Integrity Issues for IoT: From Experiment to Classification Introducing Integrity Probes

Pascal Urien
Telecom ParisTech, 23 Avenue d'Italie, Paris, France

Keywords: Security, Trust, Internet of Things, Software Update.

Abstract: This paper presents a tentative classification of IoT devices. The goal is to provide a qualitative estimation of risks induced by device hardware and software resources involved in firmware update operations. We present technical features available in existing devices, and comment associated threats. From this analysis we extract five basic security attributes: one time programmable memory, firmware downloader, secure firmware downloader, tamper resistant hardware, and diversified keys. From these parameters we deduce and comment six security classes. We describe an innovative integrity probe working with commercial programmers, of which goal is to verify a bootloader integrity.

1 INTRODUCTION

According to a report (SIA and SRC, 2015) from *the Semiconductor Industry Association* (SIA) and the *Semiconductor Research Corporation* (SRC), the Internet of Things (IoT) could involve trillions of devices by 2030. In this context "*security and privacy are two of the biggest challenges for future systems"*. The paper (Ronen and Shamir, 2016) introduces "*a new taxonomy of attacks on IoT devices, which is based on how the attacker deviates feature from their official functionality"*. It defines four types of attacking behavior, 1) Ignoring the functionality, 2) Reducing the functionality, 3) Misusing the functionality, 4) Extending the functionality. This raises a critical issue about the trust level needed by IoT devices, and how to get some integrity insurance for embedded firmware. We propose a classification model based on three software properties (bootloader, secure bootloader, and diversified keys) and two physical characteristics: OTP (One Time Programmable) memory, and tamper resistance. This approach results from experiments or analysis performed on multiple processors. We also introduce the integrity probe (ITP) concept, a firmware downloaded thanks to bootloader, of which goal is to verify the bootloader integrity.

The paper is constructed according to the following outline. Section 2 presents IoT architecture in our context; it introduces device programming protocols, bootloader, device firmware upgrade, secure bootloader and tamper resistant requirements. Section 3 comments some processors used in IoT systems and they update mechanisms; it details FLASH controller, Bluetooth SoC, Wi-Fi SoC, and AVR processors. Section 4 describes our security classification proposal dealing with six classes, based on five security attributes OTP, firmware loader, secure firmware loader, tamper resistant hardware, and diversified keys. Section 5 introduces integrity probes tested with commercial SPI programmer tokens. Finally section 6 concludes this paper.

2 IoT DEVICE ARCHITECTURE

This section attempts to define the hardware structure of IoT devices addressed by this paper.

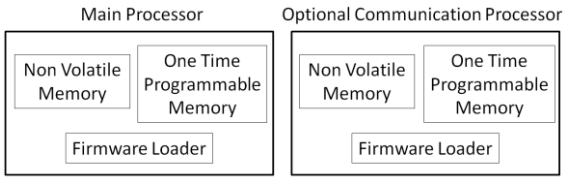

Figure 1: IoT device architecture.

An object is built (see figure 1) around a micro-controller (that we call *Main Processor*, MP) including RAM memory 1-10KB), non volatile memory (such as FLASH 10-100KB), and optional ROM (10-100KB). An optional second processor (*Communication Processor*, CP) provides communication resources (Wi-Fi, Bluetooth),

Urien, P.
Integrity Issues for IoT: From Experiment to Classification Introducing Integrity Probes.
DOI: 10.5220/0007746903440350
In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019)*, pages 344-350
ISBN: 978-989-758-369-8

according to SoC (*System on Chip*) technology. When used, it is controlled by the MP entity.

Internal memories programming can be performed by several physical ways including JTAG (Joint Test Action Group) interface, Parallel Programming (PP, a set of dedicated pins), or Serial Peripheral Interface (SPI).

SPI is a *de facto* standard widely used by the IoT industry. The SPI bus comprises four logic signals: SCLK serial clock, MOSI master out slave in, MISO master in slave out, SS slave select. Normalized SPI connectors include ground and power feeding; they have 6 or 10 pins. Serial SPI commands are used to erase, read or write memory content and special security registers such as fuse or locks as detail in section 3.4.

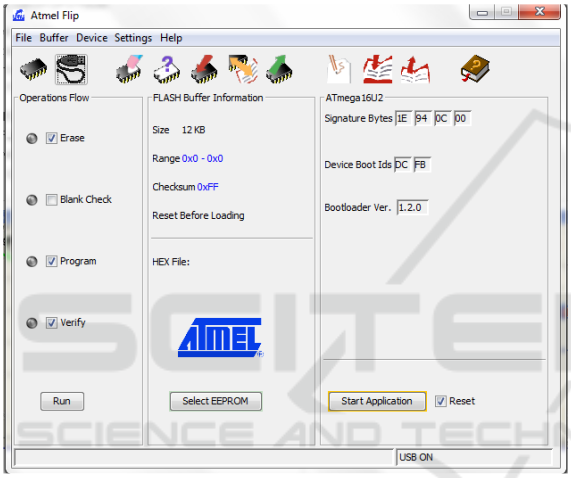

Figure 2: The Atmel free *Flip Software* is used to update AVR processor, embedding a bootloader (usually referred as DFU) implementing the DFU protocol.

In order to enable easy remote software update, bootloader firmwares are inserted during manufactory process. A bootloader is a command interpreter, typically working over a communication link (USB, UART...), used to store data in non volatile memory. Numerous open bootloaders are available, for example those belonging to the STK500 family (an ATMEL protocol) used for AVR (see section 3.4) processors, which are enhanced versions of the software designed in 2003 by Jason P. Kyle. The popular AVRDUDE (AVR Downloader/UploaDEr) open software is a utility to download/upload/manipulate the non volatile memory of AVR microcontrollers. It is compatible with STK500 bootloaders, and some SPI programmers. In section 5.1 we introduce USBasp, an open hardware/software SPI programmer.

DFU (Device Firmware Upgrade) is a protocol supported by Atmel bootloaders, usually referred as DFU firmware. After a short circuit between the reset pin and the ground, DFU is activated, and it becomes possible to download a new firmware (see figure 2). Some open versions have been developed, for example the LUFA Library (2010) written by Dean Camera.

It is obviously possible to add security features to bootloader, typically to check the software update authenticity. Firmware encryption can be needed for intellectual property requirements. Public asymmetric keys or symmetric keys are generally used for information authentication (signature checking), or firmware encryption. In that case tamper resistance could be an important requirement, since malware may be inserted in non genuine firmware, or cryptographic keys can be recovered from side channel attacks.

3 SOME IoT PROCESSORS

3.1 Flash Controller

Popular USB flash drives are built over FLASH controller chips (see figure 3), which comprise a CPU (8051 like), ROM, and RAM. Their detailed specifications are usually not publicly available.

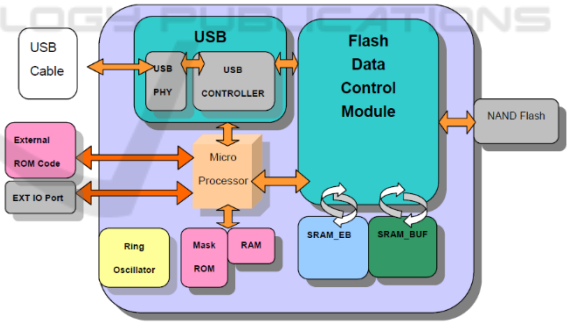

Figure 3: Structure of FLASH Controller PS2251-33 from the *Phison Electronics Corporation*. A bootloader is stored in the chip ROM.

The ROM stores a bootloader or a part of it when an external ROM is available. Thanks to this component and to USB connectivity, the drive firmware is downloaded in FLASH. Dedicated websites (flashboot.ru, www.usbdev.ru...) manage FLASH drive databases bound to controller models; they provide firmware images and recovery tools required for their upload. The firmware may include security features (Jago, 2018), for example FLASH encryption bound to user's password. Nevertheless

there is generally no security mechanism for the firmware upload. The paper (Nohl and KriBler and Lell, 2014) demonstrated attacks against FLASH drives using chips like the PS2251-33 (illustrated by figure 3); the main idea is to upload modified firmware providing USB profiles such as keyboard or network interface (Wilson, 2014).

3.2 HC05 and SoC CSR BC417143

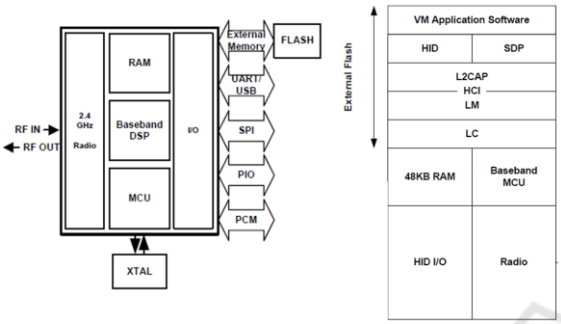

Figure 4: The CSR BC417143 SoC.

The BlueCore™-External is a single chip radio and baseband IC (see figure 4) for Bluetooth 2.4 GHz systems. It interfaces up to 8Mbit of external FLASH memory. It is manufactured by CSR (*Cambridge Silicon Radio*) a multinational fabless semiconductor company acquired by Qualcomm in 2015.

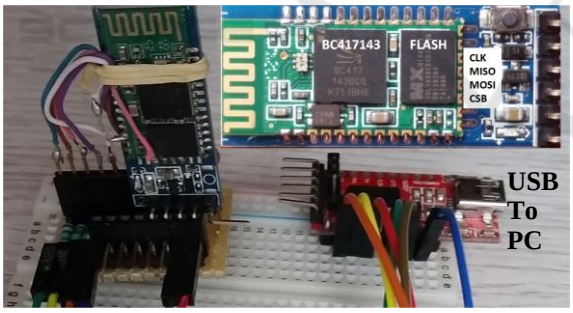

Figure 5: HC05 Bluetooth dongle (upper left) and hardware adapter needed to interface the *BlueStack* for Win32 PC.

The CSR Bluetooth software stack provides Bluetooth features, including in particular a RFCOMM profile.

The SoC (see figure 4) is equipped with 48 KB RAM, but it doesn't include ROM. According to its datasheet, the external FLASH is programmed via a serial interface (SPI) using 16-bit addresses and 16-bit words. Updates may be performed when the internal processor is running or is stopped. A DFU (*Device Firmware Upgrade*) bootloader must be loaded into the FLASH device, before the UART or USB functional interfaces can be used. This initial FLASH programming is done via the SPI interface. A dedicated programmer (USB to SPI) performs this operation, driven from *BlueSuite* proprietary software stack (illustrated by figure 6). Nevertheless a SHIM library was designed by (Willem, 2016), which in conjunction with other open tools, enables to use *BlueSuite* without dedicated adapter.

HC05/06 devices are popular Bluetooth dongles based on *BC417143* SoC with 8 Mbits FLASH. It is possible to solder wires on the four SPI pins (CLK, MOSI, MISO, CSB, see figure 5) and afterwards to dump or download the embedded firmware.

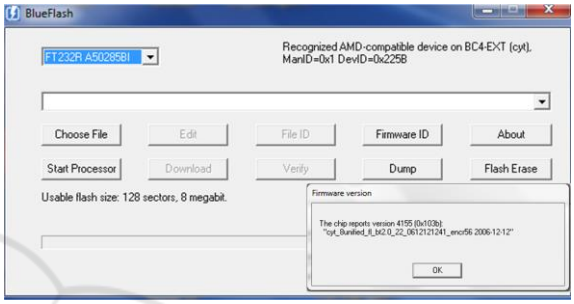

Figure 6: The *Blue Flash Software* version 2.62 may upload or download the HC05 Bluetooth dongle firmware.

3.3 SoC ESP8266

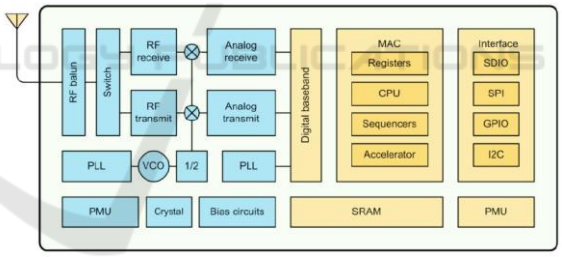

Figure 7: The ESP8266 SoC internal structure.

The ESP8266 (see figure 7) is a popular low cost Wi-Fi SoC, manufactured by *Espressif Systems*. It is based on a 32-bit RISC microprocessor, from the *Tensilica Company*, running at 80 MHz. No detailed specifications are publicly available. Nevertheless some WEB sites (Filippov, 2015) manage information addressing the physical memory layout. The SoC embeds a 64KB ROM, a set of RAMs (about 80KB, including instruction cache), 80KB of DRAM for user data, and FLASH (up to 16MB). Some ESP8266 support "*secure boot*", i.e. the firmware stored in the FLASH is encrypted by an AES key, burnt in one time programmable (i.e. OTP) fuses.

The ROM embeds a bootloader associated to an UART (Gratton, 2018), which enables firmware

downloading in external FLASH, thanks to a dedicated tool, illustrated by figure 8.

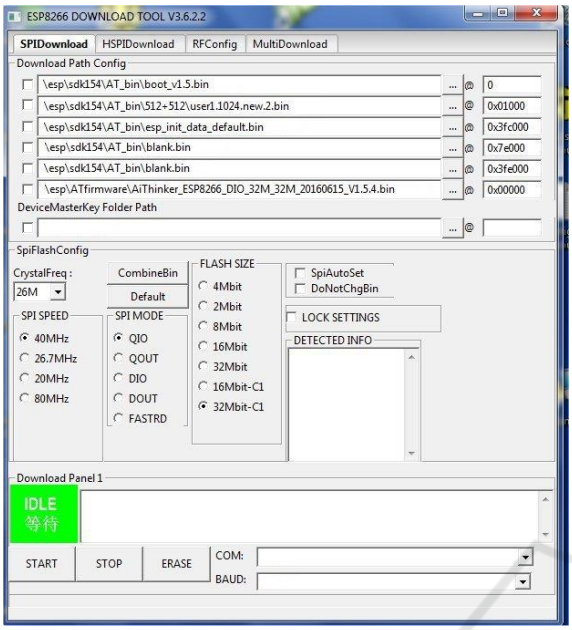

Figure 8: The ESP8266 firmware downloading tool.

The *Espressif Systems Company* provides a free non OS software development kit (*NONOS SDK*) whose code comprises compiled libraries (in particular TCP/IP stack) and open sources. Nevertheless most of market modules are manufactured by a third-party Ai-Thinker and are referred as "ESP-xx" product. Figure 8 illustrates the downloading of such SDK combined with a firmware (providing AT-Commands) from Ai-Thinker.

3.4 Atmel AVR

AVR is a family of microcontrollers developed the by the Atmel Company, acquired by Microchip Technology in 2016. It uses (see figure 9) on-chip FLASH memory for program storage; RAM and EEPROM resources are also provided for user's data. Because no ROM is available for code storage (see figure 9), such electronic chips provides computing environment that we refer as "*Bare Metal*" because FLASH memory contents can be fully erased.

The security of AVR chips is controlled by two kinds of registers: locks and fuses (see figure 10). Locks are reset thanks to commands sent over programming interfaces such as SPI. They manage the FLASH memory access policy (read and write operations), the fuses security policy (read and write policy), the bootloader optional use and its available sizes. Fuses are not erased by the SPI reset command;

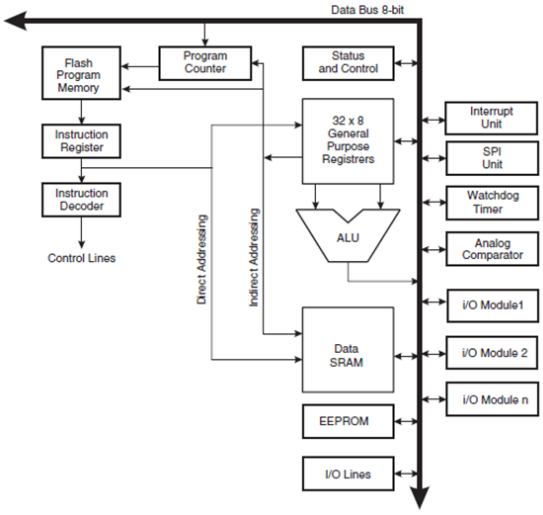

Figure 9: The ATMEGA8 AVR includes 8KB FLASH, up to 2KB bootloader, 1KB RAM, 512B EEPROM. The chip programming is managed through an SPI interface.

they are modified by dedicated commands. They control some programming features such as RESET pin or SPI protocol use, and other physical parameters dealing with clock or logical voltages.

Because AVR chips have no ROM it is not possible to permanently disable FLASH writing operations. In other words it is always possible to erase and upload code in FLASH memory.

CLEAR →	Clear Locks →	R/W Any Fuses
Serial Programming (SP) Requires RESET signal	Memories Access	Fuse: Disable RESET SP & SP
Parallel Programming (PP) Requires RESET signal	Fuses Access	Fuse: Disable SP from PP only
High Voltage Programming CLEAR under any conditions	Bootloader Configuration	Fuse: Others Voltage, Clock

Figure 10: AVR Fuses and Locks.

4 SECURITY CLASSES

Based on the previous observations, we propose security classes for IoT devices. The goal is to provide a qualitative estimation of risks induced by firmware remote updates according to device logical and hardware security resources. We identify five security attributes leading to six classes of IoT devices including or not OTP (*One Time Programming*) memory indicated by the "+" suffix.

4.1 Security Attributes

The five security attributes are the following:

4.1.1 One Time Programming (OTP)

The OTP availability means that a device cannot be fully reprogrammed, and have some permanent code and data such as cryptographic keys. When OTP is missing we qualify the device of "Bare Metal".

4.1.2 Firmware Loader (FLD)

A firmware loader is mainly a command interpreter that enables logical/remote firmware update. It avoids the use of physical procedures such as *Serial Programming* or *Parallel Programming*. It is stored in non volatile memory, either erasable or not.

4.1.3 Secure Firmware Loader (FLD-SEC)

A secure bootloader checks the authenticity and integrity of firmware updates by cryptographic means. This implies the use of symmetric secret keys, asymmetric private/public keys associated to certificates.

4.1.4 Tamper Resistant Key (TRT-KEY)

Cryptographic keys can be recovered by side-channel attacks. A tamper resistant computing environment implements hardware and software countermeasures that avoid these threats. For example bank cards include secure elements implementing such physical and logical features.

4.1.5 Diversified Key (DIV-KEY)

The use of diversified secret keys limits the side channel attack scope to a single object. The lack of tamper resistant computing and the use of single secret shared by multiple nodes may lead to major security threats. An attack against smart bulbs was detailed in the paper (Ronen and O'Flynn and Shamir and Weingarten, 2016), in which a single symmetric key shared by multiple bulbs and used for secure uploading, was recovered by a side channel attack.

4.2 Security Classes

The figure 10 illustrates our proposed classification, according to a tree structure dealing with the five security attributes previously defined.

The availability of OTP is indicated by the "+" suffix.

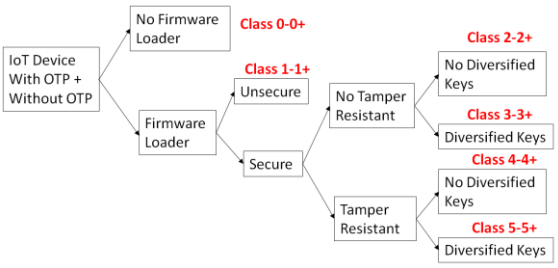

Figure 11: Definition of security classes.

Class0-0+ devices have no firmware loader, so they can only be programmed by physical means, i.e. dedicated protocols and physical ports.

Class1-1+ devices have unsecure firmware loaders, located either in FLASH or OTP. No cryptographic keys are needed; firmware can be updated during the device lifetime by logical means dealing with various hardware interfaces such as USB ports or serial links.

Class2-2+ devices use secure bootloader, without tamper resistant features and no diversified keys. A single key is shared by devices. It can be a symmetric one (i.e. AES) like in smart bulbs from Philips, or an asymmetric one for example a public key needed to check the signature of software updates.

Classe3-3+ devices use secure bootloader, without tamper resistant features, but with diversified keys. This use case typically target devices storing asymmetric AES keys.

Class4-4+ devices deal with secure bootloader, based on tamper resistant hardware, but no diversified keys. This means that it is not possible (at least very difficult) to modify the key (for example a public key used for software update authentication) or to recover a symmetric key value thanks to side channel attacks.

Class5-5+ devices comprise a secure bootloader with tamper resistant hardware and diversified keys. It enables high security IoT frameworks, in which software updates are encrypted with symmetric (AES) key bound to device serial number.

4.3 Example

AVR micro-controller units (MCU) without bootloader belong to Class0.

AVR micro-controller units (MCU) with bootloader (for example the "*Arduino*" family) belong to Class1.

USB flash drives embedding ROM and bootloader belong to Class1+.

Philips hue smart bulbs belong to Class2; they embed secure bootloader with a shared single symmetric key.

The IETF working group SUIT (*Software Updates for Internet of Things*) target Class2+ devices embedding secure bootloader with public key, and Class3+ devices supporting bootloader and diversified symmetric key.

Class4+ is a tamper resistant enhancement of Class2+, it could address devices compatible with SUIT of which public key cannot be modified.

Highly secure devices such as bank cards belong to Class5+.

5 USECASE: INTEGRITY PROBE

We believe that is not possible to check the firmware integrity for Class0 devices. Nevertheless a possible alternative is to re-program the device if a malware is suspected.

Another direction is to flash a bootloader in the device, which according to our approach, is labelled as Class1 (unsecure bootloader without OTP). Thereafter dedicated software, referred as integrity probe (ITP) is downloaded. ITP verifies the bootloader integrity thanks to algorithms based on one way function, whose result cannot be predict by malware without the knowledge of keys used to compute memory content hashes in a pseudo random way.

5.1 About USBASP

USBasp is a USB in-circuit programmer for Atmel AVR controllers, designed by Thomas Fischl (2011). It is based on ATMega8 processor, illustrated by figure 9. It uses a firmware-only USB driver, and is compatible with open framework such as ARVDUDE (*AVR Downloader/UploaDEr*), libusb, and libusbK.

An open firmware written by Thomas Fischl (2011) is freely available for these devices, as previously mentioned, it works with AVRDUDE, used for example by the Arduino IDE. USBasp tokens can be bought from numerous vendors, at very low prices (less than 5$).

Because SPI flashers are used to program devices, their firmware integrity is a major security issue. Furthermore their firmware can be easily modified during the delivery process. In the next section we develop the early concept of an innovative integrity probe, of which goal is to verify the bootloader integrity, before the loading of the ISP programmer firmware. More details are available in (Urien, 2019).

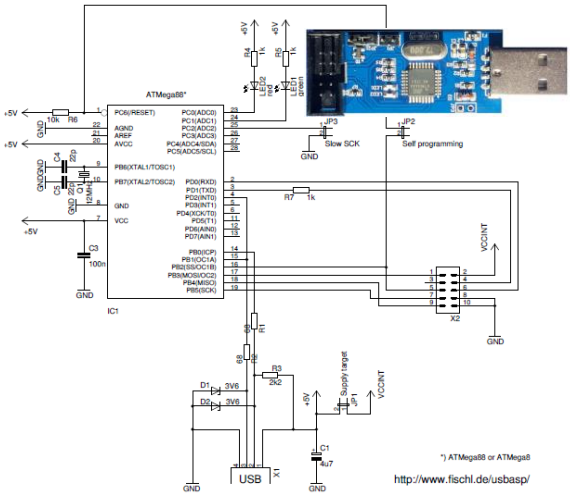

Figure 12: Schematics of the open USBasp SPI programmer. It is possible to program the device with a bootloader.

5.2 Open DFU for USBasp

The blog (Thomson, 2011) details the design of open Device Firmware Upgrade (DFU) software dedicated to commercial USBasp tokens. Thanks to this DFU the USBasp token belongs to Class1, previously defined. The DFU is activated thanks to a short circuit between ground and the reset pin of the ATMEGA8 processor; a slot of 5 seconds is thereafter available for firmware upload, handled by the AVRDUDE software. The DFU code size is 2KB. Commercial USBasp tokens can be erased and re-flashed by such DFU. This operation may be performed via the free Atmel Studio software, which supports a device programming tool working with the SPI protocol.

5.3 Integrity Probe

An integrity probe (ITP) is software whose goal is to check the bootloader (or DFU) integrity. It is downloaded in the processor memory by the bootloader (or DFU) firmware. The basic ITP concept is to hash the code of both the DFU and ITP, and the data stored in the processor memories, in a pseudo random way. In our case the DFU size is 2KB and the ITP size is 6KB. The ATMEGA8 has 1KB RAM and 512B EEPROM; so the total memory amount is 9,5KB. Memories are hashed in a pseudo random order, according to a permutation (P) working in the address space (A = [0, 9728[)

iCode= SHA3(A(0) || A(1) ||...|| A(i) ||...|| A(9727))

Unused memory locations are filled with pseudo random values, computed from a pseudo random number generator (PRNG). The code size of the SHA3 digest function is about 3KB. The less significant part (2 bytes, an integer ranging between 0 et 65635) of the integrity code (IC, 32 bytes) is displayed to the user, thanks to LEDs blinking according to each decimal digit value (up to 5 digits). Depending of the USBasp hardware the final digest value (32 bytes), including the computation time (ICT) may be read via an UART. The integrity probe runs in about 10,000ms. The execution time is also included in the integrity measurement process.

6 CONCLUSIONS

In this paper we propose a classification of IoT devices based on five security attributes. We also introduce the idea of integrity probes for Class1 devices that embeds bootloader without security or OTP. We hope that this approach could lead to a better security characterisation of industrial IoT devices.

REFERENCES

SIA, SRC, 2015, "Rebooting the IT Revolution: A Call to Action", Semiconductor Industry Association / Semiconductor Research Corporation report.

Ronen, E., Shamir, A., 2016. "Extended Functionality Attacks on IoT Devices: The Case of Smart Lights", 2016 IEEE European Symposium on Security and Privacy (EuroS&P).

Nohl, K., KriBler, S., Lell, J., 2014. "BadUSB - On Accessories that Turn Evil", Blackhat 2014 USA.

Jago, D., 2018, "Security analysis of USB drive", Master's thesis report.

Wilson, B., 2014, "Phison 2251-03 (2303) Custom Firmware & Existing Firmware Patches (BadUSB)", https://github.com/brandonlw/Psychson.

Filippov, M., 2015, ESP8266 Community Forum, "Memory Map", https://github.com/esp8266/esp8266-wiki/wiki/Memory-Map, seen January 2019

Gratton, A., 2018, ESP8266 Community Forum, "Serial Protocol", https://github.com/espressif/ esptool/wiki/Serial-Protocol, seen January 2019.

Ronen, E, O'Flynn, C., Shamir, A; Weingarten, A., 2016, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction", Cryptology ePrint Archive, Report 2016/1047.

Thomson, J., 2011, "Project Ouroboros - Reflashing A Betemcu USBasp Programmer", Jonathan Thomson's web journal, https://jethomson.wordpress.com/ 2011/08/18/project-ouroboros-reflashing-a-betemcu-usbasp-programmer/, seen January 2019.

Willem, F., 2016, CSR BlueCore USB SPI programmer/debugger, https://github.com/lorf/csr-spi-ftdi, seen January 2019.

Urien, P., 2019," Integrity Probe: Using Programmer as Root Of Trust For Bare Metal Blockchain Crypto Terminal", Fifth International Conference On Mobile And Secure Services, MobiSecServ2019.