Platform Level Security For IoT Devices

Bob Waskiewicz – Applications Engineer
Example of a Simple IoT Device
Security Qualities

- **Authenticity**
  - Unique and Immutable Identity
  - Prevent Counterfeiting/cloning
  - Protect Certificates

- **Data Confidentiality**
  - Protect Keys
  - Protect Credentials
  - Protect customer data

- **Firmware Integrity**
  - Code Integrity
  - Isolate Secure and Non Secure Processes
  - Secure Communications

- **Device Integrity**
  - Tamper Prevention
Security Framework

- Fortified
  - Disable All Debug Ports
  - Secure Firmware Update
  - Secure Boot
  - Secure Provisioning

- Trustworthiness
  - Authenticity
  - Data Confidentiality
  - Firmware Integrity
  - Device Integrity

- Security Framework Components
  - Tamper Detection
  - Crypto Hardware
  - Trusted / Certified Libraries
  - Memory Segmentation / Protection

- Technology Tour 2017
Exploring the STM32L4 Security Features
# STM32 Family vs. Security Features

<table>
<thead>
<tr>
<th>ST Family</th>
<th>Security Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32 F1</td>
<td>Debug Access Port</td>
</tr>
<tr>
<td>STM32 F3</td>
<td></td>
</tr>
<tr>
<td>STM32 F0</td>
<td></td>
</tr>
<tr>
<td>STM32 L1</td>
<td></td>
</tr>
<tr>
<td>STM32 F2</td>
<td></td>
</tr>
<tr>
<td>STM32 F4</td>
<td></td>
</tr>
<tr>
<td>STM32 F7</td>
<td></td>
</tr>
<tr>
<td>STM32 L0</td>
<td></td>
</tr>
<tr>
<td>STM32 L4</td>
<td></td>
</tr>
</tbody>
</table>

Application Note# (AN)/User Manual# (UM) ([www.st.com/mcu](http://www.st.com/mcu)) ([*infocenter.arm.com*](http://*infocenter.arm.com*))
**STM32L476 MCU**

- **Processor**
  - 80 MHz ARM® Cortex® M4

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

- **Connectivity**
  - I2C, SPI, USB

- **Analog**

---

<table>
<thead>
<tr>
<th>System</th>
<th>ART Accelerator™</th>
<th>1-Byte Flash memory with ECC Dual bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power supply internal regulator POR/PDR/PVD/BOR</td>
<td>128-Kbyte SRAM HW parity checking</td>
<td></td>
</tr>
<tr>
<td>Xtal oscillators 32 kHz + 4 to 48 MHz</td>
<td>128-byte backup data</td>
<td></td>
</tr>
<tr>
<td>Internal RC oscillators 32 kHz + 16 MHz</td>
<td>1x Quad SPI</td>
<td></td>
</tr>
<tr>
<td>Internal multispeed ULP RC oscillator 100 kHz to 48 MHz</td>
<td>Boot ROM</td>
<td></td>
</tr>
<tr>
<td>3x PLL</td>
<td>FSMC (SRAM/NOR/NAND)</td>
<td></td>
</tr>
<tr>
<td>Clock control</td>
<td>Floating point unit (FPU)</td>
<td></td>
</tr>
<tr>
<td>RTC/AWU</td>
<td>Nested Vector Interrupt Controller (NVIC)</td>
<td></td>
</tr>
<tr>
<td>SysTick timer</td>
<td>JTAG/SWD/ETM</td>
<td></td>
</tr>
<tr>
<td>2x watchdogs (independent and window)</td>
<td>Memory Protection Unit (MPU)</td>
<td></td>
</tr>
<tr>
<td>51/57/58/52/109/114 I/Os</td>
<td>2x 16-bit advanced motor control timers</td>
<td></td>
</tr>
<tr>
<td>Cyclic Redundancy Check (CRC)</td>
<td>2x ultra-low-power timers</td>
<td></td>
</tr>
<tr>
<td>Voltage scaling 2 modes</td>
<td>7x 16-bit timers</td>
<td></td>
</tr>
<tr>
<td>Touch-sensing</td>
<td>2x 32-bit timers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control</th>
<th>Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x 16-bit advanced motor control timers</td>
<td>3x SPI</td>
</tr>
<tr>
<td>2x ultra-low-power timers</td>
<td>3x PC</td>
</tr>
<tr>
<td>7x 16-bit timers</td>
<td>2x SCI</td>
</tr>
<tr>
<td>2x 32-bit timers</td>
<td>1x SDIO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Display</th>
<th>Technology Tour 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD driver (8x40)</td>
<td>1x USB OTG FS</td>
</tr>
</tbody>
</table>

Internal Memory

- 2x 12-bit DAC
- 24 channels / 5 MSPS
- 2x Op-Amps
- 2x ultra-low-power comparators
- Temperature sensor

---

*STM augmented*
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
  - Debug and Trace
  - Memory Protection Unit (MPU)
Memory Segmentation and Protection

- Fortified
  - Disable All Debug Ports
  - Memory Segmentation / Protection
  - Secure Firmware Update
  - Authenticity
  - Data Confidentiality
  - Firmware Integrity
  - Device Integrity
  - Secure Boot
  - Secure Provisioning
  - Tamper Detection
  - Crypto Hardware
  - Trusted / Certified Libraries
  - Memory Segmentation / Protection

- Trustworthiness
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 errors 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
STM32 Firewall

- Creates a specific “trust area” of code with own memory isolated from all other code areas
- Has a single call-gate interface to enter the Firewall. Any access other than the prescribed call-gate interface results in a RESET
- Ideal for protecting algorithmic IP separate from the rest of the internal application, and performing security sensitive operations (i.e. Hashing)
- Configured at Start and remains active until the next RESET
- Intrusive detection into a protected area generates a RESET

Includes DMA and / or Interrupt intrusions
STM32 Memory Features

- **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
  - BSDL is disabled
  - **Level 2 can NOT be reversed**
STM32 Memory Features

Readout Protection (RDP)

Level 0

Level 1

Level 2

Debug tools / Boot from SRAM / Boot from system memory

* Only on STM32L4
** Only on STM32L0

** Only on STM32L0
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><strong>FLASH Memory</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<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 from FLASH

• **PCROP**: RAM not protected

• **PCROP**: Set at POR via option byte configuration (temporization time), static protection of executable code AND constants (KEYS), no initialization necessary

• **Firewall**: Dynamic execution and storage 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 the STM32L4 and newer devices the WRP area is defined by “start” and “end” addresses

• In the STM32L4 devices, SRAM2 section is write protectable, (as well as RDP protectable)
STM32 Memory Features

Robust Memory Integrity and Safety

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

- Programming granularity is 64 bits (really 72 bits incl. 8-bit ECC)
Cryptographic Hardware

Fortified

Disable All Debug Ports

Secure Firmware Update

Secure Boot

Secure Provisioning

Authentication

Data Confidentiality

Firmware Integrity

Device Integrity

Trustworthiness

Tamper Detection

Crypto Hardware

Trusted / Certified Libraries

Memory Segmentation / Protection
STM32 96-bit Unique ID

- 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

- 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 AES Engine

- NIST FIPS 197 compliant implementation
STM32 Hash Processor

• Hardware acceleration that transforms original plaintext to an unreadable Message Digest

• Compliant with:
  • FIPS Pub 180-2
  • Secure Hash Standards (SHA-1*, SHA-224, SHA-256)
  • IETF RFC 1321 (MD5*)

• Supports
  • Supports data swapping
  • Supports DMA
Random Number Generator

• Random Number Generators are typical usage:
  • to generate challenge, seed in secrets, digital signatures

• Statistical properties of the RNG’s output is used to evaluate level of randomness

• Non-random or predictable output are a source weakness

• A Pseudo Random Generator is a mathematical algorithms completely computer-generated

• A True Random Generator makes use of the unpredictable physical properties of silicon to generate numbers

http://boallen.com/random-numbers.html
http://en.wikipedia.org/wiki/Random_number_generator_attack
STM32 True Random Number Generator

- 32-bit Random Number Generator based on a noise source
  - 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 Crypto Library

Software ONLY

- STM32 Firmware Crypto Library V3.1.0
  - All algorithms are based on firmware implementation without using any hardware acceleration

- STM32 Hardware Acceleration Crypto Library V3.1.0
  - Support the algorithms based on firmware implementation with hardware acceleration (Hybrid)

- The STM32 Crypto Libraries is distributed by ST as an object code library, accessed by the user application through an API
STM32 Crypto Library
CAVP FIPS Certified

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
Enable All Debug Ports
Memory Segmentation / Protection
Fortified
Authenticity
Data Confidentiality
Firmware Integrity
Device Integrity
Secure Firmware Update
Secure Boot
Secure Provisioning
Secure Provisioning
Trusted / Certified Libraries
Memory Segmentation / Protection
Tamper Detection
Crypto Hardware
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

No external components are needed due to internal filter and power monitoring.

- External RESET
- NRST
- VDD
- GP
- Filter
- PULSE GENERATOR (min 20µs)
- SYSTEM RESET
- WWDG RESET
- IWDG RESET
- Firewall RESET
- Option byte loading RESET
- Software RESET
- BOR RESET
- Low power management RESET
• **Backup Domain Contains**
  - A Calendar Real-Time-Clock
  - xx Data Bytes, Back-up SRAM
  - Separate 32k Hz oscillator for RTC

• **Tamper Detection Pins**
  - RESETs all RTC backup registers and Back-up SRAM
  - Time Stamp Event
Anti Tamper

• 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 Band-Gap voltage (supply voltage measurement)
Debug Port Protection

- Fortified
  - Disable All Debug Ports
  - Secure Firmware Update
    - Secure Boot
    - Secure Provisioning
      - Authenticity
      - Data Confidentiality
      - Firmware Integrity
      - Device Integrity
      - Trustworthiness
        - Tamper Detection
        - Crypto Hardware
        - Trusted / Certified Libraries
        - Memory Segmentation / Protection
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

More pins available for the application
Debug Ports

- Debug ports, like JTAG, represent any easy access route for hackers
- Remove test headers, debug access points from the board
- Remove from code any debug processes as these could be exploited
  - Potential Boot loader paths (UART, SPI, I2C, USB, etc)
- Restricted access to production keys
  - Only use test or dummy keys while the product is in debug mode

**NOTE**
- If the device has a lockable debug port – don’t forget to **lock it!**
Secure Boot

- Fortified
  - Disable All Debug Ports
  - Secure Firmware Update
- Secure Boot
- Secure Provisioning
- Authenticity
- Data Confidentiality
- Firmware Integrity
- Device Integrity
- Trustworthiness
  - Tamper Detection
  - Crypto Hardware
  - Trusted / Certified Libraries
  - Memory Segmentation / Protection

Secure Firmware Update

- Fortified
  - Secure Boot
  - Secure Provisioning
- Authenticity
- Data Confidentiality
- Firmware Integrity
- Device Integrity
- Trustworthiness
  - Tamper Detection
  - Crypto Hardware
  - Trusted / Certified Libraries
  - Memory Segmentation / Protection

Secure Boot
Classes of Attacks

IoT Device

Internet

Cloud Service

Software Attack

Misuse of network protocols
- Exploit communication protocol errors
- Flaws in software design / implementation

Board Level Attack

With the case opened / removed
- Test / debug port access
- Inter device bus and IO probing
- Reset, clock attacks
- Power analysis
- Temperature / electrical attacks (glitch, overvoltage)

Silicon Level Attack

Device de-packaged
- Circuit analysis and probing

Fault injection
- Laser beam
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)

- When Boot mode → User FLASH memory and **BFB2** option bit ENABLED -
  - Boot is done from FLASH memory Bank 1 or Bank 2

<table>
<thead>
<tr>
<th>Boot Mode Selection</th>
<th>Boot Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT1 / nBOOT1 (*option bit)</td>
<td>BOOT0 (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>
## Boot Loader

### 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 |  
              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 |
| **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
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.
Secure Boot Flow

- Secure Boot State Machine
- Configure WRP/RDP/PCROP
- Disable DMAs
- Enable MPU
- Enable Firewall
- Enter Protected Enclave
- Zero's SRAM
- Exit Protected Enclave
- Disable IRqs
- Enter Firewall
- Exit Firewall
- Enable IRqs
- Enter Firewall
- Secure Processing
- Disable IRQs
- Exit Protected Enclave
- Disable MPU
- Jump to User App
- Firewall Protection
- Secured by the Memory Protection Unit
Secure Processing

- Secure Processing Environment
  - Secure cryptographic functions
  - Protected access to Shared Info. data-space

- Single access point – Call Gate
  - Code segment cannot be executed without passing through it
  - Volatile and Non-Volatile Data cannot be accessed from outside

- Environment Configured for Protection of -
  - Volatile data (SRAM)
  - Non-volatile data (FLASH)
  - Code - no jump calls
Secure Engine APIs

- Wrapper of internal protected functions
  - Encryption / Decryption
  - Shared Info. - Read / Write

- User-level APIs handling
  - Parameter parsing for the single Secure Engine
  - Entering and exiting of the secure environment
Secure Firmware Update

Fortified

Disable All Debug Ports

Secure Firmware Update

Secure Boot

Secure Provisioning

Authenticity

Data Confidentiality

Firmware Integrity

Device Integrity

Trustworthiness

Tamper Detection

Crypto Hardware

Trusted / Certified Libraries

Memory Segmentation / Protection
SFU / SB Main Features

- Integrates the security features available in the Current generation of STM32 MCUs
- Support Symmetric and Asymmetric keys for encryption and verification of Firmware (AES / ECC)
- X-Cube CryptoLib supported
- VirtualCOM interface for Firmware download
- X-Cube project: Cube compliance
SFU-En Architecture

Hardware Abstraction Layer API

Secure Boot / Secure Firmware Update

Core
- sfu_app
- sfu_boot

Crypto
- AES
- CCM

STM32 CryptoLib
- AES
- CCM

MCU
- sfu_stm32_hal
- sfu_stm32xx_hal
- sfu_stm32_def

Drivers
- STM324xx
- STM32f4xx
- STM32l0xx
- STM32f0xx

Utilities

Utilities

CMSIS

Boards Support Packages

STM32 Nucleo Board
- CommIF_HW_X
- CommIF_HW_Y

X-NUCLEO Expansion Board (CommIF,)

Development Boards

Apps

Middleware
MCUs Architecture Common Approach

• Benefits
  • Maximize size optimization considering different PCROP and WRP granularity options;
  • Combine contiguous regions using the same protections in order to save WRP / MPU / PCROP regions for customer use
  • Secure Boot is the Root of Trust and it is the only responsible for signature verification and decryption, unless FLASH granularity doesn’t allow it
  • Key access and critical operations are embedded inside a Secure Engine (under Firewall), part of the Secure Boot
  • Secure Engine APIs can be shared with UserApp as a module

• Principle / Requirements
  • When Common Approach limits the security, the security level maximization has the highest priority
  • Minimize Code Size, FLASH granularity is MCU dependent
Production Flow With Personalization

Microcontroller

Fabrication

Diffusion and Test  Package

CM / OEM

Factory Personalization

- Put Secret Keys
- Store Certificates

Hardware Security Module

Device Key-Pair

MCU Code

Products

CUSTOMER
Secure Provisioning
Signed Firmware → One-Way Verification

Backroom Tools

1. Host Public Key

2. Hash

3. Sign

4. Application Firmware Package

5. IAP verifies FW Image w/signature

Application FW Image

IAP Code

Host Public Key

Application FW Image

FW Signature

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
4. FW+Dev ID Signature
5. Host verifies FW + Dev ID signature
Helping to Fortify Your IoT Solutions
ST the Strong Element!
Demos
ST Solution for Security in IoT

Solution for IoT Device

Secure Boot
Encryption
Authentication

Secure MCU
With STM32 Nucleo and sensors Expansion Board

Service Provider

CLOUD
Thank you!