IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 8, Issue 5, October - November 2020
**ISSN: 2320 – 8791 (Impact Factor: 2.317)**
**www.ijreat.org**

# Automation of Secure Boot in IIoT Edge devices

Rushi Aravind B.
*Dept. of Electronics and Communication*
*B.S. Abdur Rahman Crescent Institute of Science and Technology*
Vandalur, Chennai – 48.

Mrs. Padmausha
Asst. Prof.(Sr.Gr.), *Dept. of Electronics and Communication*
*B.S. Abdur Rahman Crescent Institute of Science and Technology*
Vandalur, Chennai – 48.

***Abstract*— Industrial Internet of Things (IIoT) is one of the highly emerging technology. Also known as the Industrial Internet or Industry 4.0, IIoT refers to the use of smart sensors and actuators to enhance manufacturing and industrial processes. An unwelcome effect of IIoT is the expansion of the attack surfaces and cyber security threat vectors. Secure Booting of IIoT edge device is one of the prevailing solutions to implement device level security. In this project we automate the secure boot process. The major goal is to develop a user-friendly tool that simplifies the complexity of generating authenticated and encrypted images for secure boot of an edge device.**

***Keywords*— Cyber Security, Secure Boot, i.MX 6UL, Industry 4.0, IIoT, Edge Device.**

## I. Introduction

Nowadays almost all Industrial IoT edge devices are connected to the cloud. These edge devices do not have the same level of security. The edge devices with low security implementations are more prone to cyber-attacks. When a hacker gains control of such an edge device, they might be able to access the data and gain control of the entire network. Hence security must be implemented in all the three layers of IIoT architecture as shown in Figure 1.

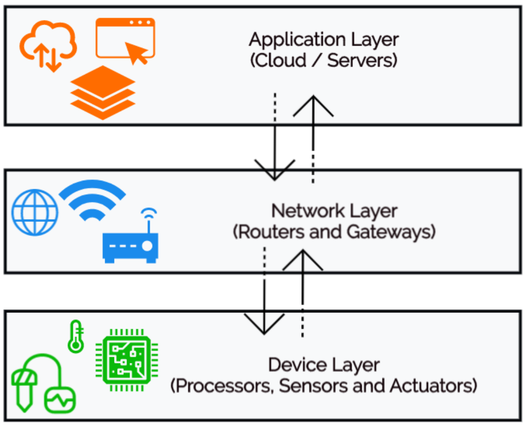


**Figure 1 Layers of Industrial IoT Architecture.**

This project targets to secure the edge devices in the device layer of IIoT architecture by implementing secure boot. Secure boot enforces authentication of application software based on cryptographic key verification each time the device is powered up or reset.

Secure transactions can be performed out by sustaining trust for the code executing on the integrated processing engine of the system. The secure authentication and optional encrypted software boot are central to establishing this root of trust.
Secure boot process is common in regular personal computers, data centres and portable devices such as smartphones and tablets. These computer systems usually include extra hardware (for example, a Trusted Program Module) to ensure the integrity of the firmware and Operating System during secure boot.
There are two broad categories of secure boot process - one is software based which increases the overall boot time by 4% and the other one is hardware based that uses cryptographic storage unit and leads to an increase in boot time by 36% [1]. There exists different ways of implementing secure boot for different embedded systems to protect the OS kernel from the attackers. It is suggested that we can use trusted hardware devices to secure the booting process and keep the kernel safe [2]. In order to provide security to edge devices as much as other devices in the network , the energy consumption of the edge devices and cost is increased .To maintain the cost of the edge devices dedicate hardware called secure elements are being proposed, these devices support faster cryptographic verification and secure storage [3].
The secure boot process extends the boot process with an additional step of verification. As a result the bootloader boots the operating system only when the integrity of the operating system has been successfully checked.
In this project, a convenient yet sophisticated software tool that automates the entire process of configuring an i.MX 6UL processor for secure boot is developed. The tool is featured to generate keys for image signing, sign the boot as well as kernel images, encrypt the image and burn the signed/unsigned/encrypted image onto the target board.

## II. Secure Boot Architecture

The processor used is NXP i.MX 6UltraLite. The i.MX 6UltraLite is a high power, high-efficiency processor family with sophisticated design of a single arm Cortex-A7 core capable of running at speeds up to 696 MHz. The i.MX 6UltraLite application processor incorporates an embedded power control system that reduces external power supply sophistication and simplifies energy sequencing. It provides several memory interfaces, including 16-bit LPDDR2, DDR3, DDR3L, NAND flash, NOR flash, eMMC, Quad SPI and a broad variety of other interfaces to connect peripherals such as WLAN, Ethernet, GPS, monitors and camera sensors [4].

Figure 2 depicts the secure boot architecture that is majorly comprised of hardware, firmware, tools and infrastructure [5].

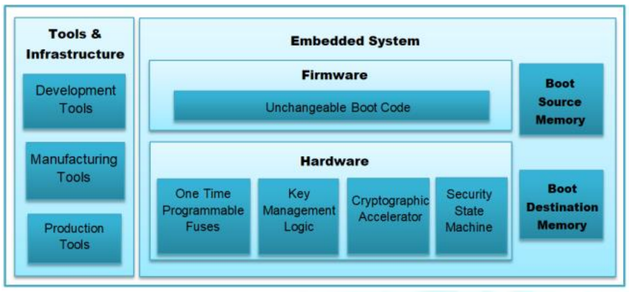


**Figure 2 Secure Boot Architecture**

## A. *Hardware*

The hardware provides physical and logical security. It is where the data security functions of the processor reside, conducting authentication based on cryptography and access control to memories and peripherals. The hardware features that enables establishment of root of trust are -

- One Time Programmable (OTP) Fuses:

The OTP (One Time Programmable) fuses are used to for root key storage. The main interface to the fuses on an i.MX device is a chip peripheral called OCOTP (On-Chip OTP). The OCOTP lets to compose and interpret the fuses.

- Hardware Security Module (HSM):

A hardware security module is a physical computing device which protects and manages digital keys, performs digital signature encryption and decryption functions, strong cryptography and other cryptographic hash functions.

## B. *Firmware*

The i.MX RT contains a substantial 96 KB ROM (Read Only Memory) which provides the functionality of boot loading and the secure boot.

- Bootloader:

The portion of the ROM firmware that always runs during the system boot is bootloader. U-Boot [6] is the tool recommended as the bootloader. To be able to boot from it, the U-boot must be loaded onto a machine. U-Boot images are board-specific and can be programmed to support boot from different sources Tools and Infrastructure.

- Kernel image:

The Freescale i.MX BSP contains a pre-built kernel image based on the 4.1.15 version of the Linux kernel and the device tree files associated with each platform [6].

- High Assurance Boot (HAB) :

The HAB library is a component of the boot ROM on i.MX processor. HAB implements boot ROM level security which cannot alter once it is fused. The version of HAB used in this project is 4.3. HAB based on the principle of digital signature. HAB digital signature is a mix of open-ssl certification, MD5 hashing and public and private key RSA-AES-DES checks. When the system boots, the HAB instructions will examine the hash values present inside the boot ROM and signed images. When these two hash values match, the HAB will allow the platform to boot the images. Else the system shall stop all the process [7].

## C. *Tools and Infrastructure*

There are various tools used in the development, manufacturing and deployment of the device. Tools for Key management resources, firmware file development,and firmware connection and installation into the system are required to enforce stable boot design.

The goal of authenticating application firmware on each and every boot is achievable with these components considered.

## III. SECURE BOOT PROCESS

The secure boot process starts with the ROM reading e-Fuses to determine the security configuration of the SoC (System on Chip) and the type of the boot device. The ROM then loads the images to memory (See Figure 3). For HAB (High Assurance Boot [5]), the bootloader image contains both the bootloader itself in addition to commands that the HAB uses to verify the image, digital signature data and public key certificate data which are collectively called Command Sequence File (CSF) [8] data. The CSF data is generated off-line using the Code-Signing Tool (CST) [9].

Once ROM has completed loading the images, execution is then passed to the secure components which will verify the signatures. For a safely configured SoC, execution is not permitted to exit the ROM when signature authentication fails. The exact behaviour on signature verification failure at the ROM stage is SoC dependent. If all signatures, including image decryption, are successful then execution is passed to the next images which can perform similar steps to verify the next boot stage by calling back into the secure API (Application Program Interface, here refers to HAB).

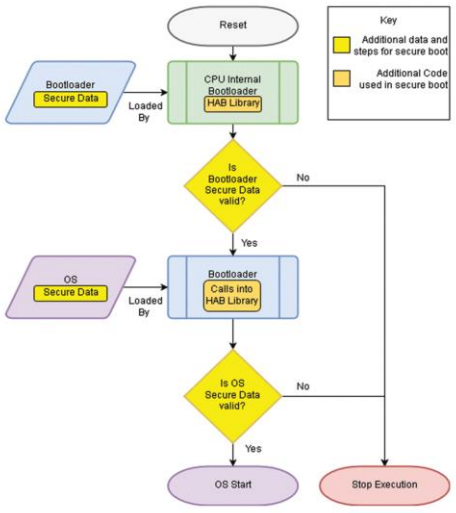


**Figure 3 Secure Boot Process**

## IV. Methodology

This section explains the algorithm and program flow of the i.MX 6UL Boot Utility Software.

### A. *Algorithm*

1. Start
2. Get the necessary inputs for key generation.
3. Generate text files with serial number and pass phrase.
4. Generate keys by passing the inputs to OpenSSL via CST.
5. Create Command Sequence File (along with Data Encryption Key commands for encrypted boot) and convert it into binary format.
6. Find the binary image size and pad the kernel image accordingly.
7. Create Image Vector Table, convert it into binary form and append it with the kernel image.
8. Attach CSF to the kernel image appended with IVT and thus signed image is generated.
9. Burn the keys and Data Encryption Key blob on to the fuses in target board.
10. Burn the image on the SD card.
11. Stop.

### B. *Flowchart*

The flow diagram of the i.MX 6UL Boot Utility Software is depicted in Figure 4

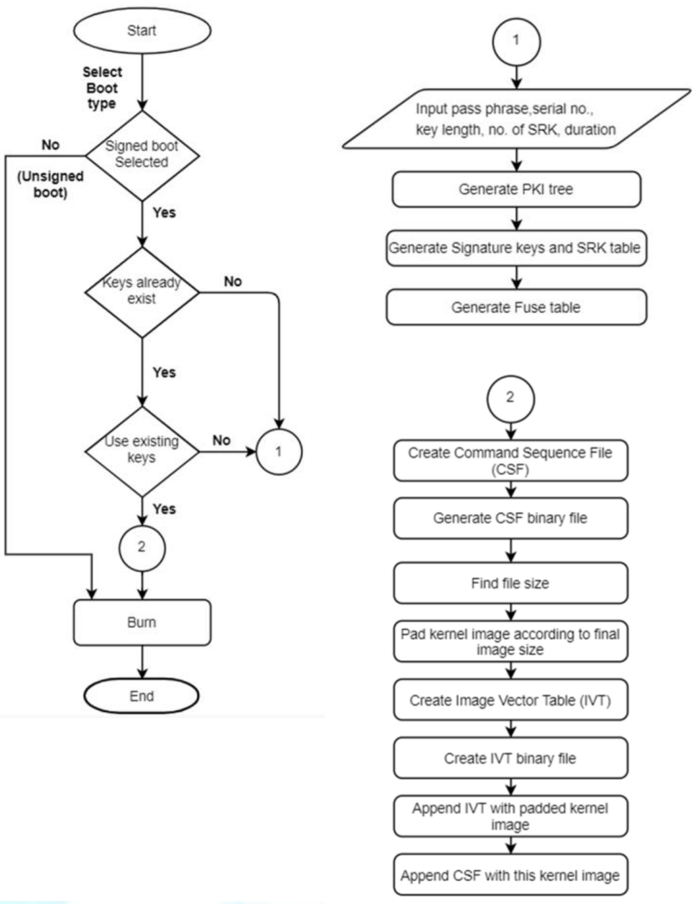

Start
Select Boot type
Signed boot Selected
No
(Unsigned boot)
Yes
Keys already exist
No
Yes
Use existing keys
No
Yes
1
2
Burn
End
1
Input pass phrase,serial no., key length, no. of SRK, duration
Generate PKI tree
Generate Signature keys and SRK table
Generate Fuse table
2
Create Command Sequence File (CSF)
Generate CSF binary file
Find file size
Pad kernel image according to final image size
Create Image Vector Table (IVT)
Create IVT binary file
Append IVT with padded kernel image
Append CSF with this kernel image

**Figure 4 Flowchart**

## V. Implementation

The i.MX 6UL Boot Utility Software is built using python and runs on Linux platform. The screenshots of the software tool are shown below (Figures 5-7)

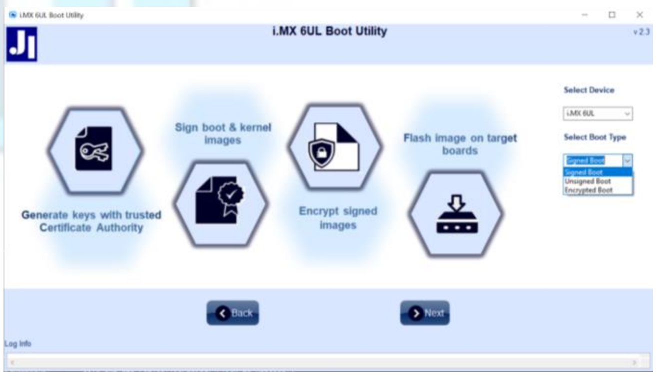
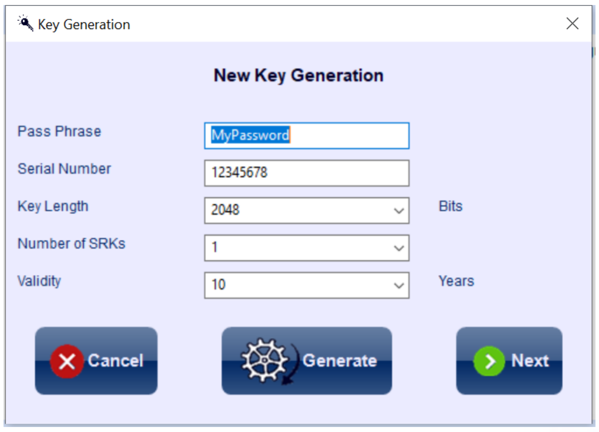


**Figure 5 Welcome Screen**



Key Generation

**New Key Generation**

| | | |
|---|---|---|
| Pass Phrase | MyPassword | |
| Serial Number | 12345678 | |
| Key Length | 2048 | Bits |
| Number of SRKs | 1 | |
| Validity | 10 | Years |

**Figure 6 Provision to generate new public and private keys**

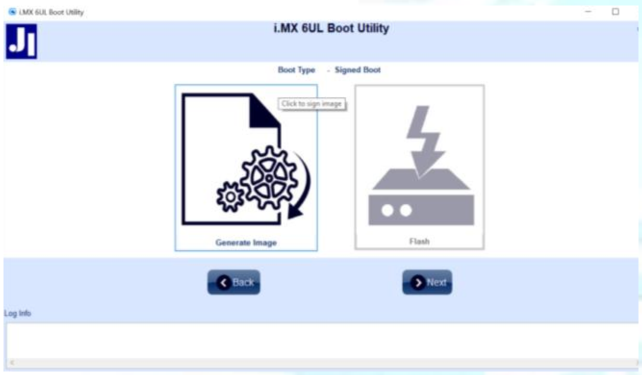


**Figure 7 Image Signing**

Table 1 shows the reduction in time taken to configure i.MX 6UL processor using the i.MX 6UL Boot Utility Tool.

| Method | Time Consumption |
|---|---|
| Manual Configuration | Approx., half an hour |
| Using i.MX 6 UL Boot Utility | 5 - 6 minutes |

**Table 1 Comparison of time consumption for device configuration**

## VI. CONCLUSION

Secure boot is one of the primary methods of protecting edge devices at device level. In this project we have developed a software tool that automates the time consuming process of configuring i.MX 6UL processor for secure boot. The actual manual configuration would take about 30 minutes whereas using the i.MX 6UL Boot Utility tool the device can be configured within 5 – 6 minutes. It makes the process simpler, faster and efficient by reducing the users' efforts by one fifth of actual manual configuration. This will enable embedded product developers to generate signed images and burn them on the go. By using this tool, the time required to generate the signed boot is reduced. The table below shows a comparison of time between manual secure boot and secure boot using the i.MX Boot Utility tool.

## ACKNOWLEDGMENT

We extend our sincere thanks to the Department of Electronics and Communication Engineering of B.S Abdur Rahman Crescent Institute of Science and Technology and Jasmin Infotech Pvt. Ltd for providing us professional guidance and support to carry out this industrial project.

## REFERENCES

[1] M. G. Y. N. O. L. M. A. Christos Profentzas, "Performance of Secure boot in Embedded system," in *15th Internation IEEE conference on Distributed computing in sensor system*, 2019.

[2] K. A. Rashmi R., "Secure Boot of Embedded Application - A Review," in *2nd International IEEE conference on Electronics, Communication and Aerospace Technology*, 2018.

[3] A. R. Tobias Schlapfer, "Security on IoT Devices with Secure Elements," in *Embedded World Conference*, 2019.

[4] "www.nxp.com," NXP, 2020. [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-6-processors/i-mx-6ultralite-processor-low-power-secure-arm-cortex-a7-core:i.MX6UL.

[5] NXP, "Realizing Today's Security Requirements: Achieving End-To-End Security with a Crossover Processor," September 2018.

[6] Freescale Semiconductor, Inc., "i.MX Linux® User's Guide," NXP, 2016.

[7] S. E.-. S. i. S. T. P. L. Ajith P V. [Online]. Available: https://www.iwavesystems.com/high-assurance-booting-imx6.

[8] NXP, "Secure Boot on i.MX 50, i.MX 53, i.MX 6 and i.MX 7 Series using HABv4," 2018.

[9] NXP, "Code-Signing Tool User's Guide," 2018.