Crypto-Bootloader – Secure in-field firmware updates for ultra-low power MCUs

Oscar Guillen
PhD Student, MCU MSP Europe Design

Bhargavi Nisarga
MCU MSP Systems Engineer

Luis Reynoso
MCU MSP Applications Engineer

Ralf Brederlow
DMTS, Kilby Labs

Texas Instruments

Abstract

In this work we present Crypto-Bootloader, a custom bootloader implemented in MSP FRAM MCUs for secure in-field firmware updates. Being increasingly used in MCU-based applications today, in-field firmware updates will be crucial in the oncoming Internet-of-Things (IoT). The updates enable new firmware images to be downloaded into the MCU's memory, and provide an effective way for product manufacturers to offer service and support to products already deployed in the field. However, if proper security measures are not in place, this feature may also be misused. In-field firmware updates are one of the first targets for attackers looking to compromise the security of a system. The consequences of successful exploitation of an embedded system through insecure in-field firmware update mechanisms can be disastrous—ranging from loss of intellectual property and product cloning, all the way to complete control of the deployed system. In this work, we address the security issues and respective measures for implementing a secure in-field firmware update process. This includes a holistic solution formed by cryptographic algorithms and security mechanisms in the protocol and bootloader implementation. We present the results of the implementation in a low-cost, ultra-low-energy, general-purpose MCU. The implementation of Crypto-Bootloader in an MSP430FRx MCU takes 3.2 KB of code and less than 1 KB of data space, and takes approximately 56 thousand cycles to decrypt, verify and program a 256-Byte packet.

I. Introduction

Supporting in-field firmware updates is an important and essential feature in today's products. Firmware updates to products that are deployed in the field offer benefits to both the product manufacturer and the end user. Benefits include: providing the ability to remotely add new features and functionalities to products that are already deployed in the field, fixing firmware bugs after a product has been released. For the product manufacturer, this feature helps reduce the number of product returns and for end users, it enables a more positive experience with the product.

The in-field firmware update process includes the following steps: new firmware image generation at the product manufacturer's end, transferring the firmware image from the product manufacturer's site to the end-product's site, and finally, loading the new firmware image into the device within the product. The new firmware image may have to be transferred and/or loaded in a non-secure environment and therefore, requires necessary security measures to ensure security of both the firmware image and the product operation itself. In-field firmware updates involving MCUs are enabled by the bootloader on the device. A bootloader is a piece of code that resides in the device's memory and has the ability to reprogram the application memory space of the device. On-chip communication modules, such as UART, I^2C, SPI or USB, are used for interfacing the bootloader to a firmware update tool, or to a host processor performing the firmware update.

Security in in-field firmware updates is critical as this feature, if misused, enables attackers to gain access to the firmware image being updated or enables attackers to manipulate the device operation. This

paper presents the Crypto-Bootloader solution for MSP MCUs that implements security measures to elevate overall security in in-field firmware updates. The following sections in the paper discuss the need, benefits and security features of the Crypto-Bootloader solution. It is structured as follows: Section II discusses remote firmware updates of network-connected MCUs in an IoT framework and the various security considerations that need to be addressed. Section III covers the security measures supported by the Crypto-Bootloader solution and the implementation level of Crypto-Bootloader on ultra-low-power MSP430FR5969 MCUs with embedded FRAM technology. Finally, Section IV summarizes the security features offered by the Crypto-Bootloader solution including code size and performance metrics.

II. In-field firmware updates in network-connected MCUs

A. Network-connected MCUs

Network-connected MCUs are becoming increasingly popular with the emergence of the IoT. Network connectivity in an IoT framework enables embedded systems to be part of a broader grid that handles unprecedented amounts of data. Figure 1 shows an example representation of a network framework. It comprises end nodes (1) with MCU devices that interface to the "Things" in the IoT, including sensors that capture the sensor information (e.g., temperature, pressure, humidity, etc.), and actuators that provide a means to act on the environment. The end nodes are connected to a local area network (LAN) (2) and communicate with a LAN gateway controller (3) that acts as a data concentrator which handles information to be transmitted/received from all the end nodes in that particular LAN network. Here, LAN also refers to smaller area networks including Home Area Network (HAN). The end nodes incorporate the required physical interface (PHY) to connect to the LAN network (e.g., Ethernet for wired LAN connection, Wi-Fi®, *Bluetooth*®, etc. for wireless LAN connection). Figure 1 shows two types of end-node implementations—one with LAN PHY interface integrated within the MCU (End Node #1) and the other with an MCU connected to an external LAN PHY interface chip (End Node #2). The LAN gateway controller on the other side connects to the wide area network (WAN) (4) and handles the required protocol to interface to the WAN network (e.g., Internet Protocol). The utility manufacturer

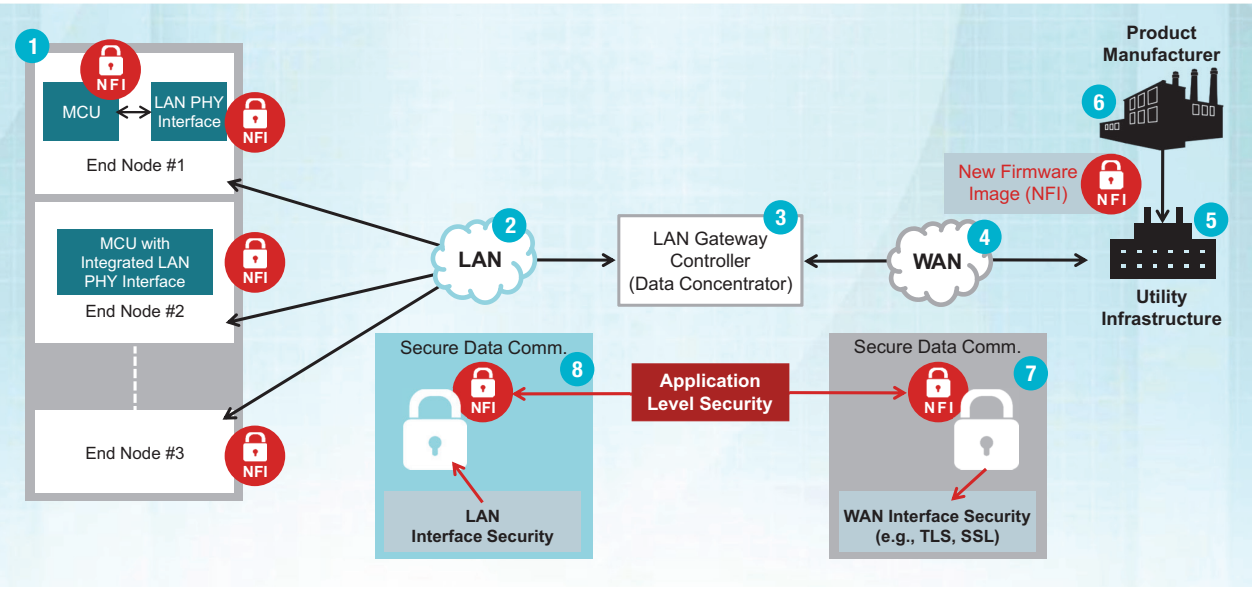

Figure 1: Example representation of network-connected MCUs in-field firmware updates

(5) uses the WAN network in order to connect to or access the end nodes in the network.

For in-field firmware updates, the network connectivity framework discussed above can be used to effectively distribute new firmware images to the network-connected MCUs. In this process, the product manufacturer generates the new firmware image and sends it to the utility infrastructure for distribution via network (6). As part of the network security, the LAN and WAN interfaces incorporate necessary security measures that are dictated by the interface protocols used in the network connectivity, shown as (7) and (8) in Figure 1. However, the network security at both LAN and WAN network levels are popular points of attacks and incidentally, there have been multiple attacks on the network security layers at both LAN (Ryan, 2013) (Vidgren, 2013) and WAN interface levels (Durumeric, 2014) (Al Fardan, 2013) (Rizzo, 2012).

This paper focuses on security measures applicable to in-field firmware updates at an application level that adds an additional layer of security to the new firmware image that is handled in the network. In other words, the new firmware image that needs to be transferred from the product manufacturer (6) to the MCUs in end nodes (1) over the network is secured at an application level, even before it enters the network. This offers many advantages:

a. The security of the new firmware image in the network does not depend on the security of the network.

b. It provides increased security to the new firmware image at the LAN gateway controller node (3) when switching between LAN and WAN network security protocols.

c. It offers an increased level of in-field firmware update security at the various nodes in the network (1), (3) and (5). For example, if an attacker has access to the end node, then this application level security ensures firmware image information is not readily accessible until the MCU application layer retrieves it.

B. Security considerations in in-field firmware updates

In order to determine application-level security measures required for in-field firmware updates, it is important to understand the security assets that need protection, the security threats that are of concern and the security boundary definition in the in-field firmware update process.

The binary image or the firmware image distributed in field that is to be downloaded onto the MCU within the end node corresponds to the “Intellectual Property” of the product manufacturer and therefore, the main asset that needs protection. Firmware images may include code, data, calibration values, authentication secrets and other intellectual property.

A security threat consists of a threat agent, an asset and an adverse action of that threat agent on that asset. And, when executed, the threat possibly compromises the security of the asset. This paper considers the following security threats in in-field firmware updates:

1. **Firmware reverse engineering:** Reverse engineering the firmware image (binary code) into assembly or a higher level language in order to analyze the functionality and contents of firmware image.
2. **Firmware alteration:** Partial modification to the firmware image distributed by the product manufacturer.
3. **Loading unauthorized firmware:** Loading an unauthorized firmware image into a device. The unauthorized firmware image may correspond to code created by an unauthorized party or firmware not intended for the specific device.
4. **Loading firmware onto an unauthorized device:** Loading the firmware image generated by the product manufacturer into a device which is not authorized.

Other threats to the system involve making the device or end node unavailable for service by interrupting the firmware update process (e.g.,

interrupt the firmware update process such that firmware is only partially updated on the device and device does not start application firmware execution as integrity of the firmware on-chip is compromised). Security measures to address these security threats are discussed in Section III.A below.

The scope of this paper assumes that the firmware image, once programmed within the MCU, is secure and therefore, software or hardware attacks on the device itself are not discussed here. Also, it is assumed that the bootloader code on the device is always functional and cannot be altered or modified.

III. Crypto-Bootloader

A. Security measures

The objective of the Crypto-Bootloader is to increase the security of firmware updates for devices in the field. To enable this, a combination of cryptographic algorithms and protocol-level measures is required. Cryptographic algorithms provide the means to protect the privacy of the content and to verify its integrity and authenticity. Protocol-level measures are employed to ensure that the version of the firmware update is correct and to protect the device from executing incomplete code.

1. Cryptographic measures

Cryptographic algorithms may be symmetric or asymmetric depending on the type of keys used for encryption and decryption. Symmetric algorithms require that the same key used during the encryption process is also used during decryption in order to retrieve the original plaintext. On the other hand, asymmetric algorithms make use of public and private keys for this means. Encryption is performed by the sender using the public key of the recipient and decryption is done using the recipient's private key. In this way, the private keys never have to be exchanged. While this offers keys distribution advantages over symmetric algorithms, asymmetric algorithms are not suited for ultra-low-power applications since they are computationally intensive, meaning that they require long computation times which in turn cause high energy consumption.

In the case of the Crypto-Bootloader we have opted for using a pure symmetric cryptography approach for the security. The secret key required to establish communication when the device is in the field is programmed when loading the original firmware image onto the device and a key update mechanism is provided in order to modify the secret key once the device is deployed in the field (as shown in Section III.B). The main building block is the Advanced Encryption Standard (AES) (NIST, 2001).

Crypto-Bootloader provides a high level of security by making use of Authenticated Encryption (AE). While encryption provides the secrecy required to maintain the payload confidentiality, it does not protect against malicious modifications. In order to verify the integrity and authenticity of the encrypted data, other cryptographic algorithms are needed. For symmetric cryptography, integrity and authentication are provided through the use of Message Authentication Codes (MAC). Encryption-only modes and MACs may be combined to enhance the security of the encrypted data. However the combination of these algorithms is not completely straightforward as naïve combinations may open the door to other types of attacks (Bellare, 2000). For this reason we chose an AE mode; specifically the Counter with CBC MAC (CCM) AE mode (Whiting, Housley, & Ferguson, 2002), which forms part of the recommended National Institute of Standards and Technology (NIST) modes of operation (Dworkin, 2004). AES CCM is used by the Crypto-Bootloader for both firmware and key updates.

2. Protocol-level measures

In addition to the security properties which are provided by the cryptographic algorithms, protocol-

level measures were added to prevent other attack vectors which cannot be covered by them. For firmware updates, measures were added to prevent downgrading and to ensure that the complete number of packets has been received. Whereas for key updates, preventing use of old keys and making sure that the correct keys are used for the right algorithm is important.

Firmware downgrading is a potential attack scenario if more than one firmware image has been encrypted using the same valid key. An attacker in possession of an old encrypted firmware image may resend it to the device reverting it to a previous, possibly vulnerable, state in order to exploit it. Firmware downgrading has been exploited to reduce the security of embedded devices, mostly in cell phones and gaming consoles (Pandya, 2008) (DeBusschere & McCambridge, 2012) (Ruan, 2014).

Similar to firmware downgrading, attacks may involve reverting keys to old values when more than one firmware image has been encrypted with the same key. To counter this, a version field is included in the protocol to ensure only newer versions of the keys are programmed during an update. Additionally, different types of keys must be used for different algorithms (Barker, 2006). Therefore, the protocol also includes a field to correctly identify the type of key being updated. More details on the different key types are given in Section III.B

Another potential threat is an attacker interrupting a firmware update process such that incomplete firmware image is executed on the device. To ensure that the complete firmware image has been received by the device before executing the new firmware image, the best approach is to store the incoming firmware image in a buffer located in non-volatile memory and only accept the update after it has been verified that the complete image has been received. However, this approach requires that both the original and the new updated image be stored on the microcontroller at the same time; thus, increasing the on-chip memory requirements up to double the original firmware size. And this, in turn, increases the overall MCU cost. The solution implemented by the Crypto-Bootloader to address this takes an alternative approach wherein packet fields at the protocol level are used to determine the number of packets which are expected to be received and only allows execution of the code once the complete update is obtained (i.e., the device remains in bootloader mode until all the expected packets have been received, decrypted, verified and written to the device memory). This approach may not be suitable for high-reliability applications as any interruption during an update can perturb the device or product operation and this may not be acceptable in the application. Applications with such high-reliability requirements must therefore opt for the former approach; however, for applications that are not sensitive to interruptions during a firmware update process and that can wait until the firmware update process completes, the latter approach would be a more effective solution. More details on the packet structure and the protocol-level security measures are given in Section III.C.

B. Keys management

The security of a cryptographic algorithm and of the system using it relies on the security of the keys. In symmetric cryptography, the same key must be held by the transmitter and the receiver of the protected information. For Crypto-Bootloader, this means that a copy of the keys used by the cryptographic functions implemented will reside in the device's non-volatile memory (NVM). The initial keys will be programmed onto the device while the device is still in the possession of the manufacturer in a trusted environment, and Crypto-Bootloader supports updating the keys in the field such that they may be updated later when the device is deployed.

Up to three types of keys are needed to be considered depending on their use: encryption keys, authentication keys and key-encryption keys. As their

```
struct cryptobsl_key_st {
  unsigned char key [CIPHER_KEY_SIZE];
  unsigned char version;
};
typedef struct cryptobsl_key_st CRYPTOBSL_KEYS;
```

Table 1: Data structure used for key storage

names imply, encryption and authentication keys are used in their respective algorithms; while Key-Encryption Keys (KEKs) are used only for the purpose of encrypting keys (Barker, 2006). Table 1 shows the data structure used to store these keys within the device.

The data structure for the keys includes the key itself as well as its version number. This metadata is used during an update to verify that the key received is in fact newer than the one currently stored; preventing keys from being reverted to old values, which could potentially weaken the security of the system.

To support key updates in the field, packets make use of authenticated encryption to preserve the secrecy of the key material and to prevent an attacker from tampering with it. Additionally protocol-level fields are used to verify that the version of the key being received is newer than the one already stored in the device and to distinguish which type of key is being used.

C. Packet format

Crypto-Bootloader uses a specific format to create the data packets that compose the encrypted firmware update. The structure of each individual data packet is as shown in Table 2 and consists of three different field types: 1) firmware fields, 2) cryptographic fields and 3) protocol-level security fields. Firmware fields correspond to fields that contain the basic firmware update material, that is, the binary data and the on-chip memory address where this data is to be stored. Cryptographic fields include information that corresponds to the cryptographic algorithms used for integrity and authenticity verification and decryption; these include the Initialization Vector (IV) and the MAC tag. Lastly, the protocol-level security fields include firmware version, packet number, number of packets and a field reserved for future use. The version of the firmware update is compared with the one of the firmware within the device to prevent it from being downgraded to an older version, even if the same valid key was used to encrypt both images. Packet number and number of packets fields help ensure that the complete image has been received. Crypto-Bootloader will prevent the updated code from being executed until it can confirm that the complete new firmware image has been received.

The packet format for key updates is slightly different, and is as shown in Table 3 on the following page. In this case, there are two different field types: 4) key fields and 5) cryptographic fields. The key fields include the key type, key version and the new key itself. The key type is a byte used to distinguish the key material according to its functionality. Typically, the possible key types are data-encryption key, data-authentication key and key-encryption key. Since Crypto-Bootloader uses AES-CCM for authenticated encryption, the same key is used for authentication and encryption. Therefore, only two types of keys are used: authenticated-encryption key and key-encryption key. The key version byte is used to prevent downgrading a key with an old value,

Table 2. Encrypted data packet

IV[2)]	Version[3)]	Packet #[3)]	#Packets[3)]	Reserved	Address[1)]	Data[1)]	MAC[2)]
IV	VER	PN	NP	RSV	AL, AM, AH	D1..Dn	TAG

Table 3. Encrypted key packet

IV[5]	Key type[4]	Key version[4]	Key[4]	MAC[5]
IV	KT	KV	KEY	TAG

even if the key is encrypted using the current key-encryption key. The key field contains the new key material. The cryptographic fields include the IV and MAC tag which are used for the underlying cryptographic algorithm.

D. Implementation details

The Crypto-Bootloader solution has been implemented on the Texas Instruments' MSP430FR5969 MCU. This device belongs to the new family of ultra-low-power MSP430FRx FRAM MCUs. Non-volatile FRAM technology enables faster write operations, practically unlimited endurance and lower power consumption. The MSP430FR5969 MCU features a Memory Protection Unit (MPU) and an AES-256 cryptographic hardware accelerator. The following describes the specific aspects of the implementation of Crypto-Bootloader.

1. Bootloader invoke options

Upon device reset, the MCU begins execution of the Crypto-Bootloader and a low-level initialization routine decides whether to stay in bootloader mode, or to execute the user application. In order to provide flexibility and reliability, Crypto-Bootloader stays in bootloader mode in any of the following conditions:

- An external invoke sequence detected after reset. This sequence can be as simple as a GPIO being held low; or, a sequence of events providing backwards compatibility with generic versions of MSP MCU bootloaders.
- A previous firmware update session that was interrupted and not successfully completed.
- The application requested execution of the bootloader.
- The application reset vector is blank.

2. External access

Allowing device access through JTAG or other non-secure bootloader methods would defeat the purpose of the Crypto-Bootloader. The MSP430FR5969 MCU can prevent device access via JTAG and the device's default bootloader (ROM BSL) by using respective signature keys within the main memory of the device. When Crypto-Bootloader is programmed into the device, it configures these keys accordingly to lock the JTAG access and disable the default device bootloader. If it is needed to unlock the device JTAG access later on, the Crypto-Bootloader provides an option to do so using packets that have been encrypted and can be authenticated.

3. Memory protection

Crypto-Bootloader makes use of the MPU to prevent unintended access to the bootloader area. A low-level initialization routine implemented within the Crypto-Bootloader configures the MPU appropriately during bootloader execution and when jumping to the application, preventing corruption of this memory area.

In addition to the MPU, Crypto-Bootloader uses the MPU-IP Encapsulation (IPE) to define a memory segment that restricts read/write access to the data within. Once enabled, the device cannot read code/data within the IPE segment and only code executed inside the IPE segment can read and modify data within the segment. The complete Crypto-Bootloader code, including the cryptographic keys is placed within this IPE-protected segment.

4. Communication interface and protocol

Crypto-Bootloader supports bootloader communication via UART and I^2C. However, only one of the interfaces can be used at a given time. The source code is written in a modular way allowing developers to change or customize the communication interface.

The protocol used by Crypto-Bootloader is an extension of the standard BSL protocol used for MSP MCUs. Optional backward compatibility can be enabled based on the contents of a reserved non-volatile location, or it can be completely disabled through pre-compiler definitions.

5. *FRAM advantages*

As mentioned previously, Crypto-Bootloader is implemented in an MCU with embedded FRAM. Advantages of FRAM in the Crypto-Bootloader solution include:

- **Erase/Write granularity:** allows partial updates of the code (as small as a single byte) without having to erase big segments of memory and allows for an easy update of the interrupt vectors without requiring vector redirection.
- **Non-volatile:** allows for an easy implementation and update of persistent variables such as the application reset vector or the encryption keys.
- **Speed:** being comparable to SRAM and significantly faster than Flash or EEPROM, FRAM allows for faster firmware updates.
- **Endurance:** practically removes this concern from the mind of programmers with guaranteed minimum 10^{15} write or erase cycles, compared to a typical flash endurance of 10^5 cycles.

IV. Summary

The Crypto-Bootloader solution presented in this paper provides a flexible and effective solution for increased security in in-field updates which can be implemented in low-power MCUs for IoT applications. Crypto-Bootloader provides an effective way for product manufacturers to offer service and support to products already deployed in the field. The solution comprises the use of standardized symmetric cryptographic algorithms in addition to protocol-level security mechanisms in order to counter the most important threats to in-field update mechanisms.

The solution has been implemented and tested on the MSP430FRxx FRAM MCU using the following configuration:

- **Bootloader communication interface:** enhanced Universal Serial Communication Interface (eUSCI) UART and I^2C
- **Device MPU and MPU-IPE:** enabled
- **Authenticated encryption method:** AES-CCM
- **AES engine:** AES256 hardware module

The code footprint of Crypto-Bootloader using the above configuration uses 3.27 KB of code and constants stored in FRAM non-volatile memory and 908 Bytes of volatile memory using RAM.

Decryption, verification and programming time of a 256-Byte packet at 8 MHz took approximately 7.03ms or 56,000 cycles.

The throughput of a firmware update will depend on other external factors including the data rates of the communication interface used, latency of the host programmer and size of packets. For benchmark purposes, an image of 8 KB sent to the device by the BSL Scripter tool described in Appendix A , using a UART at 115,200 bps was programmed in 1.49 seconds which is equivalent to 5.35 KB/sec.

V. Bibliography

- Al Fardan, N. a. (2013). Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. *Security and Privacy (SP), 2013 IEEE Symposium on* (pp. 526–540). IEEE.
- Barker, E. a. (2006). Recommendation for key management-part 1: General. *NIST special publication*.
- Bellare, M. a. (2000). Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in*

Cryptology—ASIACRYPT 2000 (pp. 531–545). Kyoto, Japan: Springer.

- DeBusschere, E., & McCambridge, M. (2012). Modern Game Console Exploitation.
- Durumeric, Z. a. (2014). The Matter of Heartbleed. *Proceedings of the 2014 Conference on Internet Measurement Conference* (pp. 475–488). Vancouver, BC, Canada: ACM.
- Dworkin, M. (2004). Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. *NIST Special Publication* 800-38C.
- NIST. (2001). FIPS 197: Advanced encryption standard (AES). *Federal Information Processing Standards Publication* 197, 1–51.
- Pandya, V. R. (2008). *iPhone security analysis*. San Jose State University.
- Rizzo, J. a. (2012). The CRIME attack. *EKOparty Security Conference*. Buenos Aires.
- Ruan, X. (2014). Cyber Security in the Mobile Age. *Platform Embedded Security Technology Revealed*, 1–25.
- Ryan, M. (2013). Bluetooth: With Low Energy Comes Low Security. *Proceedings of the 7th USENIX Conference on Offensive Technologies*. Washigton, D.C.: USENIX Association.
- Vidgren, N. a.-A.-S. (2013). Security Threats in ZigBee®-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. *System Sciences (HICSS), 2013 46th Hawaii International Conference on*.
- Whiting, D., Housley, R., & Ferguson, N. (2002). *Submission to NIST: Counter with CBC-MAC (CCM) AES Mode of Operation*. Computer Security Division, Computer Security Resource Center (NIST).

Appendix A Crypto-Bootloader Tools

The **Crypto-Bootloader** solution provides the following tools for in-field firmware update processes:

- **Bootloader Scripter Tool (MSPBSL):** User interface which enables communication with the BSL on MSP microcontrollers to modify the device's memory via UART, I^2C, SPI or USB.
- **Crypto-Bootloader for MSP430FR5x/6x FRAM MCUs:** This bootloader uses cryptographic functions to enable increased security for in-field firmware updates. A Graphical User Interface (GUI) is available for a simplified user-experience.
- **Microcontroller Bootloader Programming Board (MSP-BSL:** hardware interface providing a bridge between the PC's USB port and UART, I^2C or SPI.

The tools usage in the in-field firmware update flow is shown in Figure 3 on the following page.

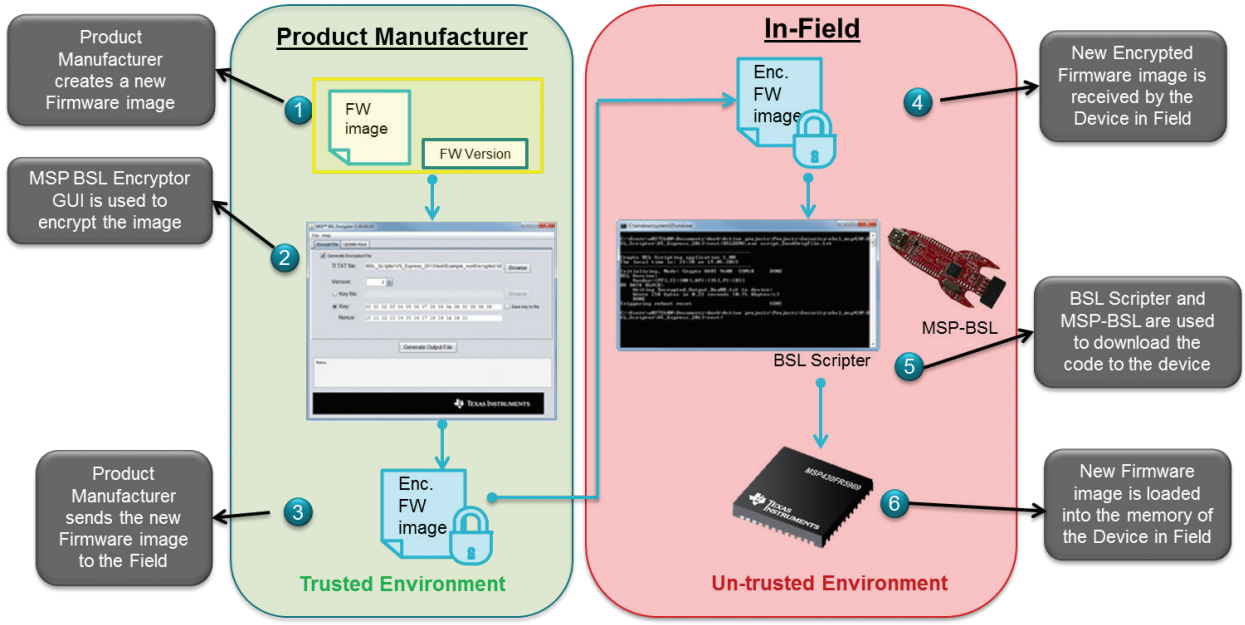

Figure 3: Crypto-Bootloader tools in in-field firmware updates flow

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265