symmetry

MDPI

Review

A State-of-the-Art Review on the Security of Mainstream IoT Wireless PAN Protocol Stacks

Georgios Kambourakis [1,*,†], Constantinos Kolias [2,†], Dimitrios Geneiatakis [1,†], Georgios Karopoulos [1,†], Georgios Michail Makrakis [2,†] and Ioannis Kounelis [1,†]

[1] European Commission, Joint Research Centre (JRC), 21027 Ispra, Italy; dimitrios.geneiatakis@ec.europa.eu (D.G.); georgios.karopoulos@ec.europa.eu (G.K.); ioannis.kounelis@ec.europa.eu (I.K.)

[2] Faculty of Computer Science, College of Engineering, University of Idaho, Idaho Falls, ID 83402, USA; kolias@uidaho.edu (C.K.); makr7178@vandals.uidaho.edu (G.M.M.)

* Correspondence: georgios.kampourakis@ec.europa.eu; Tel.: +39-0332-78-5013

† These authors contributed equally to this work.

Received: 2 March 2020; Accepted: 25 March 2020; Published: 6 April 2020

Abstract: Protocol stacks specifically designed for the Internet of Things (IoT) have become commonplace. At the same time, security and privacy concerns regarding IoT technologies are also attracting significant attention given the risks that are inherently associated with the respective devices and their numerous applications, ranging from healthcare, smart homes, and cities, to intelligent transportation systems and industrial automation. Considering the still heterogeneous nature of the majority of IoT protocols, a major concern is to find common references for investigating and analyzing their security and privacy threats. To this end, and on top of the current literature, this work provides a comprehensive, vis-à-vis comparison of the security aspects of the thus far most widespread IoT Wireless Personal Area Network (WPAN) protocols, namely BLE, Z-Wave, ZigBee, Thread, and EnOcean. A succinct but exhaustive review of the relevant literature from 2013 up to now is offered as a side contribution.

Keywords: IoT; Bluetooth Low Energy; ZigBee; Thread; Z-wave; EnOcean; Security; WPAN

1. Introduction

Internet of Things (IoT) enables the interaction of physical components and objects with the cyberspace. Precisely, IoT technologies and protocols not only make possible the exchange of data at a global scale via the traditional internet infrastructure, but also facilitate a more direct way of communication with devices in their vicinity. Recent studies [1] estimate that by 2025 more than 75 million IoT devices will be in place. Up to now, a plethora of IoT based solutions have been developed to support real-time monitoring services in domains such as smart homes, smart cities, healthcare, and workplace automation.

On the other hand, since these devices are among other ecosystems already present in our workplaces and houses, their security and privacy aspects are a decisive factor toward the acceptance of the related technologies and subsequently their market penetration. Namely, the direct impact of IoT systems in the physical world in combination with their galloping adoption rate render their security a matter of utmost urgency. Moreover, the vast majority of, e.g., home automation devices utilize wireless technologies to enable communication with each other or with a controller or gateway. Thus, due to the idiosyncrasies of the wireless channel and IoT limited processing capacities, security becomes even more important.

During the last decade, security in the IoT has been the focus of much concern and research. Numerous studies [2–6] have been published regarding the insecurity of IoT ecosystem. In particular,

Rizvi et al. [2] and HaddadPajouh et al. [3] reviewed different types of application layer attacks against IoT devices. In a more comprehensive study, the authors of [6] reviewed IoT security challenges and reported well-known attacks across the various layers of a typical IoT architecture. They also proposed possible security improvements considering IoT integration with other state-of-the-art technologies such as blockchain and fog computing. The contribution in [4] discusses how IoT systems could be exploited in order to accomplish distributed denial of service attacks (DDoS). Additionally, the authors of [5] studied security and privacy issues in a smart home environment.

The work at hand provides a detailed state-of-the-art review of the security aspects of five well-established and/or promising, modern Wireless Personal Area Network (WPAN) protocol stacks, namely Bluetooth Low Energy (BLE), ZigBee, Z-Wave, Thread, and EnOcean. We discuss and review their security characteristics and related attacks with emphasis on MAC and higher layers, while physical layer security analysis remains out of scope. To the best of our knowledge, such a comprehensive review, providing a common reference for the examination of the security aspects of the currently most widespread WPAN IoT protocols, is still missing from the relevant literature.

The remainder of the paper is structured as follows. Section 2 introduces the protocols of interest and details on their security aspects. Section 3 offers a review of the relevant literature works with a focus on attacks and countermeasures, while Section 4 summarizes key security issues of WPAN protocols. Section 5 concludes and gives pointers to future research.

2. Overview and Security Features

This section provides a concise analysis of the security features of each of the selected IoT protocols. Our goal is not to delve into the details of each implementation, given that this is provided by the respected specifications, but to ease and subserve the reading of the subsequent sections. Figure 1 offers a generic representation of the protocol stack of each considered technology with reference to the Open Systems Interconnection (OSI) model, while Table 1 summarizes the basic security features provided per protocol of interest.

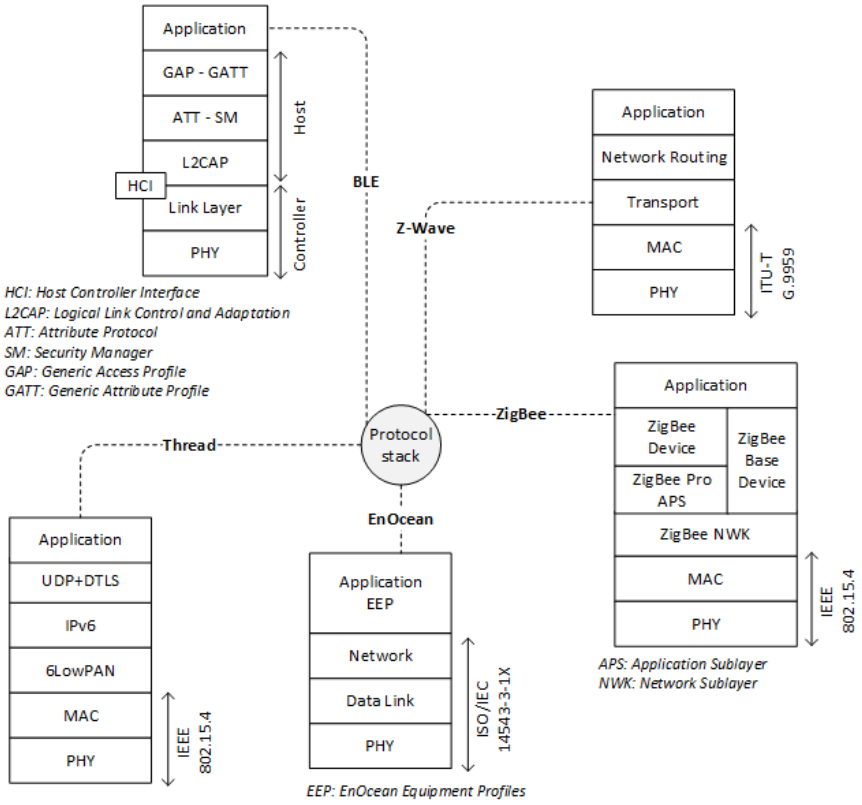

Figure 1. Overview of protocols stacks.

Table 1. Key security features per examined protocol (● = full support; ⊛ = exceptions exist).

Feature	Protocols				
	BLE	**ZigBee**	**Z-Wave S2**	**Thread**	**EnOcean**
Confidentiality	⊛ [1]	●	●	●	⊛ [5]
Msg. Authenticity & Integrity	●	●	●	●	⊛ [6]
Anti-replay	●	●	●	●	⊛ [6]
MITM protection	⊛ [2]	●	●	●	⊛ [7]
Device authentication	⊛ [3]	●	● [4]	●	● [8]

[1] Only for devices that support LE Privacy. [2] Only for devices that implement LE Secure connections. [3] Only with OOB or Numeric Comparison. [4] Not mandatory for the "S2 unauthenticated" class. [5] Only for R-ORG S telegrams. [6] Only for R-ORG S that contain a rolling code field. [7] Only for applications that conform to the EnOcean high security extension. [8] Via the teach-in procedure.

2.1. BLE

Bluetooth Low Energy (BLE) was introduced as part of Bluetooth 4.0 specification [7] to provide Bluetooth-like communication capabilities to devices with limited computational and energy resources, e.g., IoT devices. Similar to the classic Bluetooth, the communication follows the client–server (master–slave) model. The communication is primarily optimized for conserving energy, with the peripheral devices typically sending small amounts of data in bursty fashion. The maximum range of BLE is 50 m for the devices that implement the Bluetooth 4.0 specification and up to 800 m for Bluetooth 5 [8].

BLE operates in the 2.4 GHz frequency range and utilizes 40 communication channels. Since the specific band may be saturated by communicating devices using other protocols, BLE has adopted the adaptive frequency hopping mechanism as a means to deal with congestion and interference. To cope with channel congestion and interference, the devices do not statically use a single channel, but instead they hop from channel to channel staying there for a specific amount of time and transmit only one packet.

There are two types of packets, i.e., the advertising and the data packets. The advertising packets can be sent only through 3 out of the 40 channels. The Protocol Data Unit (PDU) of these packets contains a 16-bit header and 0–37 bytes payload. The rest of the 37 channels are utilized for data exchange purposes only. The Access Address has a value of *0x8e89bed6* (fixed) when a packet is a type of advertising, but that field has a variable value for data packets.

Accessing data is achieved through the Attribute (ATT) protocol. In the ATT terminology, data are referred to as attributes. To be more precise, an attribute is comprised of: (a) a handle, i.e., a number that uniquely identifies an attribute; (b) a type, i.e., what the attribute corresponds to and this is defined by a Universally Unique Identifier (UUID); (c) a value, i.e., the actual data; and (d) permissions, i.e., tags indicating whether the data can be only read, written or both. The attributes exist on the server, and the client can access these attributes. As an example, consider the heart rate profile that may exist on the slave, such as a smartwatch. The server on that slave device will offer the heart rate attribute. In this scenario, the smartphone, which will be used as a master, will request access to that attribute from the server component of the smartwatch. Lastly, the Generic Attribute (GATT) profile defines a grouping of services and attributes to facilitate attribute discovery and access.

BLE was designed to provide a secure communication channel to interacting parties while at the same time protecting the privacy of the end-users. The mechanisms adopted to achieve these design principles are: (a) the pairing and bonding; (b) traffic encryption; and (c) the device address randomization.

Pairing is a multi-phase process that takes place towards the establishment of a temporary trust (authentication) between the communicating devices. Pairing does not persist across connections. Bonding, on the other hand, is an optional sub-process which intends to extend the already established trust for a prolonged period. Once bonding has taken place, the first device pairing phases may be omitted. Thus, devices can connect much more quickly.

In Phase 1 of the pairing process, the exchange I/O capabilities along with the requirements for protection against Man-in-the-Middle (MITM) attacks take place. The capabilities determine which of the alternative pairing methods will be used in the next phase. Alternative values are No Input No Output, Display Only, Display Yes/No, Keyboard Only, and, finally, Keyboard Display. These messages are not encrypted.

In Phase 2, important cryptographic keys get created and exchanged. The type of keys heavily depends upon the version of the protocol. In Bluetooth 4.0 [7], the Temporary Key (TK) gets decided through one of the three alternative pairing modes, namely: (a) Just Works, in which the TK is set to 0 (this is not secure); (b) the PassKey Entry, in which the TK becomes equal to a 6-digit PIN registered in both devices; and (c) the Out of Band (OOB) by which the TK is exchanged using an alternative wireless technology, e.g., NFC. The TK is used to derive the Short Term Key (STK) according to the s1 function defined in the corresponding standard (Bluetooth Core Spec V4.2, Vol.3, Part H, Section 2.2.4 [9]. The STK is used to encrypt the communication subsequently. In Bluetooth 4.2 and above, a Long-Term Key (LTK) is generated instead of an STK. To do so, the two devices first exchange public keys to compute the Diffie–Hellman key (using the Elliptic Curve function P256) along with some random numbers. Although the traditional pairing modes are supported, Bluetooth 4.2 relies on a new pairing mode, namely the Numeric Comparison. This is similar to the Passkey Entry. However, the displayed number is not used for the generation of TK or any other key but rather for protection against MITM attacks. Finally, the Diffie–Hellman key, the device address, the random numbers, and the I/O capabilities of the device are used to generate the LTK. The LTK is used for link encryption instead of the STK.

In Phase 3, the STK (or LTK) encrypts the link, and various keys calculated in the previous phase get exchanged over that protected link. More specifically, the: (a) Identity Resolving Key (IRK), i.e., a 128-bit key that is used for creating and resolving random addresses; and (b) the Connection Signature Resolving Key (CSRK), i.e., a 128-bit key used for signing the traffic are sent. In versions lower than 4.2 the following are also sent: (a) the LTK; (b) the Encrypted Diversifier (EDIV); and (c) a random value (RAND). The last two keys are used for identifying the LTK in the device's internal database.

Encryption is performed using the AES-CCM cipher with an encryption key size of 128 bits. Communications get encrypted using the STK during the pairing process. The LTK is used to derive other keys, including the one used for encrypting traffic. To achieve message authentication, BLE signs packets with the CSRK. The method by which the signature gets calculated is given in Equation (1):

$$Sig = CMAC(CRSK, m||SigCntr, 64) \quad (1)$$

where $CMAC$ is a function for generating a Cipher-based Message Authentication Code utilizing the AES-128 cipher, m is the plaintext message, $SigCntr$ is a 4-byte counter that is initialized to 0 and is incremented for every message that is signed with the particular CRSK. The $SigCntr$ gets appended to the plaintext m (denoted as $||$). The output of the function is a 64-bit sequence (the length is provided as input to the function) that constitutes the signature. Devices that perform the signature verification maintain the last verified $SigCntr$ in the security database. During the subsequent verification, they compare that value with a received $SigCntr$ to check whether the received message was sent as part of a replay attack.

BLE devices advertise their presence frequently, and since they are typically mobile devices that may correspond to a signal, single owner privacy concerns arise. In further detail, the device's MAC address, which is usually a static identifier, can be used towards tracking devices and users. BLE has adopted a mechanism for randomizing the MAC addresses of devices so that a passive eavesdropper cannot follow a device after a certain period while at the same time authorized devices can resolve it. The standard identifies several MAC addresses: (a) Public Address, i.e., the real value of the device's MAC address as provided by the manufacturer; (b) Static Address, i.e., an address randomly generated after each power cycle which however remains static through this period; (c) Non-resolvable Private Address, i.e., a random address that cannot be resolved, thus needs to be communicated somehow

with a trusted party a priori; and (d) Resolvable Private Address, which is generated based on a shared secret according to a process that can be reversed only by parties who are in possession of that secret.

The method used to generate MAC addresses (Resolvable Private address) is given in Equation (2).

$$MAC = r||E_{24}IRK||(r||pad_{104}) \quad (2)$$

where r is a 24-bit random number, and E is an encryption function that uses IRK as the key and encrypts the random number r after it has been extended to 128 bits by adding 104 bits padding. The output of that function is truncated to 24 bits and then is concatenated with r to form a 48-bit MAC.

2.2. ZigBee

ZigBee is patronized by the synonymous Alliance [10] established in 2002, and enables the formation of a low-power, low data rate, wireless personal area network operating in unlicensed bands, i.e., mainly in the 2.4 GHz frequency band. The typical transmission range of ZigBee covers distances of 10–100 m with line-of-sight. The protocol's two bottom layers are based on the IEEE 802.15.4 standard [11], supporting three different network topologies, namely, star, tree, and—the more robust and resilient to failures—mesh. For creating multi-vendor interoperable products, the upper layers of the protocol stack, namely Network and Application, are described in the respective ZigBee Alliance standards, now in version 3.0. This newest version is built on the ZigBee PRO-2010 specification, which in turn is an enhanced version of the 2007 specification. Version 3.0 allows for increased interoperability (e.g., the same ZigBee 3.0 network is possible to accommodate ZigBee Home Automation, Health Care, and Light Link application profiles) as well as Internet connectivity, and thus comprises a full protocol stack by adding mesh networking and security layers along with an application framework.

ZigBee specifies three different logical device types, namely Coordinator (ZC), Router (ZR), and End Device (ZED), such as a motion sensor or a smart light bulb. The one per network ZC is in charge of bootstrapping and coordinating the network, and it may be used as a bridge to other networks. After the network is formed, the ZC and ZR never sleep and need to be continuously powered. The role of ZRs is to act as intermediate nodes between the ZC and the ZEDs and may allow for other ZRs and ZEDs to join the network. A ZED is unable to route any traffic and authorize other devices to join the network and can only communicate within the network through their parent nodes, typically ZRs. In addition, to conserve power, a ZED is able to sleep by entering the so-called low-power mode. ZCs and ZRs are also called ZigBee Full Function Devices (FFD), while a ZED is known as a Reduced Function Device (RFD). Each ZigBee network is identified by a 16- or 64-bit network ID. This number is assigned by the ZC upon creating the network. After that, each newcomer, either a ZR or ZED, is given a randomly assigned address from the device it is joining. Of course, such an address must be translated for enabling any sort of communication with devices located in an external IP network.

The commissioning procedure in ZigBee offers four different ways for a node to exchange network parameters, including the network identifier and the radio channel, and join a given network: (a) "Association" is done by means of 802.11.4 management packets; (b) "Rejoin", which employs ZigBee-oriented IEEE 802.11.4 data packets; (c) "(Re)join via orphaning" is done by means of ZigBee-oriented IEEE 802.11.4 management packets; and (d) "Out-of-Band", which is vendor-specific. The first method requires the network to be open to joining. After commissioning, and depending on the security policy, the device can be authenticated against the network.

IEEE 802.15.4 defines AES-CCM* 128-bit block cipher (which is a minor variation of CCM mode offering additional encryption capabilities) for protecting the data frames in transit and specifies the use of the security enabled flag in the frame control field and the optional Auxiliary Security Header field in the MAC frame structure; however, the standard does not specify key management procedures or entity authentication policies. These issues should be handled by the upper layers.

To this end, ZigBee security provisions are offered by the network and application layers. Specifically, ZigBee offers security services, including key establishment and transport, frame confidentiality and integrity (via the use of AES-CCM*), and replay protection. It has to be mentioned, however, that the ZigBee trust model implicitly applies solely between different devices, while the layers of the stack and the applications executed in the same device are assumed to trust each other. This means compromise of one application may straightforwardly enable the attacker to hijack any other application running on the same device.

A ZigBee network uses two symmetric 128-bit secret keys, namely the network and optionally the link or application key. The former is used to protect broadcasts, and thus it is shared between all the devices of a given network, while the latter protects unicast communications taking place at the application layer. The network key can be obtained in two ways. namely it can be pre-installed or received via key-transport, while the link key can be additionally obtained by a key establishment procedure. By default, frame encryption and authentication is provided by the network layer via the use of AES-CCM and the network key. If a link key is in place, and due to the aforementioned ZigBee's implicit trust model, the application layer may or may not enable two-tier security by means of the network and the link key. That is, if the network layer has already enabled security via the network key, then it is up to the application layer to provide additional security services via the link key. Optionally, there exists a special type of keys called "Master keys". That is, for two devices to generate link key they need to execute the key establishment procedure. In this stage, a master key serves as an initial shared secret among the devices and can be acquired by the respective device via pre-installation, user-entered data (including a QR, PIN, or password) or key-transport.

In ZigBee PRO and v.3 there exist two security modes/models, namely centralized and distributed, and subsequently a joining node embraces whichever security mode is used by the network it connects to. The first mode, uses a trust center (TC), typically a ZC, which coordinates the network, grants or disallows access to new devices, and administers the allocation of network and link keys to the joining nodes. The network key is distributed to the newly joined node encapsulated with a 16-byte pre-configured link key that is known to the node and the TC. After that, the TC may or may not provide a fresh link key to the node for securing TC-to-node communication and for the node to re-join the network if, e.g., the communication is lost. A major worry here is the type of the "pre-configured link key"; if it is a network-wide key, as in certain application profiles, then the network key depends solely on the secrecy of this single link key. In fact, this is the case with the home automation profile, which uses the "ZigBeeAlliance09" global pre-configured link key to authenticate (re)joining devices and distribute the network key. A solution to this shortcoming is offered by v.3. Even though it does not specify any application profile, but a base device behavior (BDB), which also allows for the use of a global pre-configured key for backward compatibility, it enables in addition the use of install codes or a link key establishment procedure.

Specifically, for a centralized security network, a random 6–16-byte (plus a 2-byte CRC) install code is programmed in the device during manufacturing and serves as an input to a hash function to produce a link key for that node. During commissioning, the same install code must be obtained by the TC using some out-of-band method in order for it to generate the same link key. Then, after commissioning, the network key is passed to the node encapsulated by the bilateral unique link key. It is implied that the install code should be only accessible to the legitimate owner of the device. Generally, if the same link key is used across a number of network transactions, e.g., rejoins, without rekeying, then the network key is at stake. Precisely, following a device cloning, and after issuing a rejoin request, the attacker may be able to obtain the network key via the use of a compromised link key. Thus, in cases where a pre-configured link key is used, automatic rejoining should be disabled by the TC, and only a user-instructed manual fresh join should be allowed.

If an application on a node wishes to enable secure peer-to-peer communications with another node on the same network, then the TC acts as a proxy to generate and distribute a separate link key to the involved nodes. Basically, given that the protection of the (global) network key is of utmost priority,

node-to-node communications involving sensitive, especially end-user data, should be protected by a corresponding link key. Lastly, the TC is in charge of periodically refreshing the network key, by broadcasting the new key (with a new sequence number) encapsulated with the old one, and after notifying all devices to activate the new key. This procedure may happen after a node join/leave event to provide backward/forward secrecy and defend against replay attacks. However, network key rolling should be done wisely, e.g., when the network commissioning is open or closed for joins, otherwise it may create unnecessary overhead. The simpler and less-secure distributed model assumes the existence of only ZR and ZED, so a ZR provides the network key to a joining node, either ZR or ZED. To do so, all joining ZR and ZED must be pre-configured with a link key that is used to encapsulate the network key prior to sending it to the newcomer node. Moreover, depending on the security model used, and as a first line of defense, the TC or the ZR can maintain a white list in terms of MAC addresses of the devices that are permitted to join, and thus authenticate themselves against the network.

A certificate-based device authentication and key establishment is also provisioned for some ZigBee application profiles, including smart-energy. For this, every node must carry a lightweight—as compared to X.509—Elliptic Curve Cryptography (ECC) based Qu–Vanstone "implicit certificate" issued by some trusted certification authority. If so, a link key to protect any communication between two nodes can be generated via an Elliptic Curve Menezes–Qu–Vanstone (ECMQV) key agreement scheme. Naturally, the downside of using certificates is that they require more memory and processing resources at the device, especially if the device needs to hold certificates issued from different authorities along with their root keys.

Regarding the security of the commissioning procedure, ZigBee optionally offers a user-friendly proximity-based method called "Touchlink". In this setting, the initiator, e.g., a remote control device, perceives that a peer (to be joined in the network) device is in proximity, and transfers to it the network parameters. However, for a device that has already joined the current network, but is about to be moved to a new network, the initiator may transmit to it a factory reset command. This may leave room for aggressors to hijack the device and move it to another network [12]. Thus, after a device is commissioned and joined a given network, it should ignore any Touchlink messages stemming from its network, or Touchlink should be disabled with an option to manually re-enable it if necessary by an authorized technician. Another option for commissioning, is the co-called "out-of-band". This method has the advantage that authentication and the transfer of network parameters and the (encapsulated) network key do not happen over the unprotected wireless medium. Such flexible methods include (a) Near Field Communication (NFC) commissioning, which however is susceptible to attackers equipped with an NFC reader; (b) QR code, which also suffers from the inherent weakness of NFC; and (c) Over-web, which assumes some kind of a registration phase in a web page and obtaining the required parameters online. On the downside, considering a large number of devices, this method does not scale and transposes the security concerns at the website's side.

2.3. Z-Wave

Z-Wave is a proprietary low-energy home automation protocol [13] developed in 2001. In 2012, the protocol's physical and MAC layers were specified as ITU-T recommendation G.9959 [14]. Z-Wave enables communications over a multi-hop mesh network of up to 232 nodes, which operates on the 800–900 MHz band with a physical range of up to 100 m. The protocol is supported by the Z-Wave Alliance, which provides certification of the respective products. For this, since 2016, there exists a public interoperability (application) layer to ensure that devices from different vendors are able to communicate and cooperate.

The early version of Z-Wave, namely *S0*, does not mandate encryption on the communication link, making it prone to various attacks such as eavesdropping, message manipulation and injection. Optionally, however, it can be secured by the AES-128-CCM authenticated encryption scheme. If this scheme is enabled, the respective keys are generated during the pairing phase having as input the

data sent to the nodes by the primary controller, which is in charge of managing the whole network. In the S0 version of the protocol, this phase is protected by a hard-coded default key consisted of zeros. This flaw allows an attacker to obtain the network key and consequently decrypt the traffic.

In 2016, the Z-Wave Alliance imposed stronger security called *S2*, for Z-Wave certified devices. This latest revision employs Elliptic Curve Diffie–Hellman (ECDH), namely Curve25519 with a public key length of 256 bits, for pairing proposes, thus making man-in-the-middle attacks practically powerless. Note however that specific implementations of Curve25519 may be prone to side-channel attacks, as demonstrated in [15].

All nodes in an S2 network segment use the same symmetric network key to communicate with each other. This is done because managing through the use of different (unicast) link keys among all nodes in a network would require much more resources and battery reserves, and would also demand an additional separate key for multicast communication. Thus, the ECDH shared key is used by S2 nodes to derive a temporary link key, which in turn allows the controller in charge to securely transfer one or more network keys to a newcomer node.

More specifically, S2 implementations partition logically a Z-Wave network into three distinct security classes, namely "S2 access control", "S2 authenticated", and "S2 unauthenticated", where each has a unique network key. This segmentation of keys, i.e., one per device group, provides enough guarantees that pieces of information collected from one network class cannot be used to decrypt the traffic of another class. Specifically, the first class pertains to access control devices that must be authenticated during inclusion, e.g., door locks, and therefore is considered the most trusted. The second class applies to all typical smart-home devices, e.g., light dimmers and thermal sensors. The last class specifically refers to all resource-constrained devices that may choose between being authenticated or not during inclusion. Controllers that are not able to authenticate a joining device, e.g., due to a limitation in their user interface, e.g., only a LED screen, may fall into this category. For achieving interoperability with S0 devices, an S0 network key may be distributed to S2 devices by using the temporary ECDH link key. In any case, such a key will be considered an S2 unauthenticated class key.

A joining device that is granted access by a controller to more than one of the aforementioned security classes would only accept incoming commands that are encrypted with the network key belonging to the most trusted of them. Note, however, that a network node may be granted access by a controller to a subset of the requested classes. A controller may control two different devices using, e.g., the S2 access control class key and S2 authenticated class key correspondingly. In this case, the S2 device authentication process provides guarantees to the corresponding controller that a newcomer is the legitimate physical device and not a rogue one. That is, for a new device to join the network, and depending on the user interface, a controller may require the entity that installs the device to enter a unique device-specific key, namely a sequence of decimal digits or a quick response (QR) code.

S2 is also based on AES-128-CCM mode for the provision of authenticated encryption, and AES-128-CMAC for key derivation functions. For protecting against key-leakage attacks stemming from predictable patterns in the frame payload as well as replay attacks, every transmitted frame is scrambled by the use of a 13-byte nonce before encryption. The nonce is automatically updated prior to each transmission. Finally, for cloud communications, S2 also offers the option to tunnel the Z-Wave traffic over IP by means of a transport layer security (TLS) channel.

Given the security enhancements provided by S2, the weakest link in the chain is all legacy devices which carry the S0 certification. As further discussed in Section 3.1, such devices are unlikely to be updated by the respective vendor, and therefore they constitute a security liability for the network as they remain vulnerable to a myriad of attacks.

2.4. Thread

Thread is a low-power wireless mesh networking protocol patronized by the "Thread Group" Alliance launched on July 2014 [16]. As with ZigBee, Thread operates on the 2.4 GHz ISM band and

relies on IEEE 802.15.4 radio standard for its MAC and physical layers. One of the strongest points of Thread is that it caters for seamless connectivity to any IP-based network, including Ethernet, Wi-Fi, and the Internet in general, without the help of a gateway. This is done thanks to the employment of an IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) layer [17,18], which allows IPv6 packets to be transferred back and forth over any IEEE 802.15.4 connection.

A Thread network comprises one or more border routers that provide connection either to other Thread or external networks; routers, which offer routing services to network devices; and end-devices. By default, a router does not sleep, but it can demote its capabilities for becoming an end-device. Therefore, an end-device may be router-eligible (REED) or not. A sleepy end-device is not able to relay messages on behalf of other devices, but it can communicate only via its parent router. In a Thread network partition, following an automatic process, a router device is elected among all the routers to be the routing leader or simply Leader. This device keeps a registry of the assigned router addresses, and decides which router-eligible end-device can be elevated to a router upon request. As with any router in a Thread network, a Leader is allowed to have end-device children.

As already mentioned, Thread's stack offers IPv6 addressing [19], meaning that devices support a unique local address, a domain unique address (in a Thread domain model), and one or more global unicast addresses, e.g., all local router multicast, mesh local multicast, etc. That is, the high-order n bits of an IPv6 address (prefix) designate the network, while the rest correspond to specific addresses in that network. Therefore, all the addresses in the same network are composed of the same first n bits. The network prefix is selected by the device starting the network. After that, each device employs its factory assigned 8-byte IEEE extended unique identifier (EUI) to derive its interface identifier as defined in [17]. Moreover, as per the IEEE 802.15.4 specification, the Leader is responsible for assigning a 2-byte short address, i.e., the high bits in the address field, to each router in the corresponding network partition. On the other hand, the 2-byte short address of an end-device is produced upon request, e.g., when joining the network, from the address of its parent router. As a result, duplicate address detection is enforced by Leaders for routers and by routers for end-devices. Note that the use of 2-byte short rather than 8-byte MAC addresses leads to smaller packets and consequently may optimize the network bandwidth. In any case, if a device lacks a short address, it must be addressed via its MAC address. For improved privacy and security, the specification explicitly forbids the use of EUI, with a potential exception for out-of-band commissioning. This means that after the network security credentials have been successfully obtained, a node must randomize its EUI.

The Thread network is designed to be resilient with no single point of failure, because, even in the case a Leader breaks, another router will be automatically elected to serve this role. This stands true even for sleepy end-devices; if such a device loses communication with its parent, it will choose another parent in a user-transparent manner. However, in the case where the single border router confronts a failure, the network will become unavailable.

A Thread network can accommodate up to 32 routers, which employ next-hop routing, where the link cost is calculated based on the received signal strength indicator (RSSI) of the adjacent nodes. The routing table is maintained by the Thread stack to ensure that all routers maintain connectivity to the rest of the routers in the network. Neighbor detection, establishment and configuration of secure radio links, maintenance and dissemination of link reliability data including routing costs, exchange of device capabilities information, and distribution of common configuration values such as the channel and the network ID (PAN ID) are done by means of the Mesh Link Establishment (MLE) protocol [20]. MLE messages, carried over single-hop unicast and multicast messages between routers, are transmitted encapsulated in UDP datagrams with the source and destination ports set to 19788. Multicast in Thread is based on the Multicast Protocol for Low-Power and Lossy Networks (MPL) [21], i.e., simple flooding. Lastly, above the IP layer, Thread uses the Constrained Application Protocol (CoAP) [22] over UDP. Specifically, CoAP is used by routers to exchange management messages and to configure mesh-local and multicast addresses on the devices in the network.

Given that Thread builds on top of the IEEE 802.15.4 two-layer stack, it also employs AES-CCM 128 bits for providing data frame confidentiality and integrity, where the latter service is applied to both the frame header and the payload. MAC frame replay protection is implemented by having each device maintaining an outgoing 4-byte frame counter, plus a separate frame incoming counter per neighbor. Separate outgoing/incoming 4-byte counters are used by MPL to monitor the freshness of messages transmitted by a given device. Every node exchanges such counters with its neighbors by means of MLE handshake. Similar to ZigBee, due to the exploitation of 802.15.4, key derivation and management as well as entity authentication are handled by the Thread layers. Specifically, during network creation, the Leader of the network creates randomly a symmetric key called master key (MK), which, as explained in the next paragraphs, is securely distributed to every joining node. The MK is associated to a 1-byte key index, which is computed from a 4-byte sequence number. Thread runtime communication for MAC and MLE layers is secured correspondingly by two separate 128-bit keys. These are derived from the 4-byte sequence number concatenated with the ASCII binary representation of the string "Thread" using the SHA-256 HMAC under the MK. From the result, the first 128 bits serve as the MAC key, while the rest as the MLE key. The master key, along with other security material and network parameters, including the PAN ID must be kept in the non-volatile memory of every network node. This allows for a node to rejoin the network without human intervention, e.g., after a reset. Upon the expiration of the default key rotation timer, i.e., 672 h, the sequence number is incremented by one, the KeyIndex value is updated, and a new pair of MAC and MLE keys are produced. When this happens, the outgoing MAC and MLE frame counters are reset to zero.

Overall, in addition to over-the-air message confidentiality, integrity, and authenticity, link layer security caters for access control, replay protection, and non-repudiation, thus blocking outsider attacks, including eavesdropping, impersonation, message spoofing, tampering, and injection, as well as IEEE 802.15.4 MAC generic attacks, as given in [23]. However, given that the master key is network-wide, thus compromise of any Thread device would expose it, additional security implemented at the application layer is also suggested. To this end, CoAP and/over DTLS protocol can also be exploited by applications and provide the basis for achieving secure end-to-end communications.

Device commissioning in Thread requires one device having the "Commissioner" role. This device may already participate in an existing Thread network (native commissioner) or be an external one (external commissioner), e.g., a mobile phone which is connected to a Wi-Fi, which in turn has an interface to the Thread's network Border router. To obtain the Commissioner role, which is unique across the whole network, the candidate Commissioner device needs to be first authenticated against the Leader. This phase is known as Petitioning and is done via a representative, i.e., the Border router for an external commissioner or the commissioner router for a native one. At the start of the Petitioning phase, and for achieving authentication, the Border router must learn the Commissioner's candidate credentials, namely, a pre-shared key for Commissioner (PSKc), which is derived via key stretching by a passphrase having a length between 6 and 255 bytes. This key can be either entered directly in the Border/Commissioner router or into any trusted Thread device and transferred to the router. In addition, a re-keying process on PSKc is possible by means of MAC protected management frames over CoAP. Next, on the basis of PSKc, the Commissioner candidate will trigger a DTLS handshake with the router. The router on behalf of the Commissioner candidate, will mediate with the Leader for appointing the Commissioner candidate to be the sole Commissioner/authenticator for future joiners. In the case of an external Commissioner, itself and the Border router will maintain via periodic keep-alive notifications the already established DTLS tunnel between them with the aim to secure the exchange of subsequent CoAP petitioning, management, and relay messages. In the same case, the Leader also notifies all routers about where to find the Commissioner. This is done implicitly, namely, by advertising the Border router acting on behalf of the Commissioner. Then, any potential Joiner router will be aware of where to proxy DTLS messages stemming from the Joiner.

On the other hand, the joining phase starts with an untrustworthy device, called the "Joiner", wishing to connect to the Thread network. In the beginning, the Joiner needs to locate and establish

a P2P connection with a router, namely the "Joiner router". Depending on the case, the latter entity may also be the native Commissioner (simplest case), the Border router, or some other router. Next, an authentication and key agreement (AKA) procedure between the Joiner and the Commissioner takes place. For instance, in the case of an external Commissioner, AKA messages must be relayed via the Joiner router and then the Border router, i.e., there exist three distinct connections: (a) Joiner-to-Joiner Router P2P; (b) Joiner Router to Border Router via the Thread network; and (c) Border Router to Commissioner via, e.g., a Wi-Fi. Recall, that the Border router maintains a DTLS connection to the external Commissioner. AKA comprises a DTLS handshake using a password-authenticated key exchange (PAKE) by "juggling" (J-PAKE) [24]. In fact, Thread employs an elliptic curve variant of J-PAKE based on the NIST P-256 elliptic curve. This mechanism combines ECDH for key agreement and Schnorr Non-interactive Zero-Knowledge Proof scheme [25] to: (a) authenticate the two peers based on a relatively low-entropy (8–16-character alphanumeric string) passphrase set by the manufacturer called pre-shared key for device (PSKd); and (b) produce a high-strength ephemeral shared secret called Key Encryption Key (KEK) between them. In several cases, the PKSd is printed on the Joiner as a QR code or serial number, which can be scanned by the Commissioner, e.g., a smartphone. Nevertheless, having this key printed on a sticker, e.g., underneath the device, it may make it prone to physical attacks. Note that the Commissioner authenticates the Joiner without letting it know the MK. This is done by the Joiner router, which securely distributes the MK encapsulated with KEK along with other network parameters to the Joiner. The KEK is delivered by the Commissioner to the Joiner router, and is strictly disposable, thus providing enhanced security. In the end, the Joiner terminates the secure Commissioning session and attaches itself to the Thread network. It should be pointed out that the work in [26] provides proof that J-PAKE is resistant to dictionary attacks either off- or on-line, offers forward secrecy, and session key independence. In addition, it is obvious that even if the adversary has access to PSKc and/or PSKd, it is almost infeasible to guess the ephemeral DTLS session keys and the KEK.

As already mentioned, Mesh Link Establishment (MLE) messages administer fundamental operations in Thread dynamic mesh networks, where the topology and the physical environment are subject to frequent changes. Specifically for security, MLE messages are used for disseminating security material and frame counters between the devices. Due to their importance, MLE messages enjoy confidentiality, authenticity and integrity, given that they are protected by AES-CCM as well, but with a different key. In addition, each device keeps two separate MLE counters, namely incoming and outgoing, for each of its neighbors. Naturally, MLE messages cannot be protected during network discovery and before a Joiner is commissioned and has obtained the security material. In fact, after successfully commissioned into the network, a Joiner would initiate a handshake to attach to a parent. This process comprises four messages, namely Parent Request, Parent Response, Child Request, and Child Response, and is resistant to replay attacks via the matching of a response with its corresponding request. This is done via the use of a specific Type Value Length (TLV) type called Challenge TLV, which is a randomly chosen byte string. Any subsequent, matching to this request, response message must contain the same byte string in a Response TLV. This replay protection however does not apply to MLE advertisement messages. In any case, Thread specifications put forward that pieces of data carried over unprotected MLE messages should not be considered for altering corresponding parameters acquired via secured messages.

The security of forwarding (by means of 6LoWPAN) and routing operations are addressed by the security services applied to the link and MLE layers. In this respect, before transmission, every router applies hop-by-hop encryption and integrity/authentication to 6LoWPAN headers and distance-vector information. This protects the network against legacy attacks in Mobile Ad Hoc Network (MANET), including message tampering, wormhole, black hole, etc. As already mentioned, MPL uses separate sequence numbers to track the freshness of messages transmitted by a given device. This message ordering scheme leaves room for DoS type of attacks, as described in [21]. Another security concern in MPL is the use of the Trickle algorithm [27], which regulates the transmission rate of a node to

the bare minimum needed to maintain the node internally consistent. That is, Trickle controls the send rate so as each node hears a slow stream ("trickle") of packets. In this respect, the security considerations pinpointed in [27], i.e., forcing timer resets to drive nodes to transmit packets at a much higher rate than needed, and preventing nodes from reaching consistency by suppressing transmissions with "consistent" messages, also apply to MPL, and as a result may affect Thread too. Overall, excluding insiders, the aforementioned attack options are not practical in Thread by virtue of the security provisions offered at the link layer. On the other hand, in the case of an insider or if a strong external adversary succeeds in compromising a router, then they can harvest network credentials, e.g., the master key, and form that point on elevate their attack to the network at will. In a more persistent attack scenario, the adversary who controls a router may selectively taint/pollute distance vector protocol information and 6LoWPAN headers.

2.5. Enocean

EnOcean technology invented by the homonymous company and patronized by the EnOcean Alliance is optimized for ultra-low power wireless battery-less "install and forget" devices and energy harvesting applications in building and home automation, and is standardized in ISO/IEC 14543-3-10 [28]. To achieve the aforementioned goals, the EnOcean's radio protocol (ERP) uses sub-GHz frequency bands, namely 315, 868.3, 902, and 928 MHz, depending on the legal regulations applying to each country. In addition, battery-less solutions exist employing the 2.4 GHz ISM band (IEEE 802.15.4) for enabling integration with ZigBee and BLE systems. The typical transmission range of the technology covers distances of up to 30 m in buildings and up to 100 m with line-of-sight, while the theoretical maximum data rate is 125 kbit/s. EnOcean supports the mesh network topology, where all nodes can communicate point-to-point with each other in their wireless range. EnOcean devices can also communicate with other networking realms, including TCP/IP-based, via the use of a gateway. The three lower layers of the EnOcean protocol stack are realized by means of the ISO/IEC 14543-3-1X standard, while the Alliance offers the EnOcean Equipment Profiles (EEP) application sub-layer for supporting interoperability across diverse vendors and products under this standard [29].

Telegrams (messages) in EnOcean are identified in terms of either the unique device's 32-bit chip ID or a base ID. The latter is set in a pseudo-randomized manner in terms of a modulo function of the chip ID, providing 65,536 possible base IDs in total. Two subsequent base IDs have a distance of 128. When the chip ID is used, which is the typical case, it also serves as a prevention measure against device duplication. On the downside, replacing a device for another becomes cumbersome because the teach-in procedure must be repeated against every controller. To overcome this problem, a base ID can be used, which if needed, can be altered via the serial interface too.

Regarding security services, the latest specification in v2.5 [29] denotes that it is implemented "on the OSI presentation layer of the ERP protocol stack", meaning at the network layer of the ISO/IEC 14543-3-10 standard. Generally, a telegram may be secured or not, and this information is included in the R-ORG 1-byte field, which identifies the type of message that is being communicated. Secure messages (R-ORG S) are identified by the codes 0×30, 0×31, 0×33, or 0×35. In an R-ORG S telegram, the data part of the message is always encrypted. In addition, the original message's R-ORG can optionally be encrypted after being concatenated with the data part. The option of not also encapsulating the original insecure R-ORG field of the message in the encrypted data field is to conserve energy at transmission time. As detailed in the following, EnOcean uses 128-bit AES for encryption done in 16-byte chunks, and Cipher Based Message Authentication Code (CMAC) based on AES for the authentication and integrity of R-ORG S telegrams. Note that the link layer of ISO/IEC 14543-3-10 standard also offers integrity protection via the "HASH" 4- or 8-bit field, but this is only for detecting transmission failures and not defending against malicious actors.

In addition to the aforementioned security services, a R-ORG S may contain a rolling code (RLC) field, which should be initialized to zero in every produced device, and afterwards change according to a predefined scheme. Precisely, per every P2P communication, RLC increases monotonically each time

a message is transmitted, and its value is checked by the receiver; if the value is not greater than the previous received RLC, the receiver drops the message. As already mentioned, the RLC field should be transmitted (explicit RLC strategy), but can be left out for saving energy (implicit RLC strategy). In the latter case, the two ends must keep this code in sync, that is, the RLC shall be in an RLC window range. Note however that the RLC is typically used as an input parameter to the calculation of CMAC and the encryption of the data field, therefore the receiver always indirectly checks the correctness of the RLC. Thus, if the RLC field is not transmitted, and the receiver observes a faulty, but always greater than the previous, RLC, it tries the next few RLCs upward in the window range. If no match is perceived, then the receiver returns to its local RLC value and drops the telegram. However, this scheme may lead to a deadlock if more messages than the predefined RLC window are missed by the receiving end. If so, the receiver needs to re-synchronize the RLC, typically via the teach-in procedure described in the following.

The encryption scheme used by EnOcean in called variable AES (VAES), meaning the use of the (variable) RLC value as an input to the AES procedure to increase its security. That is, the RLC is XOR-ed with the fixed AES initialization vector (IV) and the result is fed to AES. This way, the generated keystream is always different under the same secret key and until the RLC reaches its maximum value. As discussed in the following paragraph, RLC rollover may or may not be permitted. It is also implied that after changing the AES key, RLC must be set to zero. On the other hand, the CMAC operation is an encrypt-then-mac scheme done also in 16-byte chunks, more specifically it is executed after encryption over the data field, optionally the original R-ORG, and mandatorily the RLC, if the latter is used, but not transmitted with the telegram. This information is first XOR-ed with a sub-key derived from the secret key and the result is fed to AES to produce the final CMAC.

The details of secure communication between two peers are configured via a unidirectional or bidirectional teach-in procedure, where typically the device to be learned is placed in proximity to the receiver. Normally, the teach-in procedure is done via the wireless or serial interface. During this procedure the peers transmit to each other the encryption method (VAES, EnOcean High Security or no encryption), key, RLC presence, RLC size (16, 24 or 32 bits), RLC rollover flag, and CMAC size (no MAC or 3 or 4 bytes) that will be used during the operation mode. First off, the receiving device must be in learning mode to accept the teach-in messages from a transmitting peer. The latter sends the security teach-in message whenever its teach-in trigger is activated. The teach-in message carries all the aforementioned security parameters of the transmitting device, which are learned and stored by the receiver. The parameters pertaining to the cipher suite to be used are contained in the security level format (SLF) byte of the teach-in message. The specification urges that the AES key and RLC value should not be sent in the clear, but instead encapsulated by a pre-shared key (PSK) or by other means, e.g., using the serial interface. Note that the 128-bit PSK plus the SLF are normally printed on a sticker on the transmitting device, and thus can be entered or scanned (in the case of QR code or NFC tag) into the receiving device.

To defend against man-in-the-middle (MITM) attacks, the specifications [29] define the EnOcean high security extension. This extension is specially designed for applications that require a high safety level, such as door locks, and can be typically applied if at least one of the peers is line-powered. Precisely, this extension tackles scenarios where the active attacker eavesdrops on the communication and blocks the victim receiver. If successful, the aggressor is able to replay the communication at a later time. Note that such an attack is realistic because the receiver has not updated its local RLC value, and hence it will correctly parse and accept the replayed message. The remedy to this problem is offered via the use of a 32-bit random or incremental, non-repeating nonce as a challenge, and a timeout of 500 ms to guarantee time limitation bound to the specific nonce. Specifically, the communication can be authenticated in a unidirectional or bidirectional way. In the first case, the sender transmits a request for communication to the receiver and starts the timeout. The receiver generates a fresh nonce, transmits it to the sender, and starts the timeout. Now, the sender can transmit the data to the receiver in an authenticated way; either it incorporates the nonce along with the data in the generated

CMAC or, if RLC is not used, it includes the nonce as the IV in the VAES encryption phase. For any subsequent authenticated data transmission, the aforementioned process should be repeated, meaning the use of a fresh nonce.

3. Literature Review

Up to now, a significant mass of works has provided a review on the IoT security ecosystem. The paper at hand concentrates on contributions pertaining to the security of major WPAN protocols, intentionally neglecting more generic works on IoT security and others that analyze IoT frameworks from a security viewpoint (e.g., [6,30–38]). Precisely, the works considered by the present paper are organized in two groups. The first one concentrates on the security features offered by a specific wireless IoT protocol, e.g., BLE, ZigBee, or Z-Wave, and focuses on specific attacks and countermeasures. To ease the navigation through these works, we summarize them in Table 2. The second part provides a vis-à-vis or more discrete comparison of different protocols concentrating on one or more layers of the protocol's stack. In this second category of works, we also include contributions related to the security of the IEEE 802.15.4 standard, given that the 802.15.4 protocol stack is common to Thread and ZigBee protocols. More precisely, spanning a period from 2013 to 2019, the following subsections summarize in chronological order the most relevant works published either in scientific journals or conferences per each of the aforementioned groups. Note that, to the best of our knowledge, no specific work on the security of EnOcean exists so far, thus no respective discussion is included in Section 3.1, including Table 2.

Table 2. Summary of literature works per examined protocol (as discussed in Section 3.1).

Protocol	Published Works per Type of Attack
BLE	Key derivation [39–45] User tracking [46–49] Activity detection [46] Person identification [46–48] Replay attack [40,43,44] Advertisement spoofing [40,48] Exposed services [40,50] OTP authentication token interception [40] Cross-application tracking [51] Eavesdropping [41,44,46,49,52] Denial of Service [43,46] Downgrading [45]
Z-Wave	Eavesdropping [53] Replay [54] Key derivation [55] Rogue controller [56] Integrity vulnerabilities of the routing protocol, Black hole [57] Unauthorised commands [58]
ZigBee	Battery drain [59,60] Key sniffing [59] Key recovery through storage dump, Same-NONCE, Processor overloading (DoS) [61] Replay [61–63] Network discovery and device identification [12,62] Eavesdropping [62] Jamming [64] Device takeover [12,64–66] Reset device to factory settings, Permanent device disconnect, Network key extraction [12] Network and device reconfiguration [67]
Thread	Jamming and flooding, Handshake flooding, Network leave, Key compromise, Replay, Same-Nonce, Guaranteed Time Slot, PAN ID conflict, Acknowledgment, DoS, Back-off and Clear Channel Assessment manipulation, Repudiation [68] Electromagnetic side-channel, Key generation [69]

3.1. Group I

3.1.1. BLE

The work in [39] was one of the first to outline practical attacks against the BLE protocol. The author implemented a set of open-source tools, primarily a sniffer software based on the Ubertooth platform [70], which is capable of "following" BLE connections as they hop across frequencies. The process of predicting the next channel to hop is not trivial. While the state-of-the-art was capable of only monitoring newly established connections, the proposed tools can do so on pre-existing connections by recovering critical connection parameters. The authors showed that this is the first step towards other more malicious attacks, including the injection of packets as well as bypassing the BLE encryption.

The vast majority of fitness trackers fail to implement the address randomization part of the BLE specification. The authors of [46] conducted a study on BLE traffic between fitness trackers and smartphones. They pinpointed that the aforementioned implementation failure provides attackers with the capability for user tracking, activity detection, and person identification. For example, the traffic between devices has a direct correlation to the intensity levels of user activity. This implies that, by passively eavesdropping on a connection, an attacker can infer a user's activity type, e.g., walking, sitting, and running. The study showed that these issues exist on over 90% of five major fitness tracker manufacturers.

The authors of [47] conducted a large-scale study regarding the advertisements from 214 different types of BLE-equipped devices. They concluded that BLE devices leak private information that may be used for tracking, profiling, as well as fingerprinting of the end-users. More fearfully, some devices permit connections without a pre-existing relationship between the devices. The authors argued that the vast majority of existing solutions require a firmware update, which in real-life conditions is impractical. Their proposed system, coined BLE-Guardian, opportunistically uses reactive jamming to hide protected devices. It also adopts a mechanism for achieving device hiding, without interfering with the rest in the same channel.

The work in [40] introduces the tool gattacker. This tool is capable of operating in three alternative modes: (a) central mode, which gathers advertisements and other information transmitted by devices in the vicinity; (b) peripheral mode, in which an attacking device acts as a device emulator; and (c) data interception and manipulation mode, which is implemented using hook functions.

In [48], the authors outlined several inefficiencies of the BLE-based beacon services. Such are often used to provide a higher level of granularity to mobile user location inference, e.g., to the cm level. They rely on the analysis of the Received Signal Strength Indicator (RSSI) contained in BLE advertisement messages. Nevertheless, the lack of advertisement messages protection opens the door for: (a) beacon hijacking, i.e., to register beacons using beacon identifiers to be able to advertise the attacker's services instead; (b) user profiling, i.e., to register utilized beacon identifiers for the sake of tracking user's location and habits; (c) presence inference, i.e., track user movement based on static identifiers contained in the broadcasted messages; (d) beacon silencing, i.e., transmit a flood of other beacons to confuse the victim's receivers and render the beacon service unresponsive; and (e) user annoyance, i.e., deplete the power resources of a device at a higher pace.

An alternative attack is presented in [51]. As a first step, an installed application is configured to listen for advertising packets broadcasted by other BLE devices in the area. By doing so, the application can effectively derive a fingerprint of the victim user with the advertisements in the area. At a subsequent phase, the mobile application may share this database with other applications. Assuming that the victim operates in the same environment, the new application will be able to infer a link between the pseudonym of its user with the pseudonym of that user in the previous application thus, achieving cross-application tracking.

The contribution in [50] introduces a tool, namely the Profiler, which assesses the minimum level of security applied to data of BLE devices (not limited to normal operating conditions). Profiler is also

capable of checking for static PIN codes used during PassKey Entry mode, by executing a dictionary attack. The results show that at least one unauthenticated write could be performed in 83% of the tested devices. Using as a paradigm a BLE keyboard connected to a PC, the authors of [41] demonstrated that BLE devices provide no guarantees in regard to the security of the exchanged data.

A novel attack is proposed in [71], namely the BLE injection-free attack. Unlike the most popular attacks against BLE, the described attack does not rely upon packet injection or signal jamming, which automatically makes it more stealthy. Rather it takes advantage of the fact that BLE devices can only store a limited number of keys. The attacker attempts to establish a large number of connections with the target device. To be more efficient, they may rely on multiple BLE interfaces, or, alternatively, they may spoof various MAC addresses. Once the maximum number of keys supported by the device has been installed, whenever a new bonding request is received, the device will delete one of the keys to give its place to a new one. This situation effectively renders a device unable to communicate with the other party. While this is the most frequent scenario, other implementations handle this situation differently. For instance, instead of deleting keys, they downright deny connection to new users or allow insecure connections without bonding. The work in [42] details a MITM attack that can be launched against BLE. The authors showed that this vulnerability is due to the fact that a malicious entity can eavesdrop on the public keys of the devices during the initial phase of pairing.

The authors of [52] noticed that, when multiple applications are hosted under the same device (this is frequent in the mobile device ecosystem), it is possible for a malicious application to exploit the trusted relationship between the host and the device which was created by the benign authorized application. More fearfully, unauthorized applications may achieve this while requesting minimal permissions, thus operating more stealthily. Specifically, in the Android platform, the keys that are created in the pairing and bonding process are not associated between the BLE device and the mobile application, but rather between the BLE device and the operating system as a whole. Therefore, a malicious application can access the existing BLE data of other applications located in the same device without initiating pairing or reuse the connection of the legitimate mobile application and the BLE device. Moreover, the authors conducted a large-scale study that concluded that 45% of auxiliary mobile BLE applications do not implement measures to protect BLE data. Interestingly, this number rises to about 70% for medical BLE-powered applications.

Despite the address randomization mechanism, other static identifiers residing in payload of BLE advertisement packets, i.e., the UUID in combination with the lack of encryption, may lead to user and device fingerprinting. The research in [49] shows that this identifier might not only be obtained from the BLE traffic but also by analyzing the companion mobile apps' binary. An attacker that has the capability of scanning all applications in an app store can retrieve all possible UUIDs, and fingerprint all devices statically. At a subsequent stage, the attacker can scan for nearby advertisement packets to locate such devices based on the UUIDs. The experimental results indicated that 94.6% of the devices are fingerprintable by attackers, and unauthorized access can be performed in 7.4% of them.

The work in [43] demonstrates certain insecurities of BLE protocol by deploying a testbed architecture comprised of open-source software and hardware. Precisely, the authors demonstrated how encryption can be broken and well-known attacks such as DDoS and replay attacks can be executed. The same methodology has been followed by the work in [44]. Moreover, the authors of [45] studied the security of the BLE pairing process. By testing 18 different widely used BLE devices, they concluded that MITM and downgrading attacks are possible.

3.1.2. Z-Wave

The pioneering work in [53] offers a hardware and software implementation to eavesdrop on Z-Wave communications. Via this device, the authors detailed the encryption and data origin authentication security services provided at the application layer, and elaborated on a certain vulnerability in the key exchange protocol they found, which can be used by attackers who do not posses the encryption keys to remotely unlock doors.

The next year, the authors of [54], after implementing a generic wireless monitor/injector tool based on Software Defined Radio using GNU Radio and the well-known Python Scapy library, were able via a replay attack to prevent a Z-Wave alarm device from arming. The work in [55] concentrates on the key derivation strategy of Z-Wave. To solve the identified deficiencies, the authors introduced a re-keying and IV derivation scheme. In addition, the work in [72] introduces an open source tool for pen-testing Z-Wave networks and uses it to abuse fluorescent lamps.

The work in [56] capitalizes on a vulnerability to allow the placement of a rogue controller into the network, which allows building a hidden communication channel with any poorly-protected device. A more recent work on the security features of the same protocol [57] employs a real-world Z-Wave network and reverse engineered some critical networks aspects, including frame forwarding and topology management in an effort to identify intrinsic integrity vulnerabilities of the routing protocol. In addition, the authors managed to perform a black hole attack by taking advantage of the discovered vulnerabilities. Another work by some of the same authors [73] extends and experimentally evaluates a "Z-Wave Misuse-Based Intrusion Detection System (MBIDS)" originally proposed in [74].

The authors of [58] demonstrated that it is possible to send unauthorized commands to Z-Wave S2 certified products by forcing them to use the less secure Z-Wave S0 protocol and manipulating the underlying traffic. This procedure is allowed in order to keep backward compatibility with products supporting the Z-Wave S0 protocol.

3.1.3. ZigBee

The work in [59], after investigating certain ZigBee vulnerabilities, presents a couple of practical attacks against such devices. First, the authors demonstrate that it is possible to drain the battery resources of a device by constantly waking it up using a special signal. The other attack the authors investigated was feasible due to certain deficiencies in the key exchange process under the standard security level. The authors also proposed countermeasures to shield devices from these two kinds of attacks. The article in [75] contributes to the AES-CTR encryption scheme used by ZigBee. That is, the authors claimed that cryptographic operations with counter mode may be quite slow for a plethora of ZigBee devices, especially when time sensitive applications are involved. To remediate this issue, they proposed the use of ciphers based on chaotic functions. Moreover, the work in [61] offers an experimental evaluation of the ZigBee devices manufactured by a specific vendor. In their testbed, the authors considered smart-house applications, demonstrated a number of practical attacks, and proposed the corresponding remedies.

The authors of [62] exercised reconnaissance operations on a ZigBee communications for discovering all networks in proximity along with the participating devices and identified their security configuration or any other useful piece of information by eavesdropping on the network traffic. The authors also exercised a replay attack and proposed a number of countermeasures. The work in [64] elaborates on the security measures in ZigBee from a practical viewpoint, aiming at investigating the protocol's shortcomings. More interestingly, the authors implement a software framework which can be used by pen-testers to eavesdrop on ZigBee communication messages and inspect the security properties of encrypted networks. The tool, based on scapy-radio and killerbee, enables the tester to automatically execute certain network operations, including node leave or join operations, detecting insecure key transport, etc. In the same year, Cao et al. [60] introduced an energy depletion attack against ZigBee devices by exploiting certain vulnerabilities existing in the IEEE 802.15.4 standard, i.e., the MAC and physical layers of ZigBee. The authors demonstrated the impact of the attack, which ranges from DoS to replay, and propose measures to defend against it.

The work in [76] deals with the security of ZigBee Light Link (ZLL), which is nowadays very popular for smart lighting in smart-house ecosystems. The authors attempted to develop an extensible formal security model for the ZLL commissioning protocol, which enables the creation of a ZLL network from scratch and adding new nodes to it. In the course of their investigation, they realized that such an endeavor may be prone to misconstructions due to the existing implicit security assumptions

done to achieve the security goals described in the respective standards. The authors of [12] focused on the security of Touchlink commissioning procedure introduced in ZigBee 3.0 in December 2016, and they showed that it is insecure, i.e., it presents inherent flaws. This is done by performing a number of attacks against legacy ZigBee products via their own open-source penetration testing framework. Specifically, they showed that a passive attacker can extract key material from a distance of 130 m, while an active one is in position to hijack devices from distances of 190 m. The lesson learnt is that Touchlink commissioning should be discontinued given that even one Touchlink-enabled device jeopardizes the whole ZigBee network. In the same year, Ronen et al. [65] dredged up a critical vulnerability in ZigBee implementation: the Touchlink part of the ZLL protocol. The authors demonstrated that such a bug would enable the aggressor to obtain control over a great number, if not all, of the smart lamps of a city, and then turn the lights off or even entangle all hijacked smart devices in a DDoS attack.

In [66], the authors presented an attack based on Remote AT Commands, which enables a malicious user to reconfigure or disconnect IoT sensors from the network. For instance, an attacker can send a Remote AT Command to a connected sensor and force it to join a malicious network in order to capture the traffic between the sensor and the network.

More recently, the authors of [63] elaborated on the possibility for replay attacks in ZigBee and demonstrate via experiments that the frame counter used by ZigBee to defend against replay attacks will not suffice when the network uses the same network key. This is especially true when the network key is pre-configured in the devices and there is no provision of re-keying. To fix this shortcoming, the authors also propose the use of multiple network keys along with a key generating method. The work in [77] assesses several kinds of threats against ZigBee networks and proposes and evaluates a security framework destined to real-time network monitoring and detection of the respective attacks. Moreover, the authors of [78] focused on hardware device identity verification (fingerprinting) for ZigBee devices. Their scheme relies on a number of Distinct Native Attribute features extracted from selected signal responses in an effort to tell legitimate and rogue devices apart. The authors of [67] utilized remote AT Commands in a similar way as [66]. In this case, the attacker is able to change the data destination address, the node ID and the network (i.e., PAN) ID.

3.1.4. Thread

Thus far, the literature dedicated specifically to Thread protocol security is scarce. We identified the following two works published in 2018. The authors of [68] examined the security features of the Thread protocol, and, under this prism, introduced a taxonomy for the security assessment of building automation systems. They pinpointed several potential shortcomings of Thread, and they validated the results of their analysis by testing a number of attacks, including jamming, handshake flooding, network leave, and key compromise. They also proposed and assessed potential remedies for reducing the attack surface and resist certain attacks. The authors concluded that Thread is proved to be significantly more resilient to attacks than competitive non-IP solutions. The work in [69] conducts a side-channel vulnerability study in terms of differential electromagnetic analysis along with specific network mechanisms on OpenThread [79], an open-source implementation of the Thread stack. The goal of the authors was to manipulate the security material or to obtain the network credentials. After summarizing the attack vectors and vulnerabilities found, they concluded that the inherent security mechanisms of Thread make side-channel attacks considerably difficult. They also proposed a number of countermeasures to defeat this kind of attacks in future implementations.

3.2. Group II

The authors of [80] focused on two security issues of the IEEE 802.15.4 standard, namely the establishment of pairwise keys, which, as already pointed out, is unspecified and left to the upper layers, and the fact that broadcast keys are shared among multiple nodes. To mitigate these limitations, they proposed a pairwise key establishment scheme, and a practical and secure protocol

for authenticating broadcast frames. The work in [81] also highlights on the fact that the IEEE 802.15.4 standard depends on upper layers to utilize any security feature and profile they provide, including the generation and exchange of secret keys. In view of this observation, the authors introduced a security framework for the sake of proposing diverse ilks of security architectures, a scheme for bootstrapping IEEE 802.15.4 domain security, and a lightweight method to negotiate link layer secret keys between the network devices. The work in [82] focuses on the security features of various existing standards and protocols destined to the IoT ecosystem, including IEEE 803.15.4, 6LoWPAN, CoAP and DTLS. The authors also elaborated on open research issues and challenges for fueling future work in the area.

The authors of [83] provided an empirical security analysis of the "SmartThings" smart home programming platform, which amongst others supports ZigBee and Z-Wave. They performed static source code analysis of many applications and device handlers destined to this platform, and they identified several threats related to insufficiently undocumented features. They highlighted that more than half of the SmartApps offered by the respective store are over-privileged. To provide proofs regarding their findings, they mounted four different attacks ranging from stealing door lock codes to disabling the vacation mode of a smart house and triggering a fake fire alarm. In the same year, the work in [84] concentrates on system on a chip (SoC) devices, which are available for building IEEE 802.15.4 networks. Such devices typically incorporate an AES accelerator for supporting AES-CCM operations necessary for encryption and authentication of messages by the respective IoT protocol. Using the Atmel ATMega128RFA1 chip in a proof-of-concept implementation, the authors conducted side-channel power analysis to examine the possible leakages of AES accelerator, and show that this may lead to the recovery of the encryption key used by the protocol running on top of IEEE 802.15.4.

The work in [85] elaborates on the main security aspects of mainstream protocols and standards used in wireless sensor network (WSN) deployments, including IEEE 802.15.4, 6LoWPAN, and CoAP. Based on a generic WSN layered architecture, among others, the authors considered security threats and existing countermeasures on a per layer basis. The authors of [86] detailed on five wireless IoT protocols employed for home automation, namely KNX-RF, EnOcean, ZigBee, Z-Wave, and Thread. The security features of each protocol is succinctly described along with a side-by-side comparison based on several criteria. The authors of [87] surveyed the security features of BLE, LoRaWAN, ZigBee and Z-Wave protocols. They elaborated on specific shortcomings and vulnerabilities and showed how these issues have been addressed by the respective protocol throughout its development. In the same year, Dragomir et al. [88] focused on the security aspects of IoT protocols, including ZigBee and Thread, as specified by standardization bodies and industry alliances. In addition, the authors of [89] investigated smart home security under the prism of existing standards and mechanisms, including ZigBee, BLE, and Z-Wave. The authors categorize and analyze major security issues, and presented relevant threats along with the countermeasures employed by the current systems. As a side contribution, the authors offered a list of good practices for strengthening security in smart home ecosystems.

The work in [90] focuses on the ZigBee, Z-wave, and BLE protocols, and details on their security aspects and deficiencies as depicted by the relevant literature. In the same year, the authors of [91] elaborated on the security features of Z-Wave and Thread protocols in an effort to identify present and future security challenges. They referred to several types of attacks against these two protocols and detail on the ways such attacks can be addressed. After introducing a taxonomy of cyber-physical threats, the work in [92] examines a plethora of potential attacks in the smart home environment based on the presented taxonomy. The authors concentrated on both the attack vectors and the potential consequences on the systems and the occupants of such a house. They also provided a comprehensive review of existing defences along with a discussion on the open research challenges. The paper takes into consideration attack vectors pertaining to a great number of both wired and wireless protocols used in smart home realm, including Bluetooth, ZigBee, and Z-Wave.

4. Discussion

This section summarizes key issues stemming from both the presentation and analysis of each protocol of interest and the related literature.

(a) Protocols that support mesh networking infrastructure, i.e., ZigBee, Thread, EnOcean, and Z-Wave, are more robust in terms of connectivity and "service" availability in contrast to other architectures.

(b) The introduction of improved security mechanisms as a result of a new version of a protocol should not be considered a panacea. For instance, although Z-Wave S2 offers advanced security features over its predecessor, it can be still leapfrogged by evil-doers to eavesdrop on and/or tamper with certain communication messages.

(c) All the examined protocols cater for over-the-air confidentiality, source authentication, and integrity. Barring some exceptions [61–63], they also offer protection against replay attacks, which can lead to unauthorized control of nodes. These essential security properties for any wireless protocol are effective against external attacks, such as eavesdropping, impersonation, message spoofing, tampering, and injection, as well as MAC layer generic attacks. More interestingly, EnOcean provides a scheme to repel MITM attacks, but this can be only enabled in practice for line-powered devices. Not less important, some or all of the aforementioned security services are mandatory for devices/applications that require a high safety level, such as door locks, while less security-sensitive ones may not enable such services at all.

(d) While energy-preserving, the use of a network-wide key to protect communications between devices leaves more room to aggressors. Protocols mitigate this threat by means of either additional security offered at the application layer (e.g., DTLS in Thread), network segmentation which leads to segmentation of keys (e.g., security classes in Z-Wave), or EEC-based AKA procedures to secure the transfer of such a key to joining devices. The frequent change of network-wide keys can also help toward the same direction.

(e) Perhaps the most important weakness is related to devices having old certifications, for providing backward compatibility. That is, despite the fact that the protocols constantly evolve to cope with certain shortcomings, IoT devices are designed to be of long-lived and install-and-forget nature. Therefore, typically, their firmware is rarely updated by the respective vendor, and, therefore, until their withdrawal, are prone to several attacks.

(f) In addition to the employment of standardized technologies as the IEEE 802.15.4 and ITU-T G.9959 in wireless IoT protocol stacks, the use of standardized or well-established security mechanisms, such as the DTLS, CoaP, and J-PAKE protocols in Thread, is a big step towards more interoperable and robust, in terms of security, solutions. In any case, especially when it comes to security, the "not-invented-here" syndrome should be avoided because amongst others can easily lead to positive biases, which in turn may create falsely elevated expectations. On the downside, it is expected that, due to the use of ECC schemes (or more generally public key cryptography) as in DTLS, J-PAKE, ECMQV, and Curve25519, the respective IoT protocols are also susceptible to clogging/flooding attacks.

(g) At this point it has been made clear that the IoT ecosystem is highly fragmented in terms of adopted wireless communication technologies. This diversity stems mainly from the (i) computational resources and (ii) energy resources of the devices, as well as the (iii) bandwidth and (iv) security requirements of the application. Protocols such as 6LoWPAN attempt to bridge the interoperability gap at a higher layer. Nevertheless, such attempts introduce security issues of their own [93,94] and secondarily fail to cover the majority, let alone the totality, of alternative protocols. To this date, the problem of interoperability in a secure fashion in the IoT realm largely remains open with several works providing comprehensive descriptions on the issues as well as potential solutions [95–97].

(h) A large number of attacks is a product of misconfiguration or poor implementation decisions on behalf of device manufacturers. This phenomenon is particularly true in the case of BLE communications where such flaws allow the tracking of users through their mobile devices and potentially the full dismantle of the encryption services offered by the protocol.

(i) For enabling the reset of devices in an automatic manner, e.g., in the case of accidental malfunction or loss of power, some important pieces of data must be kept in the device's persistent, non-volatile storage. Note that storing such information in a centralized manner, e.g., in a trusted remote location, including the cloud, may be prone to backhaul link disconnection problems, and thus pivotal data should be stored in each network node. Such data include network information, security material, authentication and commissioning data, factory-default settings, and others, and therefore they must be sufficiently protected from compromise after a device is hijacked. Most of the protocols specifically address this issue for the case a device leaves the network (secure leaving). For instance, in Thread, this event is completed over CoAP management messages and starts when a device receives a network leave request by the Commissioner. Then, the device must delete all network security material from its persistent memory and transit itself into the uncommissioned state. In any case, the authenticity of leave messages should be guaranteed in order to prevent spoof leave attacks resulting in device isolation. In addition, certain countermeasures should be deployed against device theft incidents, namely a device is stolen and moved to another network where it can be manipulated. For instance, as explained in Section 2.2, ZigBee Touchlink commissioning may be susceptible to this threat.

(j) The use of more powerful devices acting as network coordinators (e.g., TC in ZigBee and Commissioner and Border router in Thread) can be of major help in applying and enforcing security policies in a centralized manner. As a first layer of defence, such devices can make use of white/black lists to enforce device authentication in terms of, e.g., MAC address, but they can also cater for fine-grained device authorization, e.g., if a device is allowed to rejoin the network, if a device is allowed to create a link key for P2P enabling P2P communication with another device, etc. On the negative side, such devices, including network gateways, are inherently alluring targets for attackers, and thus may create a single point of failure if their attack surface is not minimized.

(k) Closely related to the previous issue is that of device tamper protection. Naturally, this kind of defence is in many cases unrealistic to be offered by means of physical isolation, and thus it should be applied through the use of tamper-proof hardware. For the same reason, firmware updates need to be delivered in a secure manner because may also result in having the device compromised and/or be enslaved in a bot army [4,98].

(l) In cases where the WPAN network makes use of cloud services, then all nodes become susceptible to a plethora of Internet attacks if the connection to the cloud is not secure end-to-end.

(m) An interesting and timely kind of attacks against WPAN devices capitalizes on physical side-channel analysis techniques. Such techniques have been traditionally exploited for mounting attacks against cryptographic systems in general, but, as discussed in Section 3, they have been lately tested against certain WPAN protocols either for device fingerprinting [78] or for obtaining the network credentials [79,84]. A possible research direction could be the use of physical side-channel analysis for also defending intrusions against such devices. In fact, this idea is not new, as it has already been explored in recent works mainly for industrial and other kind of IoT devices [99–102].

(n) Inevitably, as the case with virtually any wireless technology, all the protocols discussed in the context of this work are prone to DoS attacks caused by radio jamming. This may result to loss of service due to—even in some cases unintentional—interference (recall that, among others, IEEE 802.11 also uses the 2.4 GHz band). Consider, for example, a door-lock

which is clogged with interference to block it from locking. Typically, this threat is mitigated by "frequency agility", including frequency hopping (e.g., BLE) or dynamic frequency selection and transmission power control (e.g., ZigBee) for migrating the network to a "quieter" channel, and raising alarms if the problem persists. Such a scheme may be implemented in a network manager/coordinator node as the case may be. The same threat, but for the upper layers, applies to key control frames, including beacons. Namely, the aggressor may flood the network controller/router with a surge of spurious beacon request messages send to a specific of several radio channels aiming to overload or paralyze the latter device.

5. Conclusions

Wireless IoT technologies and applications are in a constant state of growth with a plethora of use cases ranging from personal to industrial ecosystems. For instance, a 2019 report by On World states that "by 2024, global annual shipments of 802.15.4 mesh chipsets will reach 1 billion" [103]. This IoT boom is anticipated to be further spurred by the global roll-out of 5G mobile networks featuring low latency, extremely high bandwidth, and superior reliability as well as comprehensive (I)IoT connectivity. However, behind this technological and market evolution certain doubts exist as to the uncertainty and the potential for unintended consequences. Perhaps, the greatest concern has to do with the security aspects of such wireless technologies, given that they are inherently prone to a plethora of passive and active attacks, mainly due to the uncontrolled wireless medium they operate in.

The work at hand is the first to our knowledge to conduct a comprehensive and comparative review of the security features offered by the most commonly used or promising WPAN technologies today, namely BLE, ZigBee, Z-Wave, Thread, and EnOcean. The objective of the review is twofold: first, to present the state-of-the-art regarding the security aspects of each examined protocol, and, second, to offer a succinct, but all-encompassing, review of the works in the literature investigating certain security weaknesses per protocol. The current work can be used as a reference to anyone interested in obtaining a holistic view of the security requirements and potential pitfalls of such IoT protocols, and it is also expected to foster research efforts to the development of security-by-design solutions in the particular domain.

Author Contributions: All the authors have contributed equally to the various stages of this work, including conceptualisation, methodology, information gathering, discussion of the results, and writing of the original and revised versions of the manuscript. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding.

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

Abbreviations

The following abbreviations are used in this manuscript:

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
AES	Advanced Encryption Standard
AES-CTR	AES Counter mode of operation
AKA	Authentication & Key Agreement
ATT	Attribute (protocol)
BDB	ZigBee Base Device Behavior
BLE	Bluetooth Low Energy
CCM	Counter with CBC-MAC mode of operation
CMAC	Cipher-based Message Authentication Code
CoAP	Constrained Application Protocol
CRC	Cyclic Redundancy Check
CSRK	Connection Signature Resolving Key
DoS	Denial of Service
DTLS	Datagram Transport Layer Security protocol
ECC	Elliptic Curve Cryptography

ECDH	Elliptic Curve Diffie–Hellman
ECMQV	Elliptic Curve Manazes-Qu-Vanstone key agreement
EDIV	Encrypted Diversifier
EEP	EnOcean Equipment Profiles
ERP	EnOcean's Radio Protocol
EUI	IEEE Extended Unique Identifier
FFD	ZigBee Full Function Device
GATT	Generic Attribute
IEEE	The Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IRK	Identity Resolving Key
ISM	Industrial, Scientific and Medical radio spectrum
IV	Initialization Vector
KEK	Key Encryption Key
LoRaWAN	Long Range Wide Area Network
LTK	Long Term Key LTK
MAC	Message Authentication Code
MANET	Mobile Ad Hoc Network
MITM	Man-in-the Middle
MK	Master Key
MLE	Mesh Link Establishment protocol
MPL	Low-Power and Lossy Networks
NFC	Near Field Communication
OOB	Out of Band association model
OSI	Open Systems Interconnection model
P2P	Peer-to-Peer
PAKE	Password-Authenticated Key Exchange
PAN	Personal Area Network
PIN	Personal Identification Number
PSK	Pre-Shared Key
PDU	Protocol Data Unit
QR	Quick Response code
RAND	Random value
REED	Thread Router-Eligible Device
RFD	ZigBee Reduced Function Device
RLC	EnOcean's Rolling Code
RSSI	Received Signal Strength Indicator
SLF	EnOcean's Security Level Format
SoC	System on a Chip
STK	Short Term Key
TC	Trust Center
TK	Temporary Key
TLS	Transport Layer Security
TLV	Type Value Length
UUID	Universally Unique Identifier
VAES	Variable AES
WPAN	Wireless Personal Area Network
ZC	ZigBee Coordinator
ZED	ZigBee End-Device
ZR	ZigBee Router
ZZL	ZigBee Light Link

References

1. Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 27 February 2020).
2. Rizvi, S.; Orr, R.; Cox, A.; Ashokkumar, P.; Rizvi, M.R. Identifying the Attack Surface for IoT Network. *Internet Things* **2020**, 100162, doi:10.1016/j.iot.2020.100162. [CrossRef]
3. HaddadPajouh, H.; Dehghantanha, A.; Parizi, R.M.; Aledhari, M.; Karimipour, H. A survey on internet of things security: Requirements, challenges, and solutions. *Internet Things* **2019**, 100129, doi:10.1016/j.iot.2019.100129. [CrossRef]
4. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84, doi:10.1109/MC.2017.201. [CrossRef]

5. Geneiatakis, D.; Kounelis, I.; Neisse, R.; Nai-Fovino, I.; Steri, G.; Baldini, G. Security and privacy issues for an IoT based smart home. In Proceedings of the 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 22–26 May 2017; pp. 1292–1297, doi:10.23919/MIPRO.2017.7973622. [CrossRef]
6. Hassija, V.; Chamola, V.; Saxena, V.; Jain, D.; Goyal, P.; Sikdar, B. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access* **2019**, *7*, 82721–82743, doi:10.1109/ACCESS.2019.2924045. [CrossRef]
7. Bluetooth Core Specification v4.0. Available online: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=456433 (accessed on 3 February 2020).
8. Bluetooth Core Specification v5.2. Available online: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439 (accessed on 3 February 2020).
9. Specification of the Bluetooth System v4.2. Available online: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=441541 (accessed on 7 February 2020).
10. ZigBee Alliance. ZigBee. Available online: https://zigbeealliance.org/solution/zigbee/ (accessed on 12 December 2019).
11. IEEE. *IEEE Standard for Low-Rate Wireless Networks*; IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011); IEEE: Piscataway, NJ, USA, 2016; pp. 1–709, doi:10.1109/IEEESTD.2016.7460875. [CrossRef]
12. Morgner, P.; Mattejat, S.; Benenson, Z.; Müller, C.; Armknecht, F. Insecure to the touch: attacking ZigBee 3.0 via touchlink commissioning. In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Boston, MA, USA, 18–20 July 2017; pp. 230–240.
13. Silicon Labs. Z-Wave Specification Documents Search Page. Available online: https://www.silabs.com/search?q=specifications;page=1;x6=searchHeader;q6=Documents (accessed on 12 December 2019).
14. ITU-T. G.9959: Short Range Narrow-Band Digital Radiocommunication Transceivers—PHY, MAC, SAR and LLC Layer Specifications. Available online: https://www.itu.int/rec/T-REC-G.9959/en (accessed on 17 December 2019).
15. Genkin, D.; Valenta, L.; Yarom, Y. May the Fourth Be with You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 845–858. doi:10.1145/3133956.3134029. [CrossRef]
16. Thread Group. Thread Group Support. Available online: https://www.threadgroup.org/support#specifications (accessed on 12 December 2019).
17. Kushalnagar, N.; Montenegro, G.; Culler, D.E.; Hui, J.W. RFC 4944, Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Available online: https://tools.ietf.org/html/rfc4944 (accessed on 3 March 2020).
18. Thubert, P.; Hui, J.W. RFC 6282, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. Available online: https://tools.ietf.org/html/rfc6282 (accessed on 3 March 2020).
19. Hinden, R.M.; Deering, S.E. RFC 4291, IP Version 6 Addressing Architecture. Available online: https://tools.ietf.org/html/rfc4291 (accessed on 3 March 2020).
20. Kelsey, R. Mesh Link Establishment, Internet Draft. Available online: https://tools.ietf.org/html/draft-ietf-6lo-mesh-link-establishment-00 (accessed on 1 March 2020)
21. Hui, J.; Kelsey, R. RFC 7731, Multicast Protocol for Low-Power and Lossy Networks (MPL). Available online: https://tools.ietf.org/html/rfc7731 (accessed on 10 February 2020).
22. Shelby, Z.; Hartke, K.; Bormann, C. RFC 7252, The Constrained Application Protocol (CoAP). Available online: https://tools.ietf.org/html/rfc7252 (accessed on 10 February 2020).
23. Sastry, N.; Wagner, D. Security Considerations for IEEE 802.15.4 Networks. In Proceedings of the 3rd ACM Workshop on Wireless Security, Philadelphia, PA, USA, 1 October 2004; pp. 32–42.10.1145/1023646.1023654. [CrossRef]
24. Hao, F. RFC 8236, J-PAKE: Password-Authenticated Key Exchange by Juggling. Available online: https://tools.ietf.org/html/rfc8236 (accessed on 10 February 2020).
25. Hao, F. RFC 8235, Schnorr Non-interactive Zero-Knowledge Proof. Available online: https://tools.ietf.org/html/rfc8235 (accessed on 10 February 2020).
26. Abdalla, M.; Benhamouda, F.; MacKenzie, P. Security of the J-PAKE Password-Authenticated Key Exchange Protocol. In Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 17–21 May 2015; pp. 571–587, doi:10.1109/SP.2015.41. [CrossRef]

27. Levis, P.; Clausen, T.H. RFC 6206, The Trickle Algorithm. Available online: https://tools.ietf.org/html/rfc6206 (accessed on 10 February 2020).
28. ISO/IEC. ISO/IEC 14543-3-10:2012, Information Technology—Home Electronic Systems (HES) Architecture—Part 3–10: Wireless Short-Packet (WSP) Protocol Optimized for Energy Harvesting—Architecture and Lower Layer Protocols. Available online: http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/98/59865.html (accessed on 10 February 2020).
29. Oberhaching, E.G. EnOcean Technical Specifications. Available online: https://www.enocean-alliance.org/what-is-enocean/specifications/ (accessed on 12 December 2019).
30. Zhang, Z.K.; Cho, M.C.Y.; Wang, C.W.; Hsu, C.W.; Chen, C.K.; Shieh, S. IoT Security: Ongoing Challenges and Research Opportunities. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, Japan, 19 November 2014; pp. 230–234, ISSN: 2163-2871, doi:10.1109/SOCA.2014.58. [CrossRef]
31. Mahmoud, R.; Yousuf, T.; Aloul, F.; Zualkernan, I. Internet of things (IoT) security: Current status, challenges and prospective measures. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 16 December 2015; pp. 336–341.10.1109/ICITST.2015.7412116. [CrossRef]
32. Wurm, J.; Hoang, K.; Arias, O.; Sadeghi, A.R.; Jin, Y. Security analysis on consumer and industrial IoT devices. In Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Austin, TX, USA, 9 June 2016; pp. 519–524, ISSN: 2153-697X, doi:10.1109/ASPDAC.2016.7428064. [CrossRef]
33. Frustaci, M.; Pace, P.; Aloi, G.; Fortino, G. Evaluating Critical Security Issues of the IoT World: Present and Future Challenges. *IEEE Internet Things J.* **2018**, *5*, 2483–2495, doi:10.1109/JIOT.2017.2767291. [CrossRef]
34. Yang, Y.; Wu, L.; Yin, G.; Li, L.; Zhao, H. A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet Things J.* **2017**, *4*, 1250–1258, doi:10.1109/JIOT.2017.2694844. [CrossRef]
35. Alaba, F.A.; Othman, M.; Hashem, I.A.T.; Alotaibi, F. Internet of Things security: A survey. *J. Netw. Comput. Appl.* **2017**, *88*, 10–28, doi:10.1016/j.jnca.2017.04.002. [CrossRef]
36. Mosenia, A.; Jha, N.K. A Comprehensive Study of Security of Internet-of-Things. *IEEE Trans. Emerg. Top. Comput.* **2017**, *5*, 586–602, doi:10.1109/TETC.2016.2606384. [CrossRef]
37. Ammar, M.; Russello, G.; Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *J. Inf. Secur. Appl.* **2018**, *38*, 8–27, doi:10.1016/j.jisa.2017.11.002. [CrossRef]
38. Neshenko, N.; Bou-Harb, E.; Crichigno, J.; Kaddoum, G.; Ghani, N. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2702–2733, doi:10.1109/COMST.2019.2910750. [CrossRef]
39. Ryan, M. Bluetooth: With Low Energy comes Low Security. In Proceedings of the 7th Usenix Workshop on Offensive Technologies, Washington, DC, USA, 13 August 2013; p. 7.
40. Jasek, S. Gattacking Bluetooth Smart Devices. In Proceedings of the Black Hat USA Conference, 2016. Available online: http://gattack.io/whitepaper.pdf (accessed on 3 February 2020).
41. Willingham, T.; Henderson, C.; Kiel, B.; Haque, M.S.; Atkison, T. Testing Vulnerabilities in Bluetooth Low Energy. In Proceedings of the ACMSE 2018 Conference, Richmond, KY, USA, 29–31 March 2018. doi:10.1145/3190645.3190693. [CrossRef]
42. Gajbhiye, S.; Karmakar, S.; Sharma, M.; Sharma, S. Bluetooth Secure Simple Pairing with enhanced security level. *J. Inf. Secur. Appl.* **2019**, *44*, 170–183. doi:10.1016/j.jisa.2018.11.009. [CrossRef]
43. Sevier, S.; Tekeoglu, A. Analyzing the Security of Bluetooth Low Energy. In Proceedings of the 2019 International Conference on Electronics, Information, and Communication (ICEIC), Auckland, New Zealand, 22–25 January 2019; pp. 1–5, doi:10.23919/ELINFOCOM.2019.8706457. [CrossRef]
44. Pallavi, S.; Narayanan, V.A. An Overview of Practical Attacks on BLE Based IOT Devices and Their Security. In Proceedings of the 5th International Conference on Advanced Computing Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; pp. 694–698, doi:10.1109/ICACCS.2019.8728448. [CrossRef]
45. Zhang, Y.; Weng, J.; Dey, R.; Jin, Y.; Lin, Z.; Fu, X. On the (In)security of Bluetooth Low Energy One-Way Secure Connections only Mode. *arXiv* **2019**, arXiv:1908.10497.

46. Das, A.K.; Pathak, P.H.; Chuah, C.N.; Mohapatra, P. Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers. In Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications—HotMobile, St. Augustine, FL, USA, 23–24 February 2016; pp. 99–104, doi:10.1145/2873587.2873594. [CrossRef]
47. Fawaz, K.; Kim, K.H.; Shin, K.G. Protecting privacy of BLE device users. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 1205–1221.
48. Kolias, C.; Copi, L.; Zhang, F.; Stavrou, A. Breaking BLE beacons for fun but mostly profit. In Proceedings of the 10th European Workshop on Systems Security, Belgrade, Serbia, 23 April 2017; pp. 1–6.
49. Zuo, C.; Wen, H.; Lin, Z.; Zhang, Y. Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security—CCS '19, London, UK, 11–15 November 2019; pp. 1469–1483, doi:10.1145/3319535.3354240. [CrossRef]
50. Sivakumaran, P.; Blasco Alis, J. A Low Energy Profile: Analysing Characteristic Security on BLE Peripherals. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy—CODASPY'18, Tempe, AZ, USA, 19–21 March 2018; pp. 152–154, doi:10.1145/3176258.3176945. [CrossRef]
51. Korolova, A.; Sharma, V. Cross-App Tracking via Nearby Bluetooth Low Energy Devices. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, Tempe, AZ, USA, 19–21 March 2018; pp. 43–52, doi:10.1145/3176258.3176313. [CrossRef]
52. Sivakumaran, P.; Blasco, J. A Study of the Feasibility of Co-located App Attacks against BLE and a Large-Scale Analysis of the Current Application-Layer Security Landscape. In Proceedings of the 28th USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; p. 19.
53. Fouladi, B.; Ghanoun, S. Security evaluation of the Z-Wave wireless protocol. *Black Hat USA* **2013**, *24*, 1–2.
54. Picod, J.M.; Lebrun, A.; Demay, J.C. Bringing software defined radio to the penetration testing community. In Proceedings of the Black Hat USA Conference, Las Vegas, NV, USA, 2–7 August 2014.
55. Agosta, G.; Antonini, A.; Barenghi, A.; Galeri, D.; Pelosi, G. Cyber-security analysis and evaluation for smart home management solutions. In Proceedings of the 2015 International Carnahan Conference on Security Technology (ICCST), Taipei, Taiwan, 21–24 September 2015; pp. 1–6, ISSN: 2153-0742, doi:10.1109/CCST.2015.7389663. [CrossRef]
56. Fuller, J.D.; Ramsey, B.W. Rogue Z-Wave controllers: A persistent attack channel. In Proceedings of the 2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops), Clearwater Beach, FL, USA, 26–29 October 2015; pp. 734–741, doi:10.1109/LCNW.2015.7365922. [CrossRef]
57. Badenhop, C.W.; Graham, S.R.; Ramsey, B.W.; Mullins, B.E.; Mailloux, L.O. The Z-Wave routing protocol and its security implications. *Comput. Secur.* **2017**, *68*, 112–129, doi:10.1016/j.cose.2017.04.004. [CrossRef]
58. Munro, K.; Tierney, A. A Basic Z-Wave Hack Exposes Up to 100 Million Smart Home Devices. Available online: https://www.pentestpartners.com/security-blog/z-shave-exploiting-z-wave-downgrade-attacks/ (accessed on 3 February 2020).
59. Vidgren, N.; Haataja, K.; Patiño-Andres, J.L.; Ramírez-Sanchis, J.J.; Toivanen, P. Security Threats in ZigBee-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. In Proceedings of the 2013 46th Hawaii International Conference on System Sciences, Maui, HI, USA, 7–10 January 2013; pp. 5132–5138, doi:10.1109/HICSS.2013.475. [CrossRef]
60. Cao, X.; Shila, D.M.; Cheng, Y.; Yang, Z.; Zhou, Y.; Chen, J. Ghost-in-ZigBee: Energy Depletion Attack on ZigBee-Based Wireless Networks. *IEEE Internet Things J.* **2016**, *3*, 816–829, doi:10.1109/JIOT.2016.2516102. [CrossRef]
61. Ďurech, J.; Franeková, M. Security attacks to ZigBee technology and their practical realization. In Proceedings of the 2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 23–25 January 2014; pp. 345–349, doi:10.1109/SAMI.2014.6822436. [CrossRef]
62. Olawumi, O.; Haataja, K.; Asikainen, M.; Vidgren, N.; Toivanen, P. Three practical attacks against ZigBee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned. In Proceedings of the 2014 14th International Conference on Hybrid Intelligent Systems, Kuwait, 14–16 December 2014; pp. 199–206, doi:10.1109/HIS.2014.7086198. [CrossRef]
63. Farha, F.; Chen, H. Mitigating replay attacks with ZigBee solutions. *Netw. Secur.* **2018**, *2018*, 13–19, doi:10.1016/S1353-4858(18)30008-4. [CrossRef]

64. Zillner, T.; Strobl, S. ZigBee Exploited—The Good, the Bad and the Ugly, Black Hat 2015. Available online: https://www.blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf (accessed on 3 February 2020).
65. Ronen, E.; Shamir, A.; Weingarten, A.O.; O'Flynn, C. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017; pp. 195–212, doi:10.1109/SP.2017.14. [CrossRef]
66. Vaccari, I.; Cambiaso, E.; Aiello, M. Remotely Exploiting AT Command Attacks on ZigBee Networks. *Secur. Commun. Netw.* **2017**, *2017*, 1–9. [CrossRef]
67. Piracha, W.A.; Chowdhury, M.; Ray, B.; Rajasegarar, S.; Doss, R. Insider Attacks on ZigBee Based IoT Networks by Exploiting AT Commands. In Proceedings of the International Conference on Applications and Techniques in Information Security, Tamil Nadu, India, 22–24 November 2019; pp. 77–91.
68. Liu, Y.; Pang, Z.; Dán, G.; Lan, D.; Gong, S. A Taxonomy for the Security Assessment of IP-Based Building Automation Systems: The Case of Thread. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4113–4123, doi:10.1109/TII.2018.2844955. [CrossRef]
69. Dinu, D.; Kizhvatov, I. EM Analysis in the IoT Context: Lessons Learned from an Attack on Thread. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, 73–97, doi:10.13154/tches.v2018.i1.73-97. [CrossRef]
70. Project Ubertooth—An Open Source 2.4 GHz Wireless Development Platform Suitable for Bluetooth Experimentation. Available online: https://github.com/greatscottgadgets/ubertooth/ (accessed on 3 March 2020).
71. Santos, A.C.T.; Filho, J.L.S.; Silva, S.; Nigam, V.; Fonseca, I.E. BLE injection-free attack: A novel attack on bluetooth low energy devices. *J. Ambient. Intell. Humaniz. Comput.* **2019**, 1–11, doi:10.1007/s12652-019-01502-z. [CrossRef]
72. Hall, J. Breaking Bulbs Briskly by Bogus Broadcasts., Available online: https://shmoo.gitbook.io/2016-shmoocon-proceedings/one_track_mind/02_breaking_bulbs_briskly_by_bogus_broadcasts (accessed on 3 February 2020).
73. Fuller, J.D.; Ramsey, B.W.; Rice, M.J.; Pecarina, J.M. Misuse-based detection of Z-Wave network attacks. *Comput. Secur.* **2017**, *64*, 44–58, doi:10.1016/j.cose.2016.10.003. [CrossRef]
74. Fuller, J.D. *A Misuse-Based Intrusion Detection System for ITU-T G.9959 Wireless Networks;* Technical Report AFIT-ENG-MS-16-M-016; Air Force Institute of Technology Wright-Petterson: Wright-Patterson, OH, USA, 2016.
75. Bakhache, B.; Ghazal, J.M.; Assad, S.E. Improvement of the Security of ZigBee by a New Chaotic Algorithm. *IEEE Syst. J.* **2014**, *8*, 1024–1033, doi:10.1109/JSYST.2013.2246011. [CrossRef]
76. Müller, C.; Armknecht, F.; Benenson, Z.; Morgner, P. *On the Security of the ZigBee Light Link Touchlink Commissioning Procedure*; Gesellschaft fur Informatik e.V., Lecture Notes in Informatics (LNI): Bonn, Germany, 2016.
77. Rana, S.M.S.; Halim, M.A.; Kabir, M.H. Design and Implementation of a Security Improvement Framework of ZigBee Network for Intelligent Monitoring in IoT Platform. *Appl. Sci.* **2018**, *8*, 2305, doi:10.3390/app8112305. [CrossRef]
78. Rondeau, C.M.; Betances, J.A.; Temple, M.A. Securing ZigBee Commercial Communications Using Constellation Based Distinct Native Attribute Fingerprinting. *Secur. Commun. Netw.* **2018**, *2018*, doi:10.1155/2018/1489347. [CrossRef]
79. Google LLC. OpenThread. Available online: https://openthread.io/ (accessed on 12 December 2019).
80. Krentz, K.F.; Rafiee, H.; Meinel, C. 6LoWPAN Security: Adding Compromise Resilience to the 802.15.4 Security Sublayer. In Proceedings of the International Workshop on Adaptive Security, Zurich, Switzerland, 8–12 September 2013; doi:10.1145/2523501.2523502. [CrossRef]
81. Piro, G.; Boggia, G.; Grieco, L.A. A standard compliant security framework for IEEE 802.15.4 networks. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, South Korea, 6–8 March 2014; pp. 27–30, doi:10.1109/WF-IoT.2014.6803111. [CrossRef]
82. Granjal, J.; Monteiro, E.; Sá Silva, J. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1294–1312, doi:10.1109/COMST.2015.2388550. [CrossRef]

83. Fernandes, E.; Jung, J.; Prakash, A. Security Analysis of Emerging Smart Home Applications. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 636–654, ISSN: 2375-1207, doi:10.1109/SP.2016.44. [CrossRef]
84. O'Flynn, C.; Chen, Z. Power Analysis Attacks Against IEEE 802.15.4 Nodes. In *Constructive Side-Channel Analysis and Secure Design*; Standaert, F.X., Oswald, E., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 55–70, doi:10.1007/978-3-319-43283-0_4. [CrossRef]
85. Tomić, I.; McCann, J.A. A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols. *IEEE Internet Things J.* **2017**, *4*, 1910–1923, doi:10.1109/JIOT.2017.2749883. [CrossRef]
86. Marksteiner, S.; Jimenez, V.J.E.; Valiant, H.; Zeiner, H. An overview of wireless IoT protocol security in the smart home domain. In Proceedings of the 2017 Internet of Things Business Models, Users, and Networks, Copenhagen, Denmark, 23–24 November 2017; pp. 1–8, doi:10.1109/CTTE.2017.8260940. [CrossRef]
87. Krejci, R.; Hujnak, O.; Svepes, M. Security survey of the IoT wireless protocols. In Proceedings of the 2017 25th Telecommunication Forum (TEFOR), Belgrade, Serbia, 21–22 November 2017; pp. 1–4. doi:10.1109/FOR.2017.8249286. [CrossRef]
88. Dragomir, D.; Gheorghe, L.; Costea, S.; Radovici, A. A Survey on Secure Communication Protocols for IoT Systems. In Proceedings of the 2016 International Workshop on Secure Internet of Things (SIoT), Heraklion, Greece, 26–30 September 2016; pp. 47–62, doi:10.1109/SIoT.2016.012. [CrossRef]
89. Batalla, J.M.; Vasilakos, A.; Gajewski, M. Secure Smart Homes: Opportunities and Challenges. *ACM Comput. Surv.* **2017**, *50*, 75:1–75:32, doi:10.1145/3122816. [CrossRef]
90. Celebucki, D.; Lin, M.A.; Graham, S. A security evaluation of popular Internet of Things protocols for manufacturers. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 12–14 January 2018; pp. 1–6, ISSN: 2158-4001, doi:10.1109/ICCE.2018.8326099. [CrossRef]
91. Unwala, I.; Taqvi, Z.; Lu, J. IoT Security: ZWave and Thread. In Proceedings of the 2018 IEEE Green Technologies Conference (GreenTech), Austin, TX, USA, 4–6 April 2018; pp. 176–182. ISSN: 2166-5478, doi:10.1109/GreenTech.2018.00040. [CrossRef]
92. Heartfield, R.; Loukas, G.; Budimir, S.; Bezemskij, A.; Fontaine, J.R.J.; Filippoupolitis, A.; Roesch, E. A taxonomy of cyber-physical threats and impact in the smart home. *Comput. Secur.* **2018**, *78*, 398–428, doi:10.1016/j.cose.2018.07.011. [CrossRef]
93. Pongle, P.; Chavan, G. A survey: Attacks on RPL and 6LoWPAN in IoT. In Proceedings of the 2015 International Conference on Pervasive Computing (ICPC), Pune, India, 8–10 January 2015; pp. 1–6.
94. Hennebert, C.; Dos Santos, J. Security protocols and privacy issues into 6LoWPAN stack: A synthesis. *IEEE Internet Things J.* **2014**, *1*, 384–398. [CrossRef]
95. Noura, M.; Atiquzzaman, M.; Gaedke, M. Interoperability in internet of things: Taxonomies and open challenges. *Mob. Netw. Appl.* **2019**, *24*, 796–809. [CrossRef]
96. Di Martino, B.; Rak, M.; Ficco, M.; Esposito, A.; Maisto, S.; Nacchia, S. Internet of things reference architectures, security and interoperability: A survey. *Internet Things* **2018**, *1*, 99–112. [CrossRef]
97. Elkhodr, M.; Shahrestani, S.; Cheung, H. The internet of things: New interoperability, management and security challenges. *arXiv* **2016**, arXiv:1604.04824.
98. Kambourakis, G.; Kolias, C.; Stavrou, A. The Mirai botnet and the IoT Zombie Armies. In Proceedings of the MILCOM 2017—2017 IEEE Military Communications Conference (MILCOM), Baltimore, MD, USA, 23–25 October 2017; pp. 267–272, ISSN: 2155-7586, doi:10.1109/MILCOM.2017.8170867. [CrossRef]
99. Sayakkara, A.; Le-Khac, N.A.; Scanlon, M. A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. *Digit. Investig.* **2019**, *29*, 43–54, doi:10.1016/j.diin.2019.03.002. [CrossRef]
100. Khan, H.A.; Alam, M.; Zajic, A.; Prvulovic, M. Detailed tracking of program control flow using analog side-channel signals: A promise for IoT malware detection and a threat for many cryptographic implementations. In *Cyber Sensing*; International Society for Optics and Photonics: Bellingham, WA, USA, 2018; Volume 10630, p. 1063005, doi:10.1117/12.2304382. [CrossRef]
101. Khan, H.A.; Sehatbakhsh, N.; Nguyen, L.N.; Callan, R.L.; Yeredor, A.; Prvulovic, M.; Zajic, A. IDEA: Intrusion Detection through Electromagnetic-Signal Analysis for Critical Embedded and Cyber-Physical Systems. *IEEE Trans. Depend. Secur. Comput.* **2019**, doi:10.1109/TDSC.2019.2932736. [CrossRef]

102. Han, Y.; Etigowni, S.; Liu, H.; Zonouz, S.; Petropulu, A. Watch Me, but Don'T Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1095–1108, doi:10.1145/3133956.3134081. [CrossRef]
103. ON World Inc. 802.15.4 IoT Markets—ZigBee, Thread, 6LoWPAN, Wi-SUN and Others—A Market Dynamics Report (15th Edition). Available online: https://onworld.com/zigbee/index.htm (accessed on 20 January 2020).