecture

STM32xx CMSIS Library
Thank you!