A Novel Holistic Security Framework for In-field Firmware Updates

Sudeendra kumar K, Sauvagya Sahoo, Krishna Kiran, Ayas Kanta Swain, K.K.Mahapatra
kumar.sudeendra@gmail.com, sauvagya.nitrkl@gmail.com, krishna.kiran1217@gmail.com, swain.ayas@gmail.com, kmaha2@gmail.com
National Institute of Technology, Rourkela

***Abstract*- The software/firmware running on the electronic devices is regularly updated. In IoT devices, the updates are performed Over the Air (OTA) through internet. In the absence of proper security measures, OTA update feature can be misused. The security threats like firmware reverse engineering, loading unauthorized firmware and loading authorized firmware on unauthorized nodes will lead to misuse of intellectual property, product cloning and denial of service attack. In this paper, we propose a security framework the microcontroller/SoC devices can incorporate for secure in-field OTA firmware update process. The proposed holistic solution support JTAG security, protecting IP rights of original device manufacturer (ODM) and secure OTA update. The security framework is designed using suitable cryptographic algorithms and protocol measures to address all the security threats connected with OTA firmware/software update which is not addressed in the past techniques.**

Keywords: **OTA Update, JTAG security, Hardware Security.**

I. INTRODUCTION

Security of devices (especially processing elements like microcontrollers and System on Chips (SoC)) is very important and critical in the Internet of Things (IoT). The ODM introduce sensitive assets like cryptographic keys to access IP cores and bootloader program. Original Equipment Manufacturers (OEM) develop electronic product or part of the product using the chips from ODM and introduce their product related trade secrets in firmware and software. The confidential assets OEM introduce into device include cryptographic keys for over the air firmware/software update and digital rights management (DRM) keys etc [1] [2]. ODM, OEM and end-user (customer) are the major stake holders in the supply chain. The ODM introduces confidential assets into device during design or through its custom bootloader. OEM and end-user will use it in their application, but they must not have complete access to it. Similarly assets introduced by OEM must be secure when the device is with customer and ODM (during return material authorization, etc). The online or OTA channel is highly essential to update the systems for the changing application environments and SoC devices must facilitate secure OTA update [3] [4].

Bootloader is primary software runs on processor to mount operating system or any other application. Secure assets and license based modules in the hardware are enabled or disabled through bootloader programs. The generally followed procedure connected with bootloader program is as follows [5] [6]: -

- It is common to have multiple bootloaders in modern day SoC devices. The ODM loads the primary bootloader into the secure memory, which can load the secondary bootloader or application directly. The primary bootloader initialize the device and licensed IP cores. JTAG is used to load the primary bootloader into the secure memory.
- The OEM develops both hardware and software. OEM software may also include secondary bootloader, which initialize the device with the help of primary bootloader and all peripherals placed on the board. Secondary bootloader can be loaded into memory through different communication channels like SPI, etc other than JTAG. Secondary bootloader includes security features to protect the IP rights of the OEM.
- The JTAG access is restricted to ODM and other stake holders in the supply chain will have no access to JTAG which ensure the security of the ODM assets.
- OTA firmware update from OEM to field device is performed in the same communication channel used to load secondary bootloader.

Most of the SoC vendors facilitate the secure OTA updates. In the existing security solutions, passwords are saved in software data structures and do not address JTAG security, safety of bootloaders and OTA update holistically. In this paper, we propose a holistic security model for low end microcontrollers generally used in non OS/RTOS applications. Section II describes the security features available in modern day SoC's and prior work in both academics and industry. In section III, we propose our security design and section IV discuss the implementation. In section V, we discuss Analysis and finally, section VI concludes the paper.

II. PRIOR WORK

We discuss important security implementations in SoC devices connected with secure OTA update in this section.

ARM TrustZone: - ARM TrustZone support secure booting and facilitate secure OTA updates. ARM TrustZone facilities secure update using RSA encryption [6]. The trusted vendor (OEM or application developer) use RSA private key to generate the signature of the software code that they want to deploy. The signature along with code is loaded into flash. Pubic key is stored in every device to verify the binary loaded in the flash. The public key is not confidential and it is same for all devices. Disadvantage of this technique is same public key is stored in every chip and in most of the cases it is part of primary bootloader, which is vulnerable to attacks. ARM recommends creation of OTP (one time programmable)

memory to store the secret key, but it is not a part of ARM TrustZone technology.

X-Cube SBSFU (Secure Boot and Secure Firmware Update): - X-Cube SBSFU is a security solution found in STM32 microcontrollers. Device receives the encrypted binaries and security engine which is part of the device middleware will manage all critical data and operations [7]. SBSFU is a middleware solution and it works well when used with operating system. Passwords/keys are handled at software level which can be accessed through JTAG and vulnerable to different kinds of side channel attacks.

Crypto-bootloader of MSP430: - Crypto-bootloader in Texas Instruments MSP430 microcontroller provides security against unauthorized firmware updates and IP encapsulation [8]. AES-CCM is used for authenticated decryption of the encrypted firmware from vendor and AES-CBC to check the integrity. Keys used in cryptography are stored in device non-volatile memory. The initial keys into device are programmed by ODM in a trusted environment. The primary bootloader program loaded by ODM includes secret keys, which are stored in secure memory. The vulnerabilities in this technique are: - JTAG is not protected and key is same for all devices.

Software based firmware update techniques in automotive applications: - The authors of paper [9] propose a protocol and security architecture for OTA updates and diagnosis for automotive control units. Security architecture is based on hardware security modules (HSM) which perform all cryptographic operations. HSM is designed at application layer level and works well with the operating system (ex: - OSEK). All passwords are stored in memory as data structures which are vulnerable.

Sensitive assets like passwords to enable IP cores, etc are part of primary bootloader which are generally loaded into the SoC device through JTAG interface in the trusted environment. Primary bootloader allow porting of OEM firmware and software through other interfaces like SPI, UART etc. Primary bootloader block the JTAG interface to OEM and field technician. The sensitive assets of ODM can be accessed only through JTAG. JTAG security, IP rights of different stake holders, secure transactions between vendor and IoT edge device during OTA firmware/software update are tightly connected and must be addressed during microcontroller/SoC design holistically. In this paper, we propose a novel security architecture addressing the security vulnerabilities of the above techniques with following features: -

- Completely hardware based key and all other keys are derived from the hardware key. A unique password for every chip manufactured.
- A lightweight security framework which fit in the low end microcontrollers to support IoT security requirements like OTA update and restricted access to JTAG.

III. PROPOSED SCHEME

The main objectives of the proposed secure firmware update are: - Root of trust from hardware is preferred over software. One of the limitation of previous techniques discussed is password/key is stored in memory and access directly through software, which is vulnerable to attacks. The proposed design achieves the objectives by: cryptographic measures for encryption, protocol measures to handle supply chain issues and ensuring secure key management. These three factors must be addressed carefully. The block diagram of the proposed security architecture is shown in fig.1. The proposed framework consists of two sections: JTAG protection section and firmware update section. The JTAG section comprises of conventional JTAG 1149.1 circuit with lock/unlock mechanism. The lock/unlock mechanism consists of True Random Number Generator (TRNG), One-Time Programmable (OTP) memory, comparator and logical AND gate. The functional description of JTAG security mechanism is as follows: -

- The post manufacturing testing of chips is performed on ATE (Automatic Test Equipment). During testing, JTAG is open and ATE will apply test stimulus and capture responses from the device. At the end of testing, TRNG (32-bit) is triggered to generate a random number. Random number is written into OTP. Random number stored in OTP1 and OTP2 is read back into ATE.
- The upper word is stored into OTP2 and lower word into OTP1. OTP1 is used to protect JTAG protection and OTP2 is used for secure firmware update.
- The TAP (Test Access Port) controller state machine (FSM) is modified to introduce another extra state to handle the register in JTAG protection section. JTAG is locked when TRNG writes a random number into OTP1 during testing. The ODM after collecting the OTP data of every chip during testing will store the chip ID and OTP data in a secure database.
- ODM will use JTAG to load primary bootloader and OTP1 is used to unlock the JTAG interface. In the stream of data through TDI pin, OTP1 is loaded into Register1 and compared with data in OTP1 in a comparator. After a successful match, comparator will generate the logic '1' to unlock the "TDO" signal. JTAG can access any section of device and apply test input and collect outputs. Access to JTAG is only restricted to ODM. The primary bootloader is loaded using JTAG interface into secure memory to safeguard the interest of ODM.

The functional description for the secure firmware/application software update is as follows: -

- Secure firmware update section consists of Boot-core, AES-CBC (symmetric cipher), Interface circuit (I2C, SPI etc) and RSA (asymmetric cipher).
- ODM will derive both public key and private key for RSA encryption from the word stored in OTP2. ODM will share the both public and private key with OEM. Key used in AES-CBC, public key for RSA are included in the primary bootloader.
- Interface circuit between the boot-core and Ethernet connection can be any wired protocols like UART, I2C or SPI depending on the flexibility. Interface

circuit is a bridge between boot-core and external world.

- Primary bootloader consists of drivers to initialize all licensed IP cores, boot-core and all peripheral modules of SoC device. This will enable the OEM to load the secondary bootloader or application software without using JTAG.
- In a factory setup, OEM can load AES encrypted secondary bootloader or application software directly into program memory through interface circuit. On-chip AES-CBC decryption module decipher the binaries and bootcore will load the binaries to program memory.
- Once deployed on the field the interface circuit is connected to internet. When device request for update, the random value stored in OTP2 is encrypted in RSA and sent to OEM to identify the IoT node. OEM decrypt random number and decide whether the node require update or not. If device is eligible for update, AES encrypted binaries are transmitted through internet.
- Encrypted binaries are received through interface circuitry and decrypted in AES decryption block. The boot-core will load the decrypted binaries into program memory.
- AES include cipher block chaining (CBC) which is used to check the authenticity and integrity of the firmware/software binaries received over the air.
- The primary bootloader facilitates the rollback option, when new firmware received over the air fail authenticity and integrity tests in AES-CBC.

IV. IMPLEMENTATION

To implement and validate the proposed security framework, we have chosen openMSP430 microcontroller from opencores.org, AES-CBC, RSA and bootcore.

OpenMSP430 Microcontroller: -The openMSP430 is a clone of TI MSP430 microcontroller which is FPGA/ASIC synthesizable openly available in opencores.org [11]. The openMSP430 core available with opencores.org does not have RAM or ROM. In our experiment, we have included Verilog models of RAM and ROM for simulation and verification of the proposed security framework. The figure 2 shows the block diagram of the openMSP430 microcontroller. The memory mapping of the openMSP430 is fully configurable. The openMSP430 also consists of direct memory access (DMA) interface is a gateway to the 64kB memory space. DMA controller supports efficient connection to proposed Boot-core. Generally, boot-core logic is used to initialize the program memory at start-up from external memory interface. In this experiment same channel is used to load the program from external interface. Once the program memory is loaded, the boot-core module generates the reset pulse so that CPU starts fetching the new reset vector from the program memory.

Bootcore: - Bootcore design is based on the flash memory control logic used in the IC AT26DF081A [12]. This is a serial flash memory device used to load the code from external memory to embedded RAM or ROM. The Bootcore is designed using Verilog HDL.

AES-CBC and RSA: - Advanced Encryption Standard- Cipher Block Chaining (AES-CBC) and RSA are ciphers. The secure firmware update section consists of AES-CBC decryption and RSA design is described in Verilog HDL. General shift registers are used in simulation to implement OTP1 and OTP2. Both OTP1 and OPT2 are 16-bits in size.

JTAG/SWD: - The debug infrastructure of openMSP430 deviates from original MSP430 [11]. The custom made openMSP430 debug interface is based on two-wire protocol which uses UART or I2C protocol.

The openMSP430 debug architecture is different from conventional JTAG. In this work, we have gated the signal similar to “TDO”. The openMSP430 serial debug interface support two wire communication bus for remote debugging: - DBG_EN and DBG_FREEZE.

DBG_EN: - This signal is used to enable the serial debug interface without interfering with processor. When disabled (DBG_EN = logic 0) debug interface is held into reset.

DBG_FREEZE: - This signal is set when debug interface stops processor. The purpose of this signal is to freeze a peripheral when the processor is stopped by software debugger.

We have used DBG_EN signal to implement the security to JTAG/SWD in openMSP430. In the place of “TDO” signal in a conventional JTAG, we have used DBG_EN in lock/unlock mechanism.

Verification and Synthesis: - The simulation of whole system is performed in Cadence NC Simulator. The assembly program is compiled in TI MSP430 compiler and generated hex code is loaded into program memory through debug interface. The layered testbench using system Verilog is created to perform simulations. The primary bootloader code designed in assembly language of MSP430 initialize the bootcore and security framework. The bootloader program initializes the bootcore and facilitates the installation of secondary bootloader or application software. The complete system is synthesized using Synopsys Saed90nm library in Design Compiler.

V. ANALYSIS

Protection of JTAG: - An adversary use the JTAG to get an access to secret data once the device is deployed on field. JTAG security schemes presented in the past can be classified as: - fuse blowing schemes, integrity based schemes, access control and authentication based schemes [3] [13]. In this work, we have used access control approach to protect JTAG. The secret key to unlock the JTAG is generated and stored in OTP1. OEM or user cannot access it. As every device will have a unique random number stored in OTP1, key of one particular chip is not useful in unlocking JTAG of other device. The unique key will make the JTAG more secure.

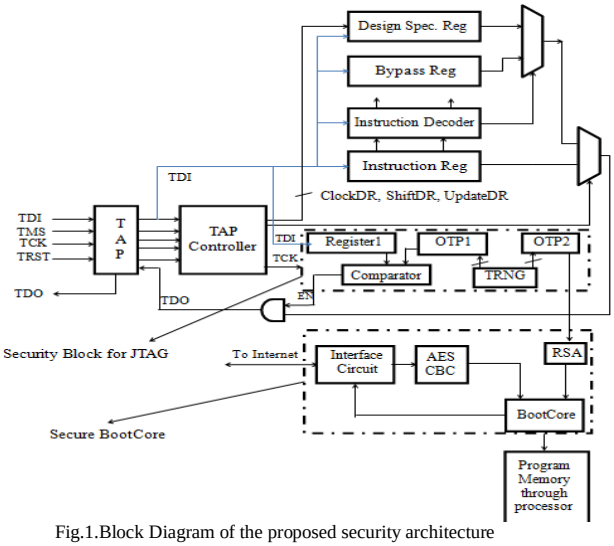

Fig.1.Block Diagram of the proposed security architecture

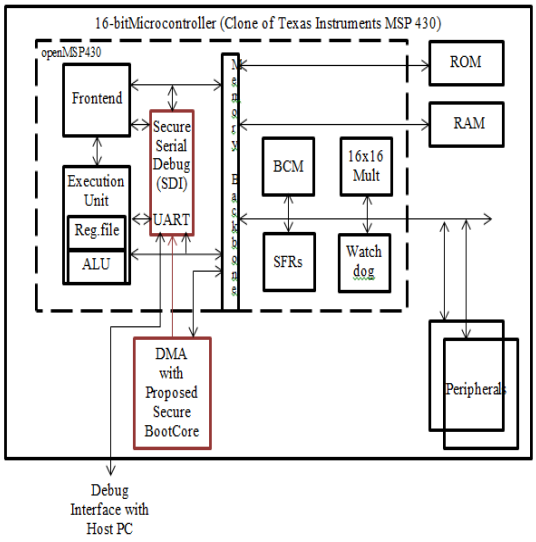

Fig. 2. openMSP430 microcontroller with proposed security block

TABLE I

COMPARISON BETWEEN PROPOSED TECHNIQUE AND EARLIER TECHNIQUES

Security Parameter	ARM TrustZone [6]	X-Cube SBSFU [7]	Crypto-bootloader [8]	Software [9]	Proposed
Protection against Firmware Reverse Engineering (RE)	Same key for all devices is vulnerable to attacks.	Not Supported	Same key for all devices is vulnerable to attacks.	Not Supported	Unique password for each device protects from RE.
JTAG Security	Protects JTAG	Not supported	Not supported	Not supported	Protects JTAG
Password Storage	Software based. Not secured	Software based. Not secured	Software based. Not secured	Software based. Not secured	Hardware based. Secured.

Firmware reverse engineering: - Reverse engineering the binaries into assembly in order to analyse the functionality and get access to secret data is reported in [10]. To perform this attack, an adversary must have an access to binaries. Most of the attack schemes on AES are based on statistics and need large number of encrypted data with the same key. Primary bootloader is accessible only through JTAG and stored in secure memory. Access to JTAG is restricted and password to unlock the JTAG is device specific which is difficult to break. The probability of adversary getting access to primary bootloader is very less unless an internal adversary leaks passwords. The primary bootloader includes device specific unique keys for RSA and AES which are stored in secure memory which can be accessed only through JTAG make secondary bootloader safe.

Adversary in ODM or OEM leaking secret key used in ciphers make the proposed security framework vulnerable. The proposed system works well when ODM and OEM are completely trusted.

Comparison with other techniques: - The comparison between different techniques with proposed security framework is presented in Table I.

VI. CONCLUSIONS

The proposed security framework include both JTAG security and secure OTA firmware update is an effective security solution for low end microcontrollers used in IoT applications. The passwords for OTA update are derived from random values stored in OTP and also unique for given device. OTA passwords are safe because they are stored in primary bootloader program which is stored in secure memory and only JTAG can access it. The unique password for every device and restricted JTAG access makes the system more secure in the field in comparison with earlier techniques.

REFERENCES

[1] Sandip Ray, et al, "Protecting the supply chain for automotives and IoTs", in the proceedings of 55th IEEE/ACM DAC- 2018.

[2] Jie Lin et al, "A Survey on Internet of Things: Architecture, Enabling Technlogies, Security and Privacy, and Applications", IEEE Internet of Things Journal Volume 4, Issue 5, Pages 1125-1142, Oct-2017.

[3] J. Da Rolt et al., "Test versus Security:Past and Present", *IEEE Trans. Emerg. Topics in Computing*., vol.2, no. 1, pp. 50-62, Mar. 2014.

[4] V.S.Varadharajan et al "Over the Air Updates for Robotic Swarms", IEEE Software, Vol.35, Issue 2, pages. 44-50, April-2018.

[5] Ryan Nakamoto, "Secure Boot and Image Authentication Technical Overview", https://www.qualcomm.com, Oct-2016.

[6] ARM Security Technology, Building a secure system using TrustZone Technology, www.arm.com, 2009.

[7] "Getting started with the X-Cube-SBSFU STM32 Cube Expansion Package", UM2262 User manual, www.st.com, April-2018.

[8] Oscar Guillen et al "Crypto-Bootloader: Secure in-field firmware updates for ultra-low MCUs", www.ti.com, September 2015.

[9] M.S. Idrees, "Secure Automotive on-board protocols: A case of over-the air firmware updates", International Workshop on Communication Technologies for Vechicles, pp 224-238, 2011.

[10] J.Park, A. Tyagi, "Using Power Clues to Hack IoT Devices: The power side channel provides for instruction-level disassembly" IEEE Consumer Electronics Magazine, Volume: 6, Issue: 3, July 2017.

[11] https://opencores.org/project/openmsp430.

[12] http://ww1.microchip.com/downloads/en/DeviceDoc/doc3600.pdf.

[13] Kyungroul lee, "A Brief Review on JTAG Security", 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2016.