IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices

Francesca Meneghello, *Student Member, IEEE*, Matteo Calore, Daniel Zucchetto, *Member, IEEE*, Michele Polese, *Student Member, IEEE*, and Andrea Zanella, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) is rapidly spreading, reaching a multitude of different domains, including personal health care, environmental monitoring, home automation, smart mobility, and Industry 4.0. As a consequence, more and more IoT devices are being deployed in a variety of public and private environments, progressively becoming common objects of everyday life. It is hence apparent that, in such a scenario, cybersecurity becomes critical to avoid threats like leakage of sensible information, denial of service (DoS) attacks, unauthorized network access, and so on. Unfortunately, many low-end IoT commercial products do not usually support strong security mechanisms, and can hence be target of—or even means for—a number of security attacks. The aim of this article is to provide a broad overview of the security risks in the IoT sector and to discuss some possible counteractions. To this end, after a general introduction to security in the IoT domain, we discuss the specific security mechanisms adopted by the most popular IoT communication protocols. Then, we report and analyze some of the attacks against real IoT devices reported in the literature, in order to point out the current security weaknesses of commercial IoT solutions and remark the importance of considering security as an integral part in the design of IoT systems. We conclude this article with a reasoned comparison of the considered IoT technologies with respect to a set of qualifying security attributes, namely integrity, anonymity, confidentiality, privacy, access control, authentication, authorization, resilience, self organization.**

***Index Terms*—Attacks, devices, Internet of Things (IoT), security.**

I. Introduction

The INTERNET of Things (IoT) is an emerging communication paradigm that aims at connecting different kinds of objects to the Internet, in order to harvest data generated by sensors, remotely control appliances and machines, monitor environments, vehicles, and buildings, and so on [1]. The number and variety of IoT devices have rapidly grown in the last years, with a prediction of over 50 billions devices connected to the Internet by 2020 [2]. Thanks to a plethora of new "smart" services and products, such as smart appliances, smart houses, smart watches, smart TVs, and so on, the IoT devices are quickly spreading in all environments, becoming everyday more pervasive. Moreover, many of such smart services require users to intentionally reveal some personal (and, sometimes, private) information in change for advanced and more personalized services. It is then clear that security and privacy should be of primary importance in the design of IoT technologies and services. Unfortunately, this is not the case for many IoT commercial products that are provided with inadequate, incomplete, or ill-designed security mechanisms.

In the last years, growing attention has been dedicated to the risks related to the use of simple IoT devices in services that have access to sensitive information or critical controls, such as, video recoding of private environments, real-time personal localization, health-monitoring, building accesses control, industrial processes, and traffic lights [3], [4]. Furthermore, some security attacks against commercial IoT devices have appeared in the mass media, contributing to raise public awareness of the security threats associated with the IoT world.

In order to make commercial IoT devices more resilient to cyber attacks, security should be taken into account right from the design stage of new products [5]. However, the wide heterogeneity of IoT devices hinders the development of well-established security-by-design methods for the IoT [6], [7]. The challenge is further complicated by the severe limits in terms of energy, communication, computation, and storage capabilities of many IoT devices. Such limits indeed prevent the possibility of adopting standard security mechanisms used in more traditional Internet-connected devices [8], and call for new solutions that, however, are not yet standardized.

Besides the technical aspects, it is also necessary to develop a *cybersecurity culture* among the IoT stakeholders, in particular manufacturers and final users. As a matter of fact, many IoT device manufacturers come from the market of low-cost sensors and actuators (e.g., home automation, lights control, video surveillance, and so on). Such devices were originally designed to work in isolated systems, for which the security threats are much more limited. As a consequence, many manufacturers do not possess a solid expertise in cybersecurity and may be unaware of the security risks associated with connecting their devices to a global network. Such a lack of know-how, together with the hectic approach to the design of new products and the need to compress costs and time-to-market have

Manuscript received January 24, 2019; revised April 21, 2019 and June 30, 2019; accepted July 31, 2019. Date of publication August 13, 2019; date of current version October 8, 2019. This work was supported by the POR FESR 2014-2020 Work Program of the Veneto Region (Action 1.1.4) through the project No.10066183 titled "Sistema domotico IoT integrato ad elevata sicurezza informatica per smart building." *(Corresponding author: Michele Polese.)*

The authors are with the Department of Information Engineering, University of Padova, 35131 Padua, Italy (e-mail: francesca.meneghello@dei.unipd.it; matteo.calore@dei.unipd.it; daniel.zucchetto@dei.unipd.it; michele.polese@dei.unipd.it; andrea.zanella@dei.unipd.it).

Digital Object Identifier 10.1109/JIOT.2019.2935189

led to the commercialization of IoT products where security is either neglected or treated as an afterthought [9]. In parallel, the final users are also not much educated in terms of security practices and often fail to implement even the most basic procedures to protect their devices as, e.g., changing the preinstalled password of the devices on first use. Such an underestimation of their role in protecting personal devices makes users themselves unaware and unintentional allies of possible attackers.

The aim of this article is hence to provide an up-to-date vision of the current IoT cybersecurity scenario, contributing to improve the awareness of the threats that IoT devices may represent. To this end, this article will discuss the origins of such threats and the possible counteractions. The problems related to cybersecurity in IoT systems have already been addressed by other works in the literature as, e.g., [6] and [10]–[15]. Unlike such papers, here we address the topic from a more practical perspective. After a quick introduction to the cybersecurity, we focus on the specific problems of the IoT domain, where devices may not even support basic features, such as random number generation or standard encryption routines. We hence consider the four communication protocols mostly used in commercial IoT devices, namely ZigBee, Bluetooth low energy (BLE), 6LoWPAN, and LoRaWAN. We briefly recall the security procedures supported by each protocol and, hence, analyze the attack surface, also reporting a series of real attacks against popular commercial IoT devices as examples of the risks associated with poorly designed security mechanisms. Furthermore, we describe the processing units, communication protocol, and cryptographic hardware and software used in some commercial IoT devices, to offer an idea of the solutions currently adopted in the market. This article can then be useful to readers and practitioners interested to grasp the more practical implications of IoT security. Furthermore, the comparative analysis presented at the end of this article reveals some gaps in the literature that call for further investigation and experimentation.

Fig. 1 provides a visual representation of this article organization, which is as follows. Sections II and III are introductory to the following analysis. More specifically, Section II recalls the main functionalities of an IoT system and the related security challenges, while Section III describes the main security algorithms and protocols considered in this article. Moreover, we will provide some details related to the implementations of security protocols in the chipsets commonly used in commercial IoT devices. Section IV is dedicated to the communication protocols for the IoT. We briefly recall the main characteristics of each technology, focusing in particular on its native security mechanisms. Then, we provide a critical analysis of the attack surface, i.e., of the possible vulnerabilities related to that protocol. In Section V, we report practical implementation details of widespread IoT solutions, discussing in particular some hardware aspect as the microcontroller and the connectivity modules which have a role in determining the security level of such devices. Finally, in Section VI, we provide a qualitative comparison of the devices considered in this article, and we conclude this article with some open research directions in Section VII.

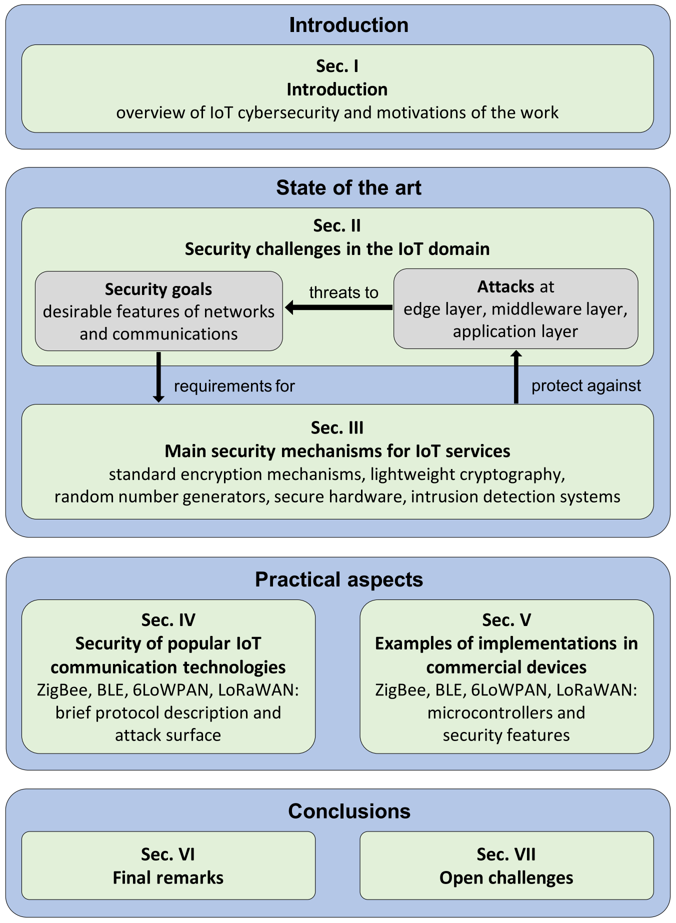
Introduction
Sec. I
Introduction
overview of IoT cybersecurity and motivations of the work
State of the art
Sec. II
Security challenges in the IoT domain
Security goals
desirable features of networks and communications
threats to
Attacks at
edge layer, middleware layer, application layer
requirements for
protect against
Sec. III
Main security mechanisms for IoT services
standard encryption mechanisms, lightweight cryptography, random number generators, secure hardware, intrusion detection systems
Practical aspects
Sec. IV
Security of popular IoT communication technologies
ZigBee, BLE, 6LoWPAN, LoRaWAN: brief protocol description and attack surface
Sec. V
Examples of implementations in commercial devices
ZigBee, BLE, 6LoWPAN, LoRaWAN: microcontrollers and security features
Conclusions
Sec. VI
Final remarks
Sec. VII
Open challenges

Fig. 1. Structure of this article and relations among the security aspects.

II. SECURITY CHALLENGES IN THE IoT DOMAIN

As discussed in the remainder of this article, the attacks against IoT devices are often simple and easy to conduct. They could be performed in order to break user privacy and leak personal sensible information. The collected data can indeed range from simple room temperature and humidity measurements, to more sensible information such as the heart-rate signal, or the user's location and living habits. Another common attack strategy consists in compromising one device in the IoT network and use it as a beachhead to perform fraudulent acts toward another network node [16].

In order to set a common ground for the discussion that will follow in the next sections, here we provide a broad overview of the IoT security requirements and of the related challenges.

A. Security Requirements

To begin with, we present a taxonomy of the security requirements for an IoT system with respect to the different operational levels, that is to say, at the *Information*, *Access*, and *Functional* level [17], [18].

Information Level: At this level, security should guarantee the following requirements.

1) *Integrity:* The received data should not been altered during the transmission.
2) *Anonymity:* The identity of the data source should remain hidden to third parties.

3) *Confidentiality:* Data cannot be read by third parties. A trustworthy relationship should be established between IoT devices in order to exchange protected information. Replicated messages must also be recognizable.
4) *Privacy:* The client's private information should not be disclosed during the data exchange. It must be hard to infer identifiable information by eavesdroppers.

Access Level: It specifies some security mechanisms to control the access to the network. More specifically, it provides the following functionalities.

1) *Access Control:* It guarantees that only legitimate users can access to the devices and the network for administrative tasks (e.g., remote reprogramming or control of the IoT devices and network).
2) *Authentication:* It checks whether a device has the right to access a network and whether a network has the right to connect the device. This is likely the first operation carried out by a node when it joins a new network [19]. Note that devices have to provide strong authentication procedures in order to avoid security threats. For example, if all the IoT devices produced by the same manufacturer are configured with the same authentication credentials, then the hacking of one device may compromise all of the security aspects at the information level.
3) *Authorization:* It ensures that only the authorized devices and the users get access to the network services or resources.

Functional Level: This level defines the security requirements in terms of the following criteria.

1) *Resilience:* It refers to network capacity to ensure security for its devices, even in case of attacks and failures.
2) *Self Organization:* It denotes the capability of an IoT system to adjust itself in order to remain operational even in case of failure of some parts due to occasional malfunctioning or malicious attacks.

B. Taxonomy of Security Attacks

Besides the requirements and mechanisms at the information, access, and functional levels, it is important to understand which are the vulnerabilities and the possible attacks at the different layers of the communication stack. As explained in [20], the communication architecture of an IoT system can be roughly divided in *Edge*, *Access*, and *Application* layers. The edge layer provides PHY and MAC functionalities for local communications. The access layer grants the connection to the rest of the world, usually through a gateway device and a *Middleware Layer* that acts as intermediary between the IoT world and the standard Internet. Finally, the *Application Layer* takes care of the service-level data communications. In the following we present a possible taxonomy of the attacks that can target these communication layers.

Edge Layer: One of the main threats at this level is represented by the *side channel attacks* [21]. The goal of these attacks is to leak information from the analysis of side signals, such as power consumption, electromagnetic emissions, and communication timing, while nodes are performing encryption procedures. Among them, the power consumption of the devices is widely exploited to guess and recover the encryption secret keys. For each encryption operation, a power trace can be captured: the power data is generally computed from the voltage difference across a resistor inserted in series with the power supply. Simple power analysis attacks try to directly interpret the power traces related to a small number of encryption rounds. Instead, the differential power analysis is a more effective and advanced approach: a bigger amount of traces are statistically analyzed in order to extract additional encryption information [22]. At the edge layer, IoT devices are also vulnerable to *hardware trojan* and DoS attacks that attempt to make resources unavailable to the legitimate users, e.g., by forcing the device to exit sleep (low-power consumption) mode in order to drain their batteries, or by jamming the radio communications. Also, the device package can be tampered with, e.g., to extract the cryptographic secrets of the device, modify its software to disguise a malicious node as a legacy one (*camouflage*), or attempt reverse engineering to figure out the details of proprietary communication protocols and possibly reserved information (as patent-covered algorithms).

Access/Middleware Layer: At this level the main attacks are *eavesdropping* (also called *sniffing*), injection of fraudulent packets and nonauthorized conversations. Even routing attacks have to be taken into account: an attacker may use this kind of attack to spoof, redirect, misdirect, or drop data packets.

Application Layer: Attacks at the Application Layer are quite different from the previous ones, since they directly target the software running on the devices rather than the communication technology. Such attacks may address the integrity of, e.g., machine learning algorithms, where the attacker manipulates the training process of the learning algorithm to induce misbehaviors. There can also be attacks on the login and authentication phases.

Fremantle and Scott [17] and Mosenia and Jha [23] presented an in-depth analysis of all these aspects, where they discuss some of the major vulnerabilities presented above, proposing solutions at different layers, from the device side to the cloud services.

In [24], the possible attacks against IoT devices are presented from a different standpoint, i.e., by considering how an attacker can exploit the IoT device for malicious purposes. The authors identify four possible approaches, as detailed below.

Ignoring the Functionality: This class includes all the attacks in which the specific functionalities of the IoT device are ignored, and only its capability to connect to the local area network (LAN) or to the Internet is exploited. For example, IoT devices can be used to create a *bot-net* (a network completely controlled by the attacker) or to penetrate the victim's home network and infect his/her computers.

Reducing the Functionality: In this case, the attacker tries to kill or limit the functionalities of the device, in order to annoy the victim or create malfunctions in a wider system. For example, this type of attack may be directed to IoT devices like smart TVs or smart refrigerators, with the aim of blocking or limiting their functioning in order to extort money from the victim for restoring their normal behavior.

Misusing the Functionality: The normal functionalities of the IoT devices are used to create discomfort to its owner. For example, an attacker may tamper a heating, ventilation, and air-conditioning (HVAC) control unit and make a certain environment uncomfortable by excessively increasing or decreasing the temperature. Similarly, the attack may target a smart light system, getting remote control over the lights in a room or building, overwriting the victims' commands.

Extending the Functionality: The IoT device is used to achieve completely different functionalities. For example, a presence sensor of an alarm system may be used to track the position of the victims in their living environment, even when the alarm system is off.

III. Main Security Mechanisms for IoT Services

In this section, we present standard security mechanisms that have been designed to satisfy the requirements described in the previous section.

Encryption: It is the main and most important operation to ensure confidentiality during the communication. It consists in changing the actual message (*plaintext*) into a different one (*ciphertext*) using a hash function that can be easily reverted only knowing a secret key. Using encryption, a possible eavesdropper can only have access to the ciphertext, but should not be able to interpret the content of the message. The encryption mechanism can be symmetric or asymmetric. In symmetric encryption, the same secret key is used both for message encryption and decryption, and hence it must be known by both the sender and the receiver. In the asymmetric case, each endpoint needs to possess its own pair of keys: a public key and the associated private key, which cannot be easily derived from the public one. The public key can be known to anyone, while the private key should be kept secret. The public and private keys are designed in a way that a message encrypted with the former can only be decrypted with the latter. Therefore, to guarantee confidentiality, the message is encrypted by the sender by using the public key of the receiver, which can then recover the original message by using its own private key.

Standard Encryption Mechanisms: The encryption process can be performed in two different ways: through a stream cipher, encrypting the plaintext bit-by-bit (or byte-by-byte) or with a block cipher, treating a block of plaintext as a whole and producing a block of ciphertext of equal length [25]. To encode long messages, block ciphers can be used in different operating modes: electronic CodeBook (ECB); cipher block chaining (CBC); cipher FeedBack (CFB); output FeedBack (OFB); and counter (see [26] for details).

One of the most used block cipher for symmetric encryption is the advanced encryption standard (AES) (or Rijndael), published in 2001 by the National Institute of Standards and Technology (NIST) [27]. AES is obtained through the cascade of N successive series of three elementary block ciphers: a substitution cipher, a transposition cipher, and a linear cipher. Fig. 2 shows the first three blocks of the chain, which are repeated to obtain the final encryption. Depending on the length (in number of bits) of the key, the algorithm takes the name of AES-128, AES-192, or AES-256.

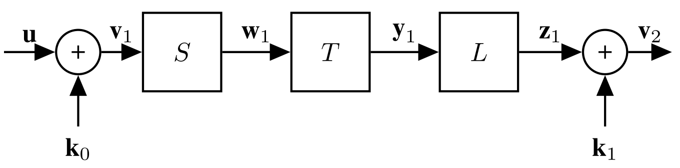

Fig. 2. Encryption for the AES block cipher. The three blocks, substitution (S), transposition (T), and linear (L) ciphers, are repeated to obtain the final encryption. In the diagram, $\mathbf{k_0}$ and $\mathbf{k_1}$ are the lower and upper parts of the encryption key, $\mathbf{u}$ denotes a block of plaintext to be encrypted, and the other symbols represent the block after each encryption module.

Examples of asymmetric cryptosystems are the Rivest Shamir Adleman (RSA) [28], the McEliece [29] and the Elgamal [30] algorithms.

Encryption can also be used to provide authentication and integrity protection, but in most cases these functionalities require additional mechanisms, in particular in IoT systems.

In practice, authentication and integrity protection are provided by means of message authentication codes (symmetric mechanisms), digital signatures (asymmetric mechanisms), and hash functions. For what concerns message authentication codes and digital signatures, the transmitted message is the result of the concatenation of the plaintext and a tag computed from the plaintext using the private key. At the receiver side, a tag is computed using the private or public key (depending on whether we opted for a symmetric or asymmetric process) and it is compared to the transmitted tag.

For example, many IoT services require to broadcast the same message to many destinations. If the message is encrypted by the transmitter, all the destinations need to decrypt the ciphertext to check the authenticity of the sender and retrieve the message. This operation requires time and drains energy from the device battery. If confidentiality is not an issue, a better solution may consist in sending the message in plaintext, attached with a tag that identifies the sender. The verification of the message authenticity can then be performed by one designated and trusted receiver, while all the other nodes can just read the message, without wasting time and resources [25].

The tag is usually obtained by using a good encryption function, such as AES, as in the cipher block chaining message authentication code (CBC-MAC) symmetric mechanism [31]. An asymmetric mechanism for tag computation is the digital signature algorithm (DSA), part of the ElGamal signatures family, published in 1991 by NIST with the federal information processing standard (FIPS) 186 [32], and revised multiple times in the following years. In 2009, for example, FIPS 186 included the elliptic curve DSA (ECDSA). An alternative asymmetric mechanism is RSA [28], which implements hash functions other than those of the previous two mechanisms. In fact, such hash functions map messages of any length into fixed length hash values, which are then transmitted with the messages. The hash functions are practically constructed through a Merkle–Damgard scheme [33], [34]. The most known functions are: MD-5 that produces a 128-bit hash, SHA-1 with a 160-bit hash, and SHA-256 and SHA-512 that produce 256 and 512-bit hash, respectively.

For what concerns IoT applications, the constrained application protocol (CoAP), defined by IETF in RFC 7252 [35], recommends to use the AES Counter with CBC-MAC, which is compactly indicated as AES-CCM. This mechanisms makes use of 128 bit keys and generate 8 bit authentication tags. The ephemeral elliptic curve Diffie Hellman (ECDHE) method is instead recommended for key establishment, and the ECDSA for authentication.

Lightweight Cryptography: Given the growth of the number of connected, low-complexity IoT devices, the research community has tried to design specific security algorithms for resource and energy constrained devices. Lightweight cryptography is a new branch of cryptography that focuses on these aspects, including new encryption block and stream ciphers, message authentication codes, and hash functions, which are conceived to be executed by devices with limited computation, communication, and storage capabilities. In 2012 the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) published the ISO/IEC 29192 standard that specifies a series of lightweight encryption mechanisms [36], included the block ciphers PRESENT [17] and CLEIFA [18]. PRINCE is another lightweight block cipher, not included in the standard [37]. Moreover, the Simon and Speck families of lightweight block cipher were presented by Beaulieu *et al.* [38]. As lightweight hash function, ISO/IEC 29192 standard proposed PHOTON [39] and SPONGENT [40]. In 2013 NIST started a lightweight cryptography project to investigate and develop solutions for real-world applications. At the beginning of 2019 NIST has published a call for algorithms for lightweight cryptography: after discussion and evaluation, the algorithms will go through a standardization process [41].

Random Number Generators: An important aspect for security is the randomness: security protocols frequently require the generation of (pseudo)random numbers for different purposes as, e.g., to create nonces during the authentication phase, to avoid replay attacks, and to generate asymmetric keys [42]. A random number generator is cryptographically secure when it produces a sequence for which no algorithm can predict in polynomial time the next bit of the sequence from the previous bits, with a probability significantly greater than $(1/2)$. According to Shannon's mathematical theory of communication, the entropy of a k-bit long (pseudo)random sequence must be as close as possible to k.

Two types of random number generators are commonly used for cryptographic applications: 1) the true random number generator (TRNG) that exploits physical noise sources and 2) the pseudo random number generator (PRNG) that expands a relatively short key into a long sequence of seemingly random bits, using a deterministic algorithm. PRNGs are typically used in real applications and technologies. In this case, since the adopted algorithms are usually known, the seed of the pseudorandom generator is the only source of randomness and, as such, it must be properly selected. A common way to generate random seeds is by exploiting different physical phenomena, such as the timing of user processes, or the thermal noise measured by the radio receiver [43]. For example, a source of random seeds may be obtained by feeding the noise signal of the radio power amplifier into the quantizer, which will then generate a stream of (ideally) independent bits. However, due to bandwidth restrictions, temperature bias, and other unavoidable factors, the bitstream may show a certain level of correlation.

Unfortunately, most of the source of randomness available in laptops and desktop PCs are not available in low-end embedded systems, such as the devices that will be analyzed in the following section. For this reason, the research has recently addressed the challenge of designing lightweight PRNG algorithms for resource constrained devices [44]–[46].

Secure Hardware: As discussed in the previous section, IoT devices are vulnerable to edge layer attacks. Most of the devices can be deployed in remote areas with a low level of protection so that an illegitimate user can perform side channel attacks. Several countermeasures have been proposed in the literature, based on the different encryption schemes. It is possible to exploit both hardware and software solutions to eliminate or, at least, randomize the signals footprint exploited by this type of attacks.

For example, physically unclonable functions (PUFs) can be adopted to improve hardware security [47]. The basic concept of PUF is to exploit little differences introduced by the fabrication process of the chip to generate a unique signature of each device. A PUF circuit provides a response to a given input challenge and, due to the intrinsic hardware differences, the responses are chip specific. As an example, an *Arbiter* PUF circuit is composed of two supposedly identical paths: for each input, the output depends on the fastest path [48]. Majzoobi *et al.* [49] proposed the lightweight secure PUFs concept, in which the response generation is resistant against reverse engineering attacks that try to emulate the PUF by parametrically modeling its behavior.

PUFs can be categorized into *strong* and *weak* [50]. If a PUF can support a number of challenge-response pairs that are exponential in the number of challenge bits, it is called strong PUF. Strong PUFs are typically used for authentication protocols that require new pairs for each operation. Arbiter PUF belongs to this category. On the other hand, weak PUFs can support a small number of challenge-response pairs and they are used for cryptographic key generation, avoiding the need to store secure keys on the devices. An example of weak PUF is the ring-oscillator described in [51].

Other hardware solutions to prevent side channel analysis attacks can be found in the literature: a hardware implementation of the SIMON algorithm is presented in [21], while a method to randomize the instruction execution cycles is shown in [52]. A software countermeasure is reported in [53] based on the randomization of a parameter used in RSA signature.

Anyway, all these techniques have some drawbacks, as the increase of power consumption of the device and the increase of the chip area. Because of the resource constraints of the IoT edge devices, it could be very hard to find effective solutions.

Intrusion Detection Systems: As discussed above, different security mechanisms have been proposed to protect the devices against threats at the different layers. However, besides preventing the attacks, it is also fundamental to be able to detect ongoing attacks. Complex anti-virus software and traffic analyzers cannot be used in IoT devices, due to resource and energy constraints. For this, lightweight intrusion detection methods have been presented in the last years [54]. For example, anomalies in system parameters, like CPU usage, memory consumption, and network throughput, may be indicative of an ongoing attack [55]. A similar approach is proposed in [56]: the energy profile is analyzed to detect anomalies in power consumption, which are linked to different types of attacks. In [57], the signatures of various attacks are derived from relevant features like packet dropping/send rate and signal strength intensity. Comparing the traffic pattern with these signatures, the attacks can be detected with good probability. Machine learning can also be exploited for intrusion detection purposes. In [58], for example, a random forest classification algorithm is used to group the traffic flows into different categories, based on some selected features. An attack is detected when some flows exhibit nonstandard patterns and are hence classified as anomalous.

IV. Security of Popular IoT Communication Technologies

As discussed in the previous sections, most of the legacy security protocols used in the standard Internet cannot be plainly applied to the IoT scenario, because of the constraints of many IoT devices in terms of computational, power, and communication capabilities [59]. Therefore, new mechanisms specifically designed for the IoT scenario have been proposed in the literature and implemented in some commercial IoT systems.

In this section, we focus on the security mechanisms implemented by some of the most popular transmission technologies used in the IoT domain, namely ZigBee, BLE, 6LoWPAN, and LoRaWAN. Moreover, we review the security vulnerabilities of these technologies, reporting different attack vectors found in the literature.

Among these technologies, ZigBee, BLE, and 6LoWPAN are predominantly used for short-range communications in homes or small offices. LoRaWAN instead is used for long-range scenarios, such as city-wide monitoring and control applications. From a protocol stack perspective, while ZigBee and BLE are full-stack technologies, 6LoWPAN and LoRaWAN cover only some layers of the stack, and therefore can be potentially used in the most diverse applications.

A. ZigBee

Description: ZigBee [60] is a two-way, wireless communication standard developed by the ZigBee Alliance. Thanks to its low cost and low power consumption, ZigBee is one of the most used technology to connect IoT devices. As shown in Fig. 3, the standard specifies the application and network layers (NWK), while the link and physical layers are taken from the IEEE 802.15.4 standard [61]. In more detail, the

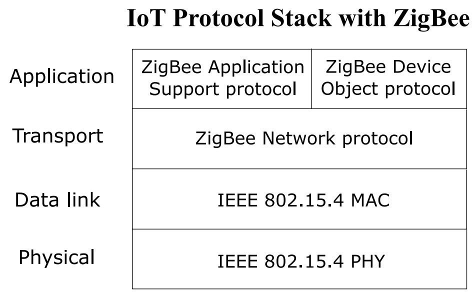
IoT Protocol Stack with ZigBee
Application
ZigBee Application Support protocol
ZigBee Device Object protocol
Transport
ZigBee Network protocol
Data link
IEEE 802.15.4 MAC
Physical
IEEE 802.15.4 PHY

Fig. 3. ZigBee protocol stack. The technical specifications for ZigBee can be found in [60].

Application Layer (APL) provides the data transmission and security services, and permits to bind the devices to two or more application entities located on the same network. The NWK provides functionalities such as routing, security, and configuration of new devices. The NWK also manages the establishment of new connections, the joining and leaving procedures, and the addressing and neighbor discovery services.

ZigBee includes different *application profiles* that define message formats and functional procedures to guarantee vendors interoperability [60].

Communications, secrecy, and authentication services are provided by message encryption and authentication, using AES in the *counter with CBC-MAC* (CCM) mode. Integrity protection is ensured by a 128 bit message integrity code (MIC) and replay protection is based on a 4 Byte frame counter.

Each ZigBee network includes a *Trust Center*, i.e., a device trusted by all the other nodes in the network. The trust center usually corresponds to the network coordinator and is responsible for 1) authenticating the devices that require to join the network; 2) deciding whether to accept or deny the join request; 3) maintaining and distributing network keys; and 4) enabling end-to-end security between devices.

The ZigBee network can either have a star topology, where the end-devices are directly connected to the coordinator, or a tree topology, when the interconnection is performed by intermediate routers. By interconnecting the routers, furthermore, it is possible to realize a mesh topology, as shown in Fig. 4.

The cryptographic routines used in ZigBee employ two 128 bit keys: the *link key* and the *network key*. The link key is used to secure unicast communications between APL entities. Each unicast communication uses a different link key, which is disclosed to the two linked entities only. The network key is needed for broadcast communications: it is shared among all the devices in the same network.

There are different ways for a device to acquire the required link or network key [62].

1) *Preinstallation:* The link or network key is installed in the device during the manufacturing process.
2) *Key Transport:* The link or network key is generated elsewhere (usually by the Trust Center) and then communicated to the device. The standard suggests to load the key using an out-of-band technique, however, it

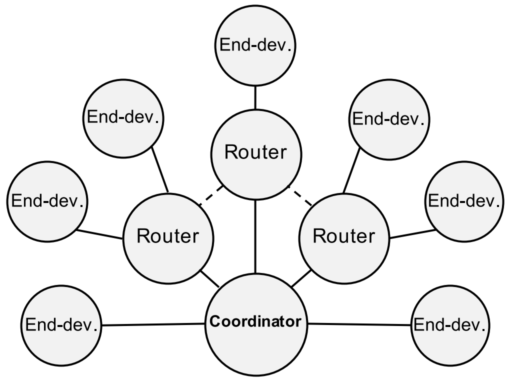

Fig. 4. ZigBee mesh topology.

includes the possibility to send the key in-band. In the latter case, the key may be sent in clear text or encrypted using a preshared key specific for each application profile. For example, for home automation devices, the preshared key is defined in the ZigBee standard and is publicly available. For ZigBee light link (ZLL) devices, instead, the preshared key *will be distributed only to certified manufacturers and is bound with a safekeeping contract*, according to the ZLL specification [63]. However, it has been leaked on the Internet in 2015, so it is now publicly known [62], [64].

3) *Key Establishment:* Through this process, a link key L_i is shared between the Trust Center and another device in the network for securing the communications between them. The procedure starts with the exchange of a trusted information, the *master key*, preinstalled during the manufacturing process. The master key is provided by the ZigBee Alliance to its members and is different for each application profile. After this phase, the device and the trust center exchange ephemeral data that are used to derive L_i. When two devices i and j need to communicate with each other, the Trust Center provides them with a link key $L_{i,j}$, encrypted using the link keys L_i and L_j, respectively. Note that this method cannot be used to generate network keys.

The process through which a new ZigBee network is set up or a new ZigBee device is added to an existing network is called *commissioning*. In addition to the commissioning procedures specific of the different application profiles, the ZigBee standard also specifies a common procedure that makes it possible to interconnect devices with different profiles.

Attack Surface: A possible attack vector in ZigBee network consists in discovering the keys used to secure the communications. For example, the repeated encryption of known and fixed messages (e.g., control messages defined in the standard) makes the system vulnerable to plaintext attacks [62]. This technique enables the recovery of a cryptographic key by having access to both the encrypted and decrypted messages. Hence, to ensure a high security level, the network key needs to be changed periodically.

A sinkhole attack against a ZigBee network is presented in [65]. The attack is performed through a malicious entity that legally joins the network, but then pretends to have an

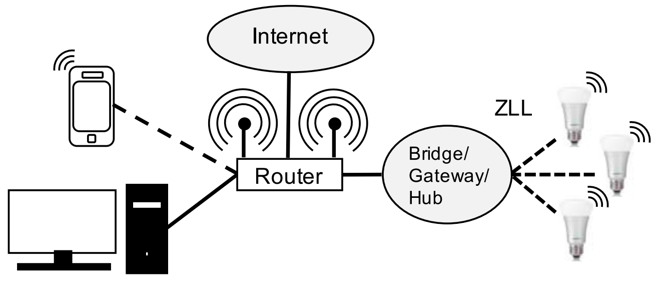

Fig. 5. General architecture of a smart light system.

efficient routing path toward the coordinator in order to attract more traffic flows. In this way, the attacker can modify or drop incoming packets. Moreover, if the malicious entity is directly connected to the Internet, all the ZigBee network is exposed to Internet attacks.

The sinkhole attack, however, requires that the malicious node is able to connect to the network and communicate with the other nodes. The *ghost* attack presented in [66], instead, does not require any knowledge of the communication keys. Its aim is to drain the ZigBee node energy, increasing the success probability of other DoS attacks. The strategy consists in injecting fake messages with increasing frame counters into the network, impersonating one legitimate node. At the receiver side, if the fake message counter is larger than that stored for the sending node, the counter is updated and the message is accepted and processed. Even if the message will be dropped during the integrity check, the node consumes some energy for the processing. The attack will also inflate the frame counter at the receiver, possibly creating a misalignment with the counter at the legitimate source, whose messages may be misinterpreted as duplicate and then disregarded.

Finally, KillerBee is a practical tool for hacking ZigBee devices [67]. The framework makes it possible to sniff and inject traffic in a ZigBee network as well as decode and manipulate packets. In [68], some attacks that exploit this tool, like replay attacks, are presented.

Other attack vectors are specific to ZLL installations. In 2012, LIFX and Philips presented their first smart lights solutions and, afterward, many other companies developed similar connected light systems. Many vendors, such as Philips, use the ZLL application profile. A general smart light system architecture is presented in Fig. 5.

Based on several reports, however, many smart light systems implement only the essential security mechanisms required to obtain the ZigBee Alliance's certification [64]. At a first analysis, it may seem unnecessary to implement many security precautions in a light system, since it does not involve the transmission of confidential information, and can still be operated manually in case the network does not work properly. However, as explained above, attackers may use these devices to relay an attack to the rest of the home or corporate network, bringing more critical devices at risk.

Morgner *et al.* [64] investigated the security level in three different ZigBee smart light systems, namely *Osram Lightify*, *GE Link*, and *Philips Hue*. The study evaluates vulnerabilities of both bulbs and interconnected devices, and reports seven different types of attack. The attacks are based on the *inter-PAN frames*, which are used to transmit touchlink commissioning commands such as scan request and scan response. These frames are neither secured nor authenticated: a malicious entity can send the same commands pretending to belong to the network. The attacker can then do illicit operations, compromising the security of the network, as better explained next.

1) *Active Device Scan:* The scan searches for ZLL devices in the range of the attacker, sending scan requests on different channels. By listening to the corresponding scan responses, the attacker can obtain a complete overview of the devices connected to the network. The three analyzed systems exhibit different behaviors: a) all light bulbs and controller from lightly respond to the attacker's scan request; b) the GE link controller does not respond; and c) the hue controller responds only if its Touchlink commissioning button has been pushed within the last 30 s.
2) *Blink Attack:* This attack can be activated after a device scan by sending to the victim device the inter-PAN command *identify request*. In this way, the device starts to blink for a default period. The *identify request* command is implemented in all the three lightbulb systems to allow the user to visually identify which device has a certain network address. As a consequence, all the systems are vulnerable to the blink attack.
3) *Reset Attack:*. The attacker performs a device scan and then resets them all to the factory state by sending the inter-PAN command *reset to factory new request*. All devices of the three lightbulb systems are vulnerable to this attack.
4) *DoS Attack and Hijack Attack:* In these attacks, the end user loses control of the victim device. Two strategies can be adopted for DoS attacks. The first consists of forcing the device to change the transmission channel, sending a *network update request* inter-PAN command including the new channel. As a second option, the attacker can cause the device to join a nonexisting network, changing its network key with arbitrary bytes. This is possible by sending the inter-PAN command *network join end-device request*: at the reception of the command, the device leaves its current network, changing its parameters according to the new configuration. The hijack attack works similarly to this second approach, with the difference that it forces the device to join an existing network chosen by the attacker. In this case, the network key of the desired network is used. All the evaluated smart light systems are vulnerable to DoS and hijack attacks. However, all of them integrate user functions to reobtain control over the attacked devices.
5) *Network Key Extraction Attack:* This attack makes it possible to find the current network key by eavesdropping the messages exchanged by the devices during the touchlink commissioning procedure. A preliminary DoS attack is needed to disconnect the device from the network. After that, the victim device will start a commissioning procedure in order to regain access to the network. Therefore, the attacker can extract the network key from the *network join end-device request*. In fact, as mentioned, the network key is encrypted using the well known master key. Only Philips hue devices are vulnerable to this attack since the touchlink commissioning procedure is not enabled in the other devices.
6) *Inject Commands Attack:* This attack makes it possible to send commands to the devices in order to control their actions. The knowledge of the current network key is needed (e.g., via the execution of the previous attacks). All the analyzed smart light systems are vulnerable to this attack.

B. Bluetooth Low Energy

Description: Bluetooth is a widely used short range wireless communication protocol. Its low energy and IoT-tailored version, named BLE, has been first introduced in the Bluetooth core specification version 4.0 [69].

A BLE network is composed of two types of devices: masters and slaves. The masters act as initiators during the communication setup and the slaves associate to them [70]. The entities are connected in a star topology, where each slave is associated with a single master, as exemplified in Fig. 6.

BLE operates in the unlicensed 2.4 GHz ISM band and uses 40 channels with a 2 MHz spacing [70]. The physical layer data rate is 1 Mb/s and the coverage range is typically over various tens of meters. The BLE MAC layer is split into two parts: advertising and data communication. 37 of the available channels are used during the transmission of data and the remaining 3 are used by unconnected entities to broadcast device information and establish connections [71].

In the data communication phase, data is normally sent in bursts to save energy. In this way, slaves can remain in sleep mode for long periods, waking up periodically to listen to the channel for possible messages from the master. The master decides the rendezvous instants with the slaves, according to a time division multiple access (TDMA) scheme. Communication reliability is provided through a stop and wait (S&W) automatic packet retransmission mechanism, based on cumulative acknowledgments.

As depicted in Fig. 7, besides physical and MAC layers, the stack entails other protocols such as the logical link control and adaptation protocol (L2CAP), and the low energy attribute protocol (ATT).

BLE encryption and authentication processes are based on AES-CCM with 128 bit keys, as for ZigBee. The symmetric key for a master–slave link is generated during the pairing procedure, which is executed as follows.

1) The devices exchange their authentication capabilities and requirements. This phase is completely unencrypted.
2) The devices generate or exchange a temporary key (TK) using one of the available pairing methods. Then they exchange some values to confirm that the TK is the same for both devices. After that, a short term key (STK) is generated from the TK. The STK will be used to encrypt the data stream.

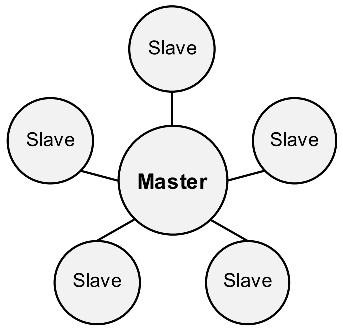

Fig. 6. BLE star topology: each slave is associated with a single master.

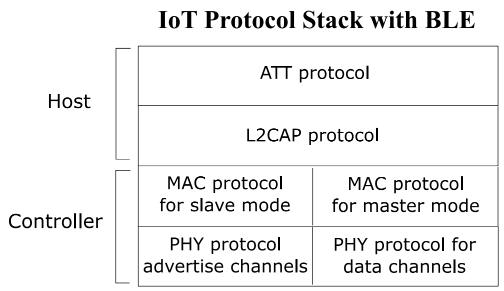
IoT Protocol Stack with BLE
Host
ATT protocol
L2CAP protocol
Controller
MAC protocol for slave mode
MAC protocol for master mode
PHY protocol advertise channels
PHY protocol for data channels

Fig. 7. BLE protocol stack. The technical specifications for BLE can be found in [69].

Finally, a *bonding* phase may optionally follow the pairing procedure: in this, the devices exchange and store common link keys (i.e., *bonds*) that can be reused when a link between the two devices is re-established at a later time.

The available pairing methods are the following three.

1) *Just Works:* The TK is set to 0. Of course, this does not provide any level of security.
2) *Out of Band:* The TK is exchanged out-of-band, e.g., using near field communication. This method provides a security level that is as high as that of the out-of-band method used to exchange the key. However, it can be inconvenient for the user.
3) *Passkey:* The TK is a six digit number that the user passes between the devices. For example, one of the devices generates the number and show it on a display: the user must then input the same number to the other device. In this case, the security level is high, but the devices need to be equipped with user's interfaces that make it possible to read and type-in the TK, which may be impractical for miniaturized IoT devices.

Starting from BLE version 4.2, a new pairing procedure has been put in place, using elliptic curve cryptography.

1) Each device generates an elliptic curve Diffie Hellman (ECDH) public-private key pair. Then, they exchange the public key with each other and derive a key, called *DHKey*, from their own secret key and the public key of the other device, using elliptic curve functions.
2) The devices use one of the available pairing methods (see below) to confirm that *DHKey* is the same for both of them and to generate a long term key (LTK) that will be used to symmetrically encrypt the data stream.
3) Optionally, the devices can perform a final step, like that for BLE version 4.1.

The pairing methods have also been updated, with the introduction of a new option and the hardening of the methods in the previous version.

1) *Just Works:* The noninitiating device generates a nonce and a confirmation value C_b, function of the nonce and the public keys of the two devices. The C_b and the nonce are then sent to the initiating device. The latter generates its own nonce and sends it to the noninitiating device. It also uses the noninitiating devices nonce and the public keys to check whether the received code C_b is valid. Clearly, this does not provide any security, since an attacker may generate its own nonce and use the public keys to create a valid confirmation value.
2) *Numeric Comparison:* This method is as just works, but the devices also generate a value which is function of the public keys and the nonces. This value must be displayed to the user, which must manually confirm that the shown number is the same in both devices. This last step solves the issue with the previous method.
3) *Out of Band:* With this method, random numbers and commitment values, which are functions of the random numbers and public keys, are exchanged in an out-of-band fashion, e.g., using near field communication. Using this method, the security level achieved is equal to the integrity and secrecy of the out-of-band method of choice.
4) *Passkey:* In this method, the user first inputs a k bit long secret passkey to both devices (or reads it from one of the devices and inputs it to the other). Then, for each bit $i = 1, \ldots, k$ of the passkey, the devices must perform a two-step procedure: a) each device generates a nonce and computes a commitment value, which is function of the nonce, the passkey, and the public keys. Commitments and nonces are then exchanged between devices. b) after that, each device recalculates the commitments as before, but exchanging the order of the two public keys, and using the nonce of the other device. If the passkey is the same, the commitment value must be equal to that found before.

The use of elliptic curve cryptography, however, is not without drawbacks: based on the experimental results presented in [72], the energy consumed to perform a single ECDH-ECDSA key exchange is more than 6000 times larger than that required by symmetric encryption techniques (236 mJ versus 38 μJ).

Attack Surface: The pairing methods just described have some important security issues. In BLE 4.0 and 4.1, there is no protection to eavesdropping and man-in-the-middle attacks during the pairing phase, except for the out of band pairing. In fact, in the Just Works pairing method the key is known, while in the Passkey method the key is easily brute-forced. In some cases, brute-force is not even required [73]–[77]. BLE 4.2 is affected by similar problems, principally for the Passkey pairing, since the passkey is verified one bit at a time [77], [78]. When the attacker is interested in eavesdropping, it can try

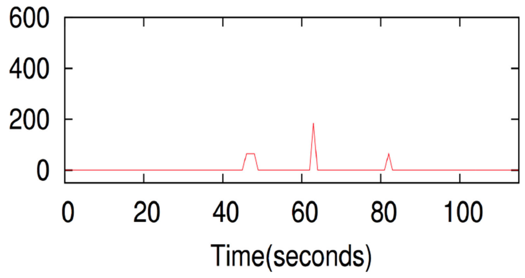

Fig. 8. Data rate [bps] versus time—*Resting* user [71].

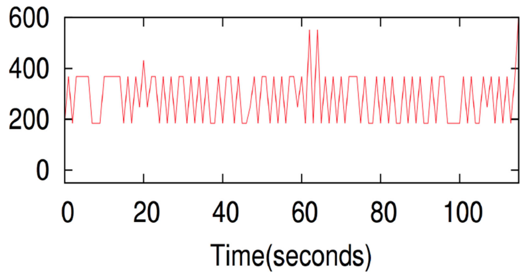

Fig. 9. Data rate [bps] versus time—*Running* user [71].

to match the confirmation value considering the current bit r_i of the key equal to 0. If the confirmation value does not match, then $r_i = 1$. When trying to directly connect to a device, instead, the attacker can consider $r_i = 0$. If the other device aborts the procedure, then $r_i = 1$. This procedure can be repeated for bit $i = 1, \ldots, k$, learning, therefore, the entire key.

Another issue is linked to the advertise mode of BLE devices. Das *et al.* [71] found that the analyzed fitness trackers are almost always in advertise mode. This is because the master device frequently disconnects from the tracker in order to preserve energy. Therefore, when the smartphone application for the fitness tracker is not running, the tracker closes its communication link, remaining in advertise mode until the next connection establishment. Also, most of the devices analyzed in [71] always expose the same MAC address. This makes it possible to capture exchanged messages and correlate over a long period of time the BLE traffic between a pair of devices. As an example, an attacker may be able to track the movements of the BLE device owner or even just verify its presence in an area. Furthermore, this attack may be used to track the user's activity: in fact, even though the packets are encrypted, the volume of data exchanged is a clear indicator of user's motions, as exemplified in Figs. 8 and 9. The figures show the difference between the BLE data traffic exchanged by a tracker and a smartphone for a resting and a running user. As a security feature, the BLE specifications allow a device to use random MAC addresses and to frequently change them. For example, the *Apple Watch* randomizes the MAC address both when it is rebooted and during normal usage at an approximately 10 min interval [79].

Arias *et al.* [80] described another type of attack that could be conducted against wearable fitness devices. This is based on firmware hacking, which is possible only if the attacker has physical access to the device. The device considered, a *Nike+ Fuelband*, contains a standard USB connector used for charging and synchronizing with the computer. This connector is also used by the manufacturer to write the firmware into the device memory. Using the same USB port, it is possible to send commands to the STM32 microcontroller in the device, enabling unrestricted read and write operations into the memory. Therefore, it is possible to flash a tampered firmware version and disable all protection mechanisms. The possible consequences of a firmware attack like this could range from the possibility of a back-door injection, in order to leak user information or credentials, to the installation of rogue services allowing for full remote control of the device by the attacker.

Furthermore, the analysis in [79] shows that, in most cases, fitness tracking applications transmit every logged event over the Internet, and in some cases, it is even unclear why such transmissions are occurring. This allows an attacker to perform data analysis on the amount of exchanged data. On top of that, the authors found that Garmin devices do not use an encrypted protocol to transmit data between the mobile device and the Garmin servers, except during the account creation phase. By exploiting this vulnerability, a man-in-the-middle attack is able to capture e-mail addresses and session identifiers, which are transmitted in clear-text. A similar result is presented in [81], using FitBit devices: in this case, login credentials are forwarded in clear-text in an HTTP POST request.

At DEF CON 2016, researchers presented several security vulnerabilities of a large number of door lock and padlock devices that use BLE to communicate with the user's smartphone [82], [83]. They found that the passwords of devices like *Quicklock Door lock*, *Quicklock Padlock*, and *iBluLock Padlock* are transmitted in clear text and hence can be easily eavesdropped. Other devices use passwords composed by a small number of characters, making brute force attacks feasible. The *Ceomate Bluetooth Smart Door lock* and other devices are vulnerable to replay attacks due to the bad implementation of the encryption mechanisms. The *Okidokey Smart Door lock* can be opened by just changing the value of one byte that brings the door lock in an error state, forcing it to open. The *Mesh Motion Bitlock Padlock* is vulnerable to man-in-the-middle attack: the attacker is able to impersonate the lock to steal the password sent by the user's smartphone. The vulnerabilities of the *August door lock* are investigated in [84] and [85]. However, August seems to be very proactive to fix the discovered issues: when in 2015 the hard-coded encryption key was revealed, the code was patched in 24 h [86].

Finally, as for ZigBee, attack softwares can also be found for BLE. For example, *GATTack* enables to break different aspects of the BLE security [87].

C. 6LoWPAN and CoAP

Description: 6LoWPAN and CoAP are two IETF protocols that can be implemented in IoT devices to ease their interaction with standard IP-based systems. As illustrated in Fig. 10, 6LowPAN is an IPv6 adaptation protocol for resource-constrained devices that communicate over low power and

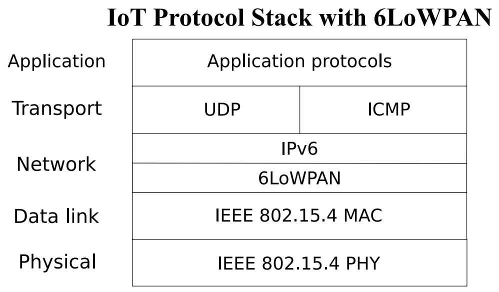

Fig. 10. 6LoWPAN protocol stacks, as specified in [88].

lossy links, such as IEEE 802.15.4 [88], [89]. 6LowPAN makes use of compression and fragmentation mechanisms to reduce the size of the IP datagrams and remove most of redundant fields. CoAP, instead, is a RESTful protocol at the Application Layer, laying on top of the UDP transport protocol. It has been designed to be easily mapped into HTTP via proxies, to support retransmissions, sleepy devices, and resource discovery. The use of UDP makes it possible to avoid the support of the rather sophisticated connection-control mechanisms of TCP, but on the other hand requires to account for out-of-order message delivery and loss packets.

At the PHY and MAC layers, networks employing 6LowPAN and CoAP typically relay on protocols from the IEEE 802.15.4 family. Instead, the routing within the IoT network is usually based on the IPv6 routing protocol for low-power and lossy networks (RPL), defined in RFC 6550 [90]. RPL has been mainly designed for multipoint to point communications, such as those in wireless sensor networks. However, it also supports point to multipoint (sink broadcast) and point-to-point (leaf nodes communicating with each other). RPL builds a directed acyclic graph (DAG) based on a root node called Low power and lossy border router (LBR), usually being the device responsible for the management of a group of nodes and representing the border between two networks. From the DAG, RPL creates a destination oriented direct acyclic graph (DODAG) tree, exemplified in Fig. 11. The DODAG contains only one root and is loop free. Starting from the DODAG root, devices broadcast their DODAG information objective (DIO) message, which contains device and link metrics. The global repair and local repair mechanisms are used in case of a broken link: the first recalculates the whole topology, while the second operates locally, by informing all the children of a node that they need to update their parent.

Attack Surface: A hypothetical attacker can target the RPL or operate at the adaptation layer, based on the level of control over the network that it wants to achieve.

a) Attacks against RPL: Many of the attacks on 6LoWPAN focus on redirecting traffic and disrupting the routing tree. In the following we report some examples of such attacks [91]–[93].

1) *Clone ID and Sibyl Attacks:* In the clone ID attack, the malicious node clones the identity of another node. In the sibyl attack, the attacker uses the identity of several entities at the same time. In this way, the malicious entity can access and redirect a large amount of network traffic. These types of attack can be detected by keeping track of the number of instances of each identity or by monitoring the geographical location of the devices.
2) *Sinkhole Attack:* This is the same type of attack described for ZigBee: the malicious node declares to the sink very efficient routing paths toward the other nodes, gaining control over a large part of the traffic flows.
3) *Selective Forwarding and Black Hole Attacks:* These attacks take place when a node of the network, that is supposed to forward the packets along the correct routing path, discards some of the traffic (selective forwarding), or all the traffic (black hole), that passes through it. Possible solutions may be the creation of disjoint or dynamic paths inside the DAG.
4) *Hello Flooding Attack:* The *Hello* message is used by a node in a 6LoWPAN network to announce its presence. If a node receives a Hello message, it assumes that the sender node is in its neighborhood, and can thus be directly reachable. An attacker can exploit this mechanism by broadcasting *Hello* messages using a transmission power larger than that permitted. In this way, a substantial number of nodes consider the attacker as a neighbor. However, when one of them tries to use the new link, the sent packets will be lost, since the legacy transmit power level is not sufficient to guarantee good communication. This type of attack can be avoided using link layer acknowledgments to check the message reception.

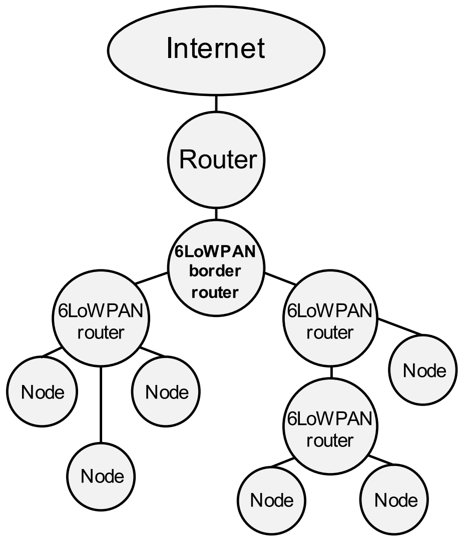

Fig. 11. DODAG tree built for 6LoWPAN routing.

Contrary to the previous ones, the following attacks relies on the exploitation of the RPL service messages [91]–[93].

1) *Local Repair Attack:* The malicious node continuously sends local repair messages. This forces repeated updates of the network topology, even if there is no connectivity problem. These operations are costly both in terms of computational resources and energy, causing service degradation and early energy depletion for battery operated devices.

2) *Version Number Attack:* The *version number* is a field of DIO messages that is incremented at each rebuilding of the DODAG. The malicious transmission of DIO messages with a higher version number may force the whole DODAG to be unnecessarily rebuilt. Again, this causes service degradation and energy depletion.

b) Attacks from the Internet side: Neither 6LoWPAN nor CoAP provide secrecy, authentication, or integrity protection. Therefore, the use of 6LoWPAN and CoAP without additional security measures makes the devices fully accessible from the Internet. A proposal has been made to extend CoAP in order to provide built-in security, but it has not yet been included in the standard [94]. The CoAP specifications, instead, suggest the use of datagram TLS (DTLS) to provide secrecy, authentication, and integrity protection [35], [95]. Alternatively, IPsec [96] can be used to provide authentication and encryption at the IP level.

If no encryption and authentication mechanisms are used, an attacker could easily gain access to the IoT network and eavesdrop sensitive information from the data flow. Additionally, the attacker may gain the control of the sensor nodes, e.g., to create a bot-net [97]. As an example, Hernandez *et al.* [98] presented a modification of the *Nest thermostat* firmware to use it as part of a bot-net. The bot-nets may then be used to launch DoS attacks to other targets. This kind of attack can be detected through an analysis of the node traffic: in fact, nodes in a bot-net usually transmit more data than clean devices, in order to maintain the bot-net.

c) Attacks at the adaptation layer: The forwarding of packets between the public Internet and the 6LoWPAN network is implemented at the border router. The lack of authentication and the limited computational resources of the devices that perform the adaptation make this mechanism vulnerable to attacks. Two attacks that can be performed at this level, *fragment duplication* and *buffer reservation*, are presented in [99]. The fragment duplication attack relies on the fact that a node cannot verify at the 6LoWPAN layer if a received fragment belongs to the same IPv6 packet of the previous ones, since this control is performed at higher layers. If a malicious node injects fragments with the same header of the legitimate 6LoWPAN packet, the target node cannot distinguish between them and the legitimate ones. Therefore, it cannot decide which fragments have to be used during packet reassembly procedure. This causes the reconstruction of a corrupted IPv6 packet, which is consequently dropped.

The buffer reservation attack leverages the limited memory of the network nodes. In the 6LoWPAN network, receiving nodes must reserve buffer space to reassemble the fragments that belong to the same IPv6 packet. When the reassembly buffer is assigned to one IPv6 packet, received fragments of other IPv6 packets are dropped. Since buffer space reservation is kept for 60 s, if an arbitrary fragment is transmitted by the attacker to the target node, the latter will not be able to receive further fragmented packets in the following minute. Consecutive repetitions of this attack cause a long term DoS to the targeted device, while employing just a small amount of the malicious node resources.

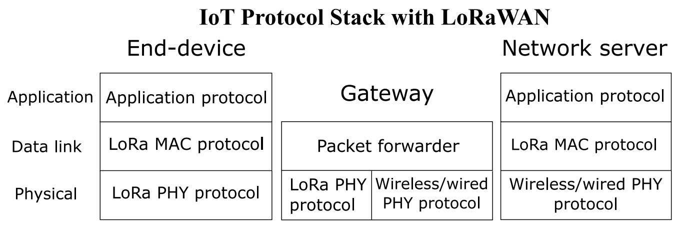

Fig. 12. Protocol stacks of LoRaWAN end-devices, gateway, and network server. The document in [100] provides the specifications for LoRaWAN.

In order to protect 6LoWPAN networks from the attackers, intrusion detection systems specifically tailored to IoT networks have been studied [91]–[93]. An intrusion detection system monitors the network parameters and can identify signs of intrusions or attacks. Intrusion detection systems for 6LoWPAN networks are optimized to save the largest amount of network resources. Due to the vast attack surface, intrusion detection systems should operate both at the adaptation, RPL, and Application Layers. Therefore, a hybrid architecture is needed, in which a centralized module, installed on the border router, cooperate with distributed modules installed on internal nodes.

D. LoRaWAN

Description: LoRaWAN, introduced in 2015 by the LoRa Alliance [100], is a link layer protocol that sits on top of the LoRa physical layer. The protocol stack is shown in Fig. 12. As for the previous protocols, LoRaWAN is optimized for battery-powered end-devices. LoRaWAN has a star-of-stars topology that includes end-devices, gateways, and network server, as exemplified in Fig. 13.

In an LoRaWAN network, the end-devices communicate via single-hop links to one or more gateways, which are themselves connected to a single network server via legacy IP technologies. LoRa communication uses channels in the 868/900 MHz ISM band. The data rate ranges from 0.3 Kb/s to 50 Kb/s with a communication range of many kilometers. The communication is bidirectional and is always initiated by the end-device. After each uplink transmission, the end-device opens two *downlink windows* in different sub-bands to receive data from the network server. The protocol used to access the channel is ALOHA [101], [102].

LoRa end-devices can belong to three different classes, namely A, B, and C, which are associated to different operation modes. The operation mode of class A, described above, has to be implemented by all LoRaWAN devices. Class B and C, instead, offers some additional features. Class B devices can open extra receive windows at scheduled times to enable the reception of unsolicited messages from the network server. Class C devices, instead, are expected to be connected to the power grid and, then, can keep the receive window always open.

The commissioning procedure by which a device can join an LoRaWAN network is named over the air activation (OTAA). This procedure leverages on some information stored on the device: the end-device identifier (DevEUI), the application identifier (AppEUI), and the application key (AppKey). The

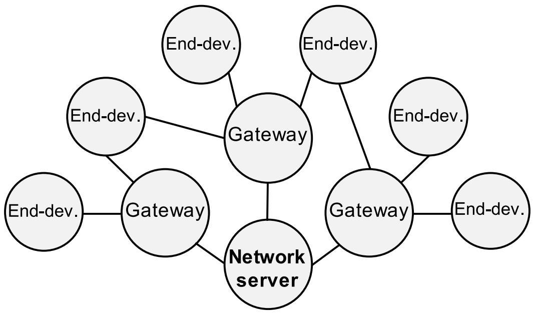

Fig. 13. LoRaWAN star-of-star topology.

first two are unique global identifiers for the end-device and the application provider, respectively. Instead, the AppKey is an AES-128 key assigned by the application owner and is specific to the end-device. The uniqueness of the key is needed to ensure security: if the key is shared between devices, traffic eavesdropping becomes possible. The join procedure is initiated by the end-device by sending a *join request*, including the DevEUI and AppEUI. If the device is allowed to join the network, the network server replies with a *join accept* message. The message includes the network identifier (NetID), the application nonce (AppNonce), and the end-device address (DevAddr). The NetID and DevAddr are 32 bits long: the first uniquely identifies the network, while the second identifies the end-device within the network. The AppNonce is used by the end-device to derive the network session key (NwkSKey) and the application session key (AppSKey), which are specific for each end-device. The latter and the network server use the session keys to encrypt and decrypt the payload of the messages. Specifically, NwkSKey is used for MAC commands, while AppSKey is employed for application specific messages. These keys are also used to verify the MIC, to guarantee data integrity. In fact, the MIC is a 4 bytes tag, obtained by encrypting the message with NwkSKey using AES-128.

Once the end-device has joined the LoRaWAN network, all future messages are encrypted using a combination of NwkSKey and AppSKey. The payload encryption is performed using AES-128, and is based on the following procedure. First, a number of 16-byte information blocks $\{A_i\}$, are created. Each block contains the DevAddr, two frame counters, and a byte indicating the stream direction. Each block A_i is then encrypted to obtain the corresponding S_i block using the NwkSkey, if the payload to be encrypted consists only in MAC management commands, and the AppSKey otherwise. The payload is finally encrypted by the XOR operation between the message itself and the sequence of blocks S_i [100].

Attack Surface: The first weakness of the LoRaWAN protocol is related to *key management* [103]. AppKey, NwkSKey, and AppSKey are all stored in the end-devices. Therefore, using a *side channel analysis attack* it can be possible to recover the keys exploiting the variations in power consumption or electromagnetic emissions from the transceiver during the encryption. This is eased by the fact that LoRaWAN devices are expected to work in an unattended fashion and in remote locations, and hence surreptitious physical access to the device is possible. NwkSKeys and AppSKeys are stored also in the network server that generated them. Violation of the network server by other means, e.g., by attacking a nonsecure service running on the same computer, may give access to the LoRaWAN network and applications.

Furthermore, because of the way the protocol is designed, nodes must share the same NwkSKey and AppSKey if they need to support multicast messages. In this case, discovering the keys from just one node will give access to all the other communications. To overcome this problem, the nodes need to be able to differentiate between multicast and unicast communications. In the first case, they should use the common key, while the unique NwkSKeys and AppSKeys should be preferred for unicast communications. In this way, if the attacker corrupts the multicast key, only multicast messages would be insecure.

An attacker may also focus on the link between the gateway and the network server. Even if, in principle, any legacy IP protocol can be used for such a connection, in practice many deployments relay on the protocol provided by Semtech, which is based on plain unprotected UDP. Therefore, while the LoRaWAN packets payload is encrypted, still it is possible to disrupt the network services by forging or modifying network management packets.

Class B networks introduce additional threats because of beacons and multicast messages [103]. Beacon messages are not encrypted: they represent a source of information about the network and provide a way to inject malicious data into it. In fact, beacons can also be generated by an attacker, and, since a node cannot distinguish between malicious and genuine beacon messages, the network operations could be easily disrupted.

V. Examples of Implementations in Commercial Devices

In general, IoT devices include at least two microcontrollers: one responsible for the management and processing of the data and the other for connectivity. In the following of this section we investigate the characteristics of the processors in charge of the connectivity aspects, with a particular focus on their security features.

Most of the considered devices use ARM microcontrollers of the Cortex-M series. Especially, the M0, M0+, and M23 microcontrollers are designed for applications that require minimal costs, power, and size. Because of these characteristics, they are the most adopted in embedded applications. Instead, the M3 and M4 models offer a balance between performance and energy efficiency. Lastly, the M7 is the most powerful controller, designed for high performance embedded applications [104]. Fig. 14 reports the relative performance of different Cortex-M designs with respect to the Cortex M0. Differently from the more powerful Cortex-A and Cortex-R processors, the Cortex-M series is provided only with a *Memory Protection Unit*, which is a trimmed down version of the *Memory Management Unit*: it only provides memory protection, instead of providing full virtual memory management. Also, the Cortex-M microcontrollers are equipped with

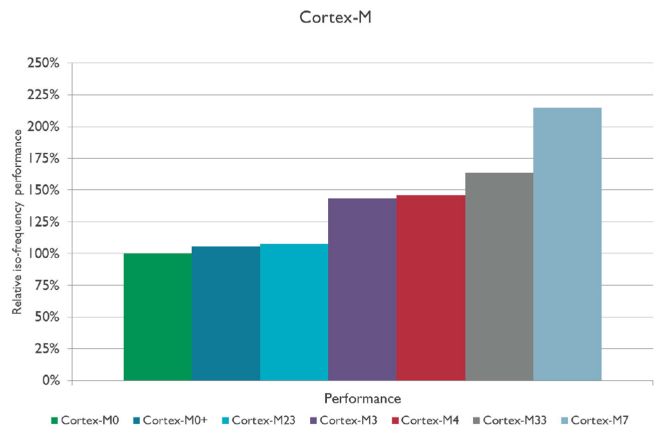

Fig. 14. Arm Cortex-M series [105].

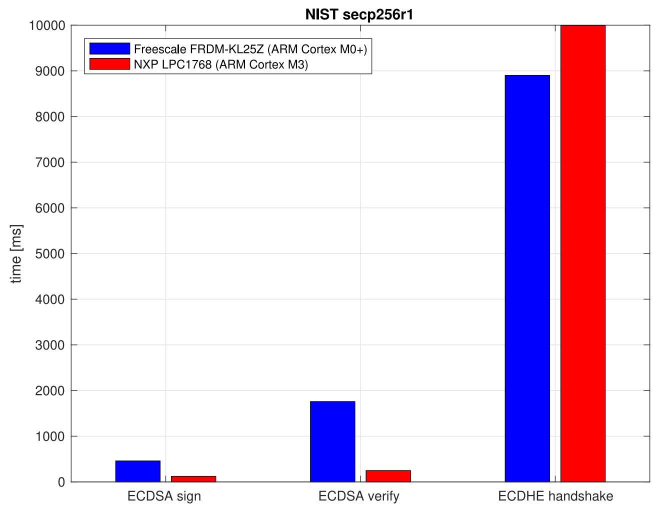

Fig. 15. Performance comparison of elliptic curve cryptography using the NIST *secp256r1* curve in different microcontrollers [106].

less RAM and flash memory with respect to microcontrollers from the other two classes [42].

Microcontrollers from the Cortex-M family do not integrate any hardware pseudorandom number generator, nor any module supporting cryptographic algorithms such as AES. Therefore, support for cryptographic algorithms is implemented via software or by dedicated co-processors, which is the most common solution in commercial devices.

For what concerns software cryptographic routines, performance could wildly vary between different microcontrollers, even within the same family. In this regard, Tschofenig and Pegourie-Gonnard from ARM have collected performance figures of different cryptographic algorithms implemented in software in the M0/M0+ and M3/M4 microprocessors [42], [106]. The algorithms taken into account are ECC, SHA, and AES. Tests were performed on two different development boards: the NXP LPC1768 and the Freescale FRDM-KL25Z. The former embeds an ARM Cortex-M3 CPU at 96 MHz, 512 KB flash memory and 32 KB RAM. The latter is controlled by an ARM Cortex-M0+ CPU at 48 MHz, 128 KB flash memory and 16 KB RAM.[1] The authors exploit different NIST *secp*r1* elliptic curves used in cryptography and show performance results using different optimization settings. Fig. 15 reports the results for the two development boards: in particular, the performance of different phases of elliptic curve cryptography algorithms are compared when using the NIST *secp256r1* curve. Both CPU speed and RAM usage have a significant impact on performance. Therefore, it is crucial to reach the optimal tradeoff between RAM usage and performance. Tschofenig and Pegourie-Gonnard state that, usually, the increased performance is worth the additional RAM usage.

A. *ZigBee Implementations*

As introduced in Section IV-A, different smart light systems adopt the ZigBee protocol. The *Philips HUE* system is one of them and presents multiple hardware implementations (Table I). Most of the light bulbs available on the market embed the wireless module ATMEL SAMR21, which has two components: 1) the ATMEL SAMD21 core (based on the ARM Cortex M0+ architecture) and 2) the AT86RF233 radio transceiver, which supports both ZigBee and 6LoWPAN protocols. The ARM Cortex M0+ microcontroller does not implement any hardware cryptographic routine, since it lacks a hardware pseudorandom number generator. To overcome this problem, the AT86RF233 incorporates a two-bit random generator where the source of randomness is given by the noise observations [107], [108]. The generated random number can be used both for the creation of a random seed for the AES key generation. In addition, a dedicated hardware module is included to perform the AES-128 encryption procedure. To use this module, a 128-bit nonmodifiable initial key needs to be preinstalled in the device. The source of entropy is updated every 1 μs. Other Philips HUE products (e.g., the *Philips Go*) embed a different 8-bit microcontroller, the ATMEL M2564RFR2, which still uses the just described mechanism as the entropy source. The *HUE Bridge*, another product in the Philips HUE line, embeds the Texas Instruments CC2530 microcontroller, which integrates an Intel 8051 microprocessor and a co-processor for AES encryption and decryption. Pseudorandom numbers are generated using a 16-bit linear feedback shift register, which can be seeded with random data from the noise in the radio analog-to-digital-converter [109].

The *Nest Protect* smoke alarm and the *Nest thermostat* also use ZigBee for communications. They adopt the EM357 system-on-a-chip (SoC) from Silicon Labs, which is based on the 32-bit ARM Cortex-M3 and includes an IEEE 802.15.4 radio transceiver [110]. The SoC is provided with additional hardware to perform AES encryption: the AES CCM, CBC-MAC, and CTR modes are implemented in hardware. The 16-bit seed is generated from the analog circuitry thermal noise.

B. *BLE Implementations*

As mentioned in Section IV-B1, most of the considered fitness bands embed microprocessors from the ARM Cortex M

[1]The considered boards have no hardware pseudorandom number generator, which means that the low entropy of the generated numbers would not be suitable for real world deployments. This, however, does not affect the speed of the considered cryptographic functions, therefore the results are still significant.

TABLE I
SOME ZLL PRODUCTS

Product	Transceiver	(Radio) MCU
Philips HUE	AT86RF233	SAMD21
Philips Go	M2564RFR2	
HUE Bridge	CC2530	
Nest Protect	EM357	
Nest thermostat	EM357	

TABLE II
SOME BLE PRODUCTS

Product	Transceiver	(Radio) MCU
FitBit Flex 2	nRF8001	CC2540
Fitbit Charge HR	nRF8001	CC2540
Fitbit Charge 2	BlueNRG	
Nike+ Fuelband	CC2564B	MSP430F5
Nike+ Fuelband SE	CSR1010A	

family. For what concerns the BLE connectivity module, as shown in Table II, both the popular *FitBit Flex 2* and *FitBit Charge HR* use the nRF8001 by Nordic semiconductor for the BLE connectivity [111]. The latter works together with the Texas Instruments CC2540 microcontroller, which provides an AES security co-processor for encryption and decryption purposes [112]. A different design choice has been taken for the newer *FitBit Charge 2*, which uses the BlueNRG SoC by STMicroelectronics. This SoC includes an ARM Cortex M0 microcontroller, which embeds an AES security co-processor, and a separate Bluetooth module [113].

The *Nike+ Fuelband* uses two different modules: the CC2564B BLE radio interface and the 16-bit MSP430F5 MCU, both from Texas Instruments [114]. The first of these modules has a 128-bit hardware encryption accelerator that, however, only supports the version 4.0 security features. The more recent *Nike+ Fuelband SE* uses the Qualcomm Bluetooth Smart IC CSR1010A, which has a dedicated module for AES encryption, but is limited to the BLE 4.1 specification too [115].

C. Devices With 6LoWPAN Stack

The *Lifx Color 1000* light bulb system implements the 6LoWPAN protocol, and uses the Texas Instruments SoC CC2538 [116]. This SoC is based on the ARM Cortex M3 and implements AES in hardware. For 6LoWPAN applications, the ATMEL SAMR21, already discussed in Section V-A, can also be used. Another solution with AES hardware implementation is the LTC5800-IPM by Linear Technology, which is based on the ARM Cortex M3 [117].

D. LoRa Devices

LoRa development boards use the STM32L052T8Y6 MCU coupled with the Semtech SX1272 radio transceiver [118]. The MCU is based on a Cortex M0+ microprocessor. Differently from the previous protocols, not many consumer-level IoT products that implement LoRaWAN are available. In fact, LoRaWAN is more commonly used in sensor networks to collect data from large areas: the products using this protocol are heavily customized for each application.

VI. FINAL REMARKS

Given the widespread adoption of IoT solutions for sensor networks, domotics, and health application, providing secure communications in IoT networks is of paramount importance. In this scenario, the end user is often unaware of the security issues related to the devices which are part of her/his everyday life. For example, without protections against eavesdropping, IoT wireless networks can expose sensitive personal information. Also, without authentication, malicious attackers can masquerade as legitimate devices and disrupt IoT network operations. In addition to the issues related to the protocol and network security design, the reduced computational capabilities and the need for low energy consumption limit the cryptographic functionalities that can be installed in IoT devices.

As we have discussed in this article, in ZigBee and BLE implementations, ease-of-use is favored over strong security. This is because of the supposedly noncritical applications they usually support, underestimating the risk that such technologies can be exploited to enter more critical systems, for example when a dual-stack device is attached to both a critical and noncritical system.

6LoWPAN and LoRaWAN adopt two complementary security strategies. Since LoRaWAN can be used, for very simple applications, without any transport and application layers, the protocol has been designed with strong security in mind and mandatory packet encryption and authentication. The designers of 6LoWPAN and CoAP, instead, decided to delegate the security aspects to other layers and hardened only the aspects strictly connected to the protocol operation (e.g., routing and fragmentation). This allows, for example, the use of CoAP over LoRaWAN, without incurring in the cost for encryption and key exchanges at multiple levels.

Additional security extensions to existing standards have been proposed in the literature, but they have not been adopted by standardization entities and in commercial devices yet. For example, [72] proposes *CryptoCoP*, an encryption protocol for resource-constrained devices that supports energy-efficient symmetric key cryptography for BLE. Bonetto *et al.* [119] proposed another interesting solution to decentralize computationally intensive tasks of devices to a trusted and unconstrained node of the network; this node is then responsible for the calculation of the master session key on behalf of a group of constrained IoT devices. Zhou and Piramuthu [120] investigated some issues with security improvement proposals found in the literature for IoT devices, particularly for the *Fitbit* fitness trackers. In [121], an algorithm for IoT connection establishment and key exchange, between node and mediator, based on timestamps is proposed. The time-based secure key generation approach aims at efficiently managing and renewing the keys to provide a trustful connection, while guaranteeing the integrity of data transmitted over an insecure channel. Timestamps and nonces are also used to avoid attacks

TABLE III
SUMMARY OF THE SECURITY GOALS, VULNERABILITIES AND MECHANISMS FOR THE ANALYZED TECHNOLOGIES

Technology	Information level (Integrity, Anonymity, Confidentiality, Privacy)	Access level (Access control, Authentication, Authorization)	Functional level (Resilience, Self organization)
Zigbee	AES-CCM mode for message encryption and authentication using two different 128-bit keys for unicast (link key) and broadcast communications (network key); 128-bit MIC for integrity protection; 4-byte frame counter for reply protection. **Vulnerabilities**: if the network key is not changed periodically, an attacker that has discovered the key, is able to decrypt all the exchanged messages.	The Trust Center controls the access to the network: it is responsible for device authentication, join request management, network and link keys distribution. **Vulnerabilities**: the frames used to transmit commissioning commands are not authenticated, therefore an attacker can send these commands pretending to belong to the network.	In case of broken link, the device has to request again for access the network. **Vulnerabilities**: an attacker can force this situation to eavesdrop information during the subsequent commissioning process.
BLE	AES-CCM mode for message encryption and authentication using a 128-bit key. **Vulnerabilities**: if the device MAC address is not frequently changed, an attacker can violate user anonymity.	The master device acts as initiator in the communication setup and controls the access to the network. **Vulnerabilities**: in most of the pairing methods the key is known or can be brute-forced by an attacker during the process.	In case of broken link, the device has to request again for access the network.
6LoWPAN and CoAP	Neither 6LoWPAN nor CoAP provide secrecy, authentication, or integrity protection. **Vulnerabilities**: if no additional security mechanisms are used, the attacker can gain access to the devices. The lack of authentication allows the attacker to make the network service unavailable.	The LBR is responsible for the management of the network devices; the RPL offers the routing functionality though the creation of the DODAG tree. **Vulnerabilities**: an attacker can clone the identity of a node if there is no control on the number of instances of each device identity. If no disjoint paths are created inside the DAG and no link layer acknowledgments are sent, an attacker can make the service unavailable.	In case of broken link, DAG global repair and local repair mechanisms are used.
LoRaWAN	AES for message encryption with two 128-bit keys for command (network session key) and application messages (application session key); 32-bit MIC for integrity protection. **Vulnerabilities**: the keys are all stores in the devices and can be discovered by an attacker. If the same key is used for unicast and broadcast communications, an attacker that eavesdropped the key can decrypt all the messages.	The network server manages the join process. **Vulnerabilities**: as the UDP protocol is used, an attacker can modify the management packets and make the service unavailable. In class B networks, an attacker can extract information from the beacon messages that are sent unencrypted.	In case of broken link, the device has to request again for access the network.

TABLE IV
SUMMARY OF THE CLASSES OF ATTACKS THAT HAVE BEEN IDENTIFIED AGAINST DEVICES IMPLEMENTING A SPECIFIC TECHNOLOGY. EMPTY CELL: NO ATTACK REPORTED IN THE LITERATURE. FULL CELL: AN ATTACK HAS BEEN REPORTED IN THE LITERATURE. THE TARGET OF THE ATTACK IS ENCODED IN THE CELL COLOR: RED FOR ATTACKS TO THE INFORMATION LEVEL, BLUE FOR THE ACCESS LEVEL, GREEN FOR THE FUNCTIONAL LEVEL

Technology	Key-related attacks (eavesdropping, plain-text attacks, key extraction attacks, side-channel analysis)	DoS attacks to the data plane (sink hole attack, black hole attack, flooding)	DoS attacks to the device (reset, ghost attack, hijacking, forging of control packets, energy attacks)	Replay attacks	Attacks to the privacy of the communications (device scan, device tracking and user activity profiling)
Zigbee	🟥🟦	🟩	🟩	🟥	
ZLL	🟥🟦		🟩		🟥
BLE	🟥🟦		🟩	🟥	🟥
6LoWPAN	🟥🟦	🟩	🟩		
LoRaWAN	🟥🟦		🟩		

based on packet fragmentation in 6LoWPAN, as explained in [122].

VII. OPEN CHALLENGES

Despite the many mechanisms proposed by security experts, the field of IoT cybersecurity still offers a number of open research challenges. In Table III is reported the security mechanisms that the different technologies implement to fulfill the requirements specified by the security goals at the information, access and functional levels. In the same table, the vulnerabilities that have been identified in the literature are summarized. These vulnerabilities allow a malicious entity to attack the IoT devices and threat the security goals. In Table IV, the attacks described in this article are grouped into five relevant classes and, for each of them, the technologies vulnerable to attacks of that type are highlighted. A blank cell means that, to the best of our knowledge, there are not works in the literature that study the vulnerability of that technology to that specific class of attacks.

From the table we can see that both ZigBee and 6LoWPAN suffer from attacks against the network communication protocols, as the routing information in the packets are not usually authenticated. Similarly, they are subject to energy or resource depletion attacks that force the nodes to process a large number of forged packets. Full header and control packets authentication would block many of such attacks and improve the functional level robustness of the network. The deployment of such comprehensive authentication procedures, however, is hindered by two factors. The first is linked to the lack of a widely accepted lightweight encryption algorithm that could be implemented in hardware, or be supported by the most common cryptographic libraries for embedded devices. The second relates to the supplementary information to be added to the packets for this additional authentication requirement, which would increase the already significant overhead incurred for short payload messages. Therefore, there is a need for strong authentication mechanisms that use a small amount of additional space in the packet, or for mechanisms using alternative types of information, e.g., acquired from the physical layers. Physical layer authentication is a possible solution, but its maturity level is still insufficient for its implementation in commercial products.

A second open challenge is related to the management of shared cryptographic keys, which is a common weakness of the different protocols (see Table IV). As mentioned in the protocol analysis, commissioning and configuration procedure often require a shared key among all the nodes in the network. The management of such a key, however, is a complex task, since a key update requires its secure and timely distribution over the whole network. Moreover, additional overhead is required to guarantee the consistency of the shared key among the devices. Since a shared key is more likely to be discovered by attackers, key rotation should happen quite frequently, further exacerbating the problems. To avoid such complexity in shared key management, a mechanism is needed to allow devices to create and update shared keys independently, for example by using information available to all devices but unknown to attackers.

As we have seen, the physical implementation of a given technology is as important as its theoretical design. For example, the use of microcontrollers with poor quality entropy sources may allow an attacker to extract the key used in the cryptographic routines. Also, devices that do not randomize identifiers transmitted in clear, like MAC addresses, can be exploited to attack the users' privacy.

The *European Union Agency for Network and Information Security*, in [123], evaluates good practices to secure the life-cycle of IoT products and servers, looking at all the security aspects in this scenario. Security measurements are categorized into the three phases of IoT devices life-cycle: starting from the development of the products to their usage, passing through their integration in smart networks.

While the amount of issues reviewed in this article is consistent, we believe that a more careful design of the devices and the networks will make IoT systems really secure and will enable their use also for critical applications.

REFERENCES

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] D. Evans, "The Internet of Things. How the next evolution of the Internet is changing everything," San Jose, CA, USA, Cisco Internet Bus. Solutions Group, White Paper, Apr. 2011. Accessed: Jun. 2019. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[3] H. Almuhimedi *et al.*, "Your location has been shared 5,398 times! A field study on mobile app privacy nudging," in *Proc. 33rd Annu. ACM Conf. Human Factors Comput. Syst.*, 2015, pp. 787–796.

[4] S. Misbahuddin, J. A. Zubairi, A. Saggaf, J. Basuni, S. A-Wadany, and A. Al-Sofi, "IoT based dynamic road traffic management for smart cities," in *Proc. 12th Int. Conf. High Capacity Opt. Netw. Enabling Emerg. Technol.*, Dec. 2015, pp. 1–5.

[5] M. R. Warner, "Internet of Things cybersecurity improvement act of 2017," in *Proc. 115th U.S. Congr.*, Sep. 2017, p. 1691.

[6] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1294–1312, 3rd Quart., 2015.

[7] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an analysis of security issues, challenges, and open problems in the Internet of Things," in *Proc. IEEE World Congr. Services*, Jun. 2015, pp. 21–28.

[8] Y. B. Saied, "Collaborative security for the Internet of Things," Ph.D. dissertation, Institut National des Télécommun., Évry, France, Jun. 2013.

[9] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, Jul. 2017.

[10] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2014, pp. 417–423.

[11] K. Zhao and L. Ge, "A survey on the Internet of Things security," in *Proc. 9th Int. Conf. Comput. Intell. Security*, Dec. 2013, pp. 663–667.

[12] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[13] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Security Appl.*, vol. 38, pp. 8–27, Feb. 2018.

[14] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

[15] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1606–1616, Apr. 2019.

[16] Marketwired. (Jan. 2014). *Proofpoint Uncovers Internet of Things Cyberattack*. Accessed: Jun. 2019. [Online]. Available: https://www.proofpoint.com/us/proofpoint-uncovers-internet-things-iot-cyberattack

[17] P. Fremantle and P. Scott, "A survey of secure middleware for the Internet of Things," *Peer J. Comput. Sci.*, vol. 3, p. e114, May 2017.

[18] R. H. Weber, "Internet of Things—New security and privacy challenges," *Comput. Law Security Rev.*, vol. 26, no. 1, pp. 23–30, Jan. 2010.

[19] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.

[20] C. Pielli, D. Zucchetto, A. Zanella, L. Vangelista, and M. Zorzi, "Platforms and protocols for the Internet of Things," *EAI Endorsed Trans. Internet Things*, vol. 15, no. 1, p. e5, Oct. 2015.

[21] A. Singh, N. Chawla, J. H. Ko, M. Kar, and S. Mukhopadhyay, "Energy efficient and side-channel secure cryptographic hardware for IoT-edge nodes," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 421–434, Feb. 2019.

[22] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.*, 1999, pp. 388–397.

[23] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet of Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Oct./Dec. 2017.

[24] E. Ronen and A. Shamir, "Extended functionality attacks on IoT devices: The case of smart lights," in *Proc. IEEE Eur. Symp. Security Privacy*, Mar. 2016, pp. 3–12.

[25] W. Stallings, *Cryptography and Network Security Principle and Practice*, 6th ed. Upper Saddle River, NJ, USA: Pearson, 2014.

[26] M. Dworkin, "Recommendation for block cipher modes of operation—Methods and techniques," document SP800-38A, NIST, Gaithersburg, MD, USA, Nat. Inst. Stand. Technol., Dec. 2001.

[27] *Specification for the Advanced Encryption Standard (AES)*, NIST Standard FIPS 197, Nov. 2001.

[28] R. L. Rivest, A. Shamir, and L. M. Adleman, "Cryptographic communications system and method," U.S. Patent US4 405 829 A, Sep. 20, 1983.

[29] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," *Deep Space Netw. Progr. Rep.*, vol. 44, pp. 114–116, Jan. 1978.

[30] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[31] *Information Technology—Security Techniques—Message Authentication Codes—Part 1: Mechanisms Using a Block Cipher*, ISO/IEC Standard 9797-1:2011, Mar. 2011.

[32] E. Barker, *Digital Signature Standard (DSS)*, NIST Standard FIPS 186-1, Dec. 1998.

[33] R. C. Merkle, "A certified digital signature," in *Proc. Conf. Theory Appl. Cryptol.*, vol. 435, Jul. 1989, pp. 218–238.

[34] I. B. Damgård, "A design principle for hash functions," in *Proc. Conf. Theory Appl. Cryptol.*, vol. 435, Jul. 1989, pp. 416–427.

[35] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," Internet Eng. Task Force, RFC 7252, Jun. 2014.

[36] *Information Technology—Security Techniques—Lightweight Cryptography*, ISO/IEC Standard 29192-2012, Jan. 2012.

[37] J. Borghoff, "PRINCE—A low-latency block cipher for pervasive computing applications," in *Advances in Cryptology—ASIACRYPT*, X. Wang and K. Sako, Eds. Berlin, Germany: Springer, 2012, pp. 208–225.

[38] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf.*, Jun. 2015, pp. 1–6.

[39] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," in *Advances in Cryptology—CRYPTO*, P. Rogaway, Ed. Berlin, Germany: Springer, 2011, pp. 222–239.

[40] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "SPONGENT: A lightweight hash function," in *Cryptographic Hardware and Embedded Systems—CHES*, B. Preneel and T. Takagi, Eds. Berlin, Germany: Springer, 2011, pp. 312–325.

[41] Information Technology Laboratory. (2019) *Lightweight Cryptography Project*. Accessed: Jun. 2019. [Online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography

[42] H. Tschofenig and M. Pegourie-Gonnard, "Performance of state-of-the-art cryptography on ARM-based microprocessors," in *Proc. NIST Lightweight Cryptography Workshop*, Jul. 2015. [Online]. Available: https://www.nist.gov/news-events/events/2015/07/lightweight-cryptography-workshop-2015

[43] D. Eastlake, J. Schiller, and S. Crocker, "Randomness requirements for security," Internet Eng. Task Force, RFC 4086, Jun. 2005.

[44] K. Mandal, X. Fan, and G. Gong, "Design and implementation of warbler family of lightweight pseudorandom number generators for smart devices," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 1, pp. 1:1–1:128, Feb. 2016.

[45] A. B. O. López, L. H. Encinas, A. M. Muñoz, and F. M. Vitini, "A lightweight pseudorandom number generator for securing the Internet of Things," *IEEE Access*, vol. 5, pp. 27800–27806, 2017.

[46] M. Bakiri, C. Guyeux, J.-F. Couchot, L. Marangio, and S. Galatolo, "A hardware and secure pseudorandom generator for constrained devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3754–3765, Aug. 2018.

[47] B. Halak, M. Zwolinski, and M. S. Mispan, "Overview of PUF-based hardware security solutions for the Internet of Things," in *Proc. IEEE 59th Int. Midwest Symp. Circuits Syst.*, Oct. 2016, pp. 1–4.

[48] M. Roel, *Physically Unclonable Functions: Constructions, Properties and Applications*. Heidelberg, Germany: Springer, 2013.

[49] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2008, pp. 670–673.

[50] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.

[51] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, San Diego, CA, USA, Jun. 2007, pp. 9–14.

[52] Y. Nozaki and M. Yoshikawa, "Shuffling based side-channel countermeasure for energy harvester," in *Proc. IEEE 7th Glob. Conf. Consum. Electron.*, Oct. 2018, pp. 714–715.

[53] Y. Zhang, X. Zheng, and B. Peng, "A side-channel attack countermeasure based on segmented modular exponent randomizing in RSA cryptosystem," in *Proc. 11th IEEE Singapore Int. Conf. Commun. Syst.*, Nov. 2008, pp. 148–151.

[54] B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.

[55] F. Li, A. Shinde, Y. Shi, J. Ye, X. Li, and W. Z. Song, "System statistics learning-based IoT security: Feasibility and suitability," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6396–6403, Aug. 2019.

[56] F. Li, Y. Shi, A. Shinde, J. Ye, and W. Z. Song, "Enhanced cyber-physical security in Internet of Things through energy auditing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5224–5231, Feb. 2019.

[57] H. Sedjelmaci, S. M. Senouci, and T. Taleb, "An accurate security game for low-resource IoT devices," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9381–9393, Oct. 2017.

[58] J. Li, Z. Zhao, R. Li, and H. Zhang, "AI-based two-stage intrusion detection for software defined IoT networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2093–2102, Apr. 2019.

[59] G. Caparra, M. Centenaro, N. Laurenti, S. Tomasin, and L. Vangelista, "Wireless physical-layer authentication for the Internet of Things," in *Proc. Glob. Internet Things Summit*, Jun. 2017, pp. 390–417.

[60] *ZigBee Specification, Revision 20*, document 053474r20, ZigBee Alliance, Davis, CA, USA, Sep. 2012.

[61] *Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks*, IEEE Standard 802.15.4, Oct. 2003.

[62] T. Zillner, "ZigBee smart homes: A hacker's open house," in *Proc. CRESTCon Conf.*, May 2016. [Online]. Available: https://www.youtube.com/watch?v=eK-kI7Q9DE4

[63] *ZigBee Light Link Standard, Revision 1.0*, document 11-0037-10, ZigBee Alliance, Davis, CA, USA, Apr. 2012.

[64] P. Morgner, S. Mattejat, Z. Benenson, C. Müller, and F. Armknecht, "Insecure to the touch: Attacking ZigBee 3.0 via touchlink commissioning," in *Proc. 10th ACM Conf. Security Privacy Wireless Mobile Netw.*, Jul. 2017, pp. 230–240.

[65] L. Coppolino, V. DAlessandro, S. DAntonio, L. Levy, and L. Romano, "My smart home is under attack," in *Proc. IEEE 18th Int. Conf. Comput. Sci. Eng.*, Oct. 2015, pp. 145–151.

[66] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-ZigBee: Energy depletion attack on ZigBee-based wireless networks," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 816–829, Oct. 2016.

[67] *KillerBee*. Accessed: Jun. 2019. [Online]. Available: https://github.com/riverloopsec/killerbee

[68] J. Wright. *KillerBee: Practical ZigBee Exploitation Framework*. Accessed: Jun. 2019. [Online]. Available: https://www.willhackforsushi.com/presentations/toorcon11-wright.pdf

[69] *Bluetooth Core Specification 4.0*, Bluetooth SIG, Kirkland, WA, USA, Jun. 2010.

[70] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of Bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, Aug. 2012.

[71] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Uncovering privacy leakage in BLE network traffic of wearable fitness trackers," in *Proc. 17th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2016, pp. 99–104.

[72] R. Snader, R. Kravets, and A. F. Harris, III, "CryptoCoP: Lightweight, energy-efficient encryption and privacy for wearable devices," in *Proc. Workshop Wearable Syst. Appl.*, Jun. 2016, pp. 7–12.

[73] A. Y. Lindell, "Attacks on the pairing protocol of Bluetooth v2.1," presented at the Black Hat USA Conf., Jun. 2008.

[74] A. Y. Lindell, "Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1," in *Proc. Cryptograph. Track RSA Conf.*, Apr. 2009, pp. 66–83.

[75] J. Barnickel, J. Wang, and U. Meyer, "Implementing an attack on Bluetooth 2.1+ secure simple pairing in passkey entry mode," in *Proc. 11th IEEE Int. Conf. Trust Security Privacy Comput. Commun.*, Jun. 2012, pp. 17–24.

[76] M. Ryan, "Bluetooth: With low energy comes low security," in *Proc. 7th USENIX Workshop Offensive Technol.*, Aug. 2013. Accessed: Jun. 2019. [Online]. Available: https://www.usenix.org/conference/woot13/workshop-program/presentation/Ryan

[77] W. K. Zegeye, "Exploiting Bluetooth low energy pairing vulnerability in telemedicine," in *Proc. Int. Telemetering Conf.*, Oct. 2015, pp. 317–326.

[78] T. Rosa, "Bypassing passkey authentication in Bluetooth low energy," Cryptol. ePrint Archive, Rep. 2013/309, May 2013. Accessed: Jun. 2019. [Online]. Available: https://eprint.iacr.org/2013/309

[79] A. Hilts, C. Parsons, and J. R. Knockel, "Every step you fake: A comparative analysis of fitness tracker privacy and security," Open Effect, Canada, Rep., Apr. 2016. Accessed: Jun. 2019. [Online]. Available: https://openeffect.ca/reports/Every_Step_You_Fake.pdf

[80] O. Arias, J. Wurm, K. Hoang, and Y. Jin, "Privacy and security in Internet of Things and wearable devices," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 1, no. 2, pp. 99–109, Apr./Jun. 2015.

[81] M. Rahman, B. Carbunar, and U. Topkara, "Secure management of low power fitness trackers," *IEEE Trans. Mobile Comput.*, vol. 15, no. 2, pp. 447–459, Feb. 2016.

[82] A. Rose and B. Ramsey, "Picking Bluetooth low energy locks from a quarter mile away," presented at DEF CON 24, 2016. Accessed: Jun. 2019. [Online]. Available: https://www.youtube.com/watch?v=8h9nbMB1eTE

[83] Merculite Security. (2016) *Hacking Bluetooth Low Energy Locks*. Accessed: Jun. 2019. [Online]. Available: https://github.com/merculite/BLE-Security

[84] M. Fuller, M. Jenkins, and K. Tjølsen. (2017). *Security Analysis of the August Smart Lock*. Accessed: Jun. 2019. [Online]. Available: https://courses.csail.mit.edu/6.857/2017/project/3.pdf

[85] M. Ye, N. Jiang, H. Yang, and Q. Yan, "Security analysis of Internet-of-Things: A case study of August smart lock," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Atlanta, GA, USA, May 2017, pp. 499–504.

[86] S. Hall. (Mar. 2015). *Making Smart Locks Smarter (AKA. Hacking the August Smart Lock)*. Accessed: Jun. 2019. [Online]. Available: http://blog.perfectlylogical.com/post/2015/03/29/Making-Smart-Locks-Smarter

[87] *GATTack.io Outsmart the Things*. Accessed: Jun. 2019. [Online]. Available: https://gattack.io/#

[88] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," Internet Eng. Task Force, RFC 4944, Sep. 2007.

[89] G. Mulligan, "The 6LoWPAN architecture," in *Proc. 4th Workshop Embedded Netw. Sensors*, Jun. 2007, pp. 78–82.

[90] R. Alexander *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks," Internet Eng. Task Force, RFC 6550, Mar. 2012.

[91] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LoWPAN in IoT," in *Proc. Int. Conf. Pervasive Comput.*, Jan. 2015, pp. 1–6.

[92] A. Rghioui, A. Khannous, and M. Bouhorma, "Denial-of-service attacks on 6LoWPAN-RPL networks: Issues and practical solutions," *J. Adv. Comput. Sci. Technol.*, vol. 3, no. 2, pp. 143–153, 2014.

[93] A. Mayzaud, R. Badonnel, and I. Chrisment, "A taxonomy of attacks in RPL-based Internet of Things," *Int. J. Netw. Security*, vol. 18, no. 3, pp. 459–473, May 2016.

[94] A. E. Yegin and Z. Shelby, "CoAP security options," IETF Internet Draft, Oct. 2011. Accessed: Jun. 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-yegin-coap-security-options-00

[95] J. Mattsson and F. Palombini, "Comparison of CoAP security protocols," IETF Internet Draft, Jul. 2018. Accessed: Jun. 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-lwig-security-protocol-comparison-01

[96] K. Seo and S. Kent, "Security architecture for the Internet protocol," Internet Eng. Task Force, RFC 4301, Dec. 2005.

[97] E. J. Cho, J. H. Kim, and C. S. Hong, "Attack model and detection scheme for botnet on 6LoWPAN," in *Proc. Asia–Pac. Netw. Oper. Manag. Symp.*, Sep. 2009, pp. 515–518.

[98] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart nest thermostat: A smart spy in your home," presented at the Black Hat USA Conf., Aug. 2014. Accessed: Jun. 2019. [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home-WP.pdf

[99] R. Hummen, J. Hiller, H. Wirtz, M. Henze, H. Shafagh, and K. Wehrle, "6LoWPAN fragmentation attacks and mitigation mechanisms," in *Proc. 6th ACM Conf. Security Privacy Wireless Mobile Netw.*, Apr. 2013, pp. 55–66.

[100] A. Bertolaud *et al.*, "LoRaWAN 1.1 specification," LoRa Alliance, Fremont, CA, USA, Oct. 2017.

[101] M. Capuzzo, D. Magrin, and A. Zanella, "Confirmed traffic in LoRaWAN: Pitfalls and countermeasures," in *Proc. Annu. Mediterranean Ad Hoc Netw. Workshop*, Jun. 2018, pp. 1–7.

[102] M. Capuzzo, D. Magrin, and A. Zanella, "Mathematical modeling of LoRaWAN performance with bi-directional traffic," in *Proc. IEEE Glob. Commun. Conf.*, Dec. 2018, pp. 206–212.

[103] R. Miller, "LoRa security: Building a secure LoRa solution," Basingstoke, U.K., WR InfoSecurity, White Paper, Mar. 2016. Accessed: Jun. 2019. [Online]. Available: https://labs.mwrinfosecurity.com/publications/lo/

[104] *Cortex-M*, ARM, Cambridge, U.K. Accessed: Jun. 2019. [Online]. Available: https://www.arm.com/products/processors/cortex-m

[105] *Cortex–M Series Performance Graph*, ARM, Cambridge, U.K. Accessed: Jun. 2019. [Online]. Available: https://web.archive.org/web/20170828182449/

[106] H. Tschofenig and M. Pegourie-Gonnard. (Mar. 2015). *Performance Investigations*. Accessed: Jun. 2019. [Online]. Available: http://www.ietf.org/proceedings/92/slides/slides-92-lwig-3.pptx

[107] "Low power, 2.4GHz transceiver for ZigBee, RF4CE, IEEE 802.15.4, 6LoWPAN, and ISM applications," Datasheet AT86RF233, Atmel, San Jose, CA, USA, Jul. 2014. Accessed: Jun. 2019. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8351-MCU_Wireless-AT86RF233_Datasheet.pdf

[108] *Atmel SAM R21E / SAM R21G SMART ARM-Based Wireless Microcontroller Datasheet*, Atmel, May 2016. Accessed: Jun. 2019. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/SAM-R21_Datasheet.pdf

[109] *A True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications—CC2530 Datasheet*, Texas Instrum., Dallas, TX, USA, Feb. 2011. Accessed: Jun. 2019. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2530.pdf

[110] *EM351/EM357 High-Performance, Integrated ZigBee/802.15.4 System-on-Chip Datasheet*, Silicon Labs, Austin, TX, USA, Aug. 2013. Accessed: Jun. 2019. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/EM35x.pdf

[111] *nRF8001 Single-Chip Bluetooth Low Energy Solution Product Specification 1.3*, Nordic Semicond., Trondheim, Norway, Mar. 2015. Accessed: Jun. 2019. [Online]. Available: https://www.nordicsemi.com/?sc_itemid=%7B0EAB29C6-60BA-436D-AE00-2090F3AD62C4%7D

[112] *2.4-GHz Bluetooth Low Energy System-on-Chip—CC2540 Datasheet*, Texas Instrum., Dallas, TX, USA, Jun. 2013. Accessed: Jun. 2019. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2540.pdf

[113] *BlueNRG Bluetooth Low Energy Wireless System-on-Chip Datasheet*, STMicroelectronics, Geneva, Switzerland, Dec. 2017. Accessed: Jun. 2019. [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/group3/ac/c1/ad/80/54/fa/49/9d/DM00262983/files/DM00262983.pdf/jcr:content/translations/en.DM00262983.pdf

[114] *CC256x Dual-Mode Bluetooth Controller Datasheet*, Texas Instrum., Dallas, TX, USA, Jan. 2016. Accessed: Jun. 2019. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2564.pdf

[115] *CSR1010 QFN Datasheet*, Qualcomm, San Diego, CA, USA, Jan. 2015. Accessed: Jun. 2019. [Online]. Available: https://www.qualcomm.com/media/documents/files/csr1010-data-sheet.pdf

[116] *CC2538 Powerful Wireless Microcontroller System-on-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications Datasheet*, Texas Instrum., Apr. 2015. Accessed: Jun. 2019. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2538.pdf

[117] "SmartMesh IP node 2.4GHz 802.15.4e wireless mote-on-chip," Datasheet LTC5800-IPM, Linear Technol., Milpitas, CA, USA, Dec. 2015. Accessed: Jun. 2019. [Online]. Available: http://cds.linear.com/docs/en/datasheet/5800ipmfa.pdf

[118] "Long range, low power RF transceiver 860-1000 MHz with LoRa technology," Datasheet SX1272, Semtech, Camarillo, CA, USA, Mar. 2017. Accessed: Jun. 2019. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272

[119] R. Bonetto, N. Bui, V. Lakkundi, A. Olivereau, A. Serbanati, and M. Rossi, "Secure communication for smart IoT objects: Protocol stacks, use cases and practical examples," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw.*, Jun. 2012, pp. 1–7.

[120] W. Zhou and S. Piramuthu, "Security/privacy of wearable fitness tracking IoT devices," in *Proc. 9th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2014, pp. 1–5.
[121] R. Giuliano, F. Mazzenga, A. Neri, and A. M. Vegni, "Security access protocols in IoT capillary networks," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 645–657, Jun. 2017.
[122] H. Kim, "Protection against packet fragmentation attacks at 6LoWPAN adaptation layer," in *Proc. Int. Conf. Converg. Hybrid Inf. Technol.*, Aug. 2008, pp. 796–801.
[123] C. Lévy-Bencheton, E. Darra, G. Tétu, G. Dufay, and M. Alattar, "Security and resilience of smart home environments: Good practices and recommendations," Eur. Union Agency Netw. Inf. Security, Athens, Greece, Rep., Dec. 2015. Accessed: Jun. 2019. [Online]. Available: https://www.enisa.europa.eu/publications/security-resilience-good-practices. doi: 10.2824/360120.

Francesca Meneghello (S'19) received the B.Sc. degree in information engineering and the M.Sc. degree in telecommunication engineering from the University of Padova, Padua, Italy, in 2016 and 2018 respectively, where she is currently pursuing the Ph.D. degree with the SIGNET Research Group, Department of Information Engineering, under the supervision of Prof. M. Rossi.

Her current research interests include deep learning architectures and signal processing with application to remote radio frequency sensing and wireless networks.

Ms. Meneghello was a recipient of the Best Student Paper Award at WUWNet 2016 and the Best Student Presentation Award at the IEEE Italy Section SSIE 2019.

Matteo Calore received the B.Sc. degree in information engineering and the M.Sc. degree in telecommunication engineering from the Department of Information Engineering, University of Padova, Padua, Italy, in 2016 and 2018.

For several years, he has collaborated with the SIGNET Research Group, University of Padova, on many projects. His current research interests include next generation of cellular networks (5G) and machine learning in the domain of low-powered devices.

Daniel Zucchetto (S'17–M'19) received the B.Sc. degree in information engineering in 2012 and the M.Sc. degree in telecommunication engineering from the University of Padova, Padua, Italy, in 2012 and 2014, respectively. He received the Ph.D. degree from the University of Padova with a thesis on solutions for large scale, efficient, and secure Internet of Things (IoT).

He visited Telenor Research, Fornebu, Norway, in 2016 and Intel Labs Europe, Dublin, Ireland, in 2019. Since April 2019, he has been a Senior Systems Engineer with the Centre for Intelligent Power, Eaton Corporation, Dublin. His current research interests include smart energy systems, low-power wide area network technologies and next generation cellular networks (5G), with particular focus on their application to the IoT.

Michele Polese (S'17) received the B.Sc. degree in information engineering and the M.Sc. degree in telecommunication engineering from the University of Padova, Padua, Italy, in 2014 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering, under the supervision of Prof. M. Zorzi.

He visited New York University (NYU), New York City, NY, USA, in 2017, AT&T Labs, Bedminster, NJ, USA, in 2018, and Northeastern University, Boston, MA, USA, in 2019. He is collaborating with several academic and industrial research partners, including Intel, Santa Clara, CA, USA, InterDigital, Wilmington, DE, USA, NYU, AT&T Labs, University of Aalborg, Aalborg, Denmark, King's College London, London, U.K., Northeastern University, and NIST, Gaithersburg, MD, USA. His current research interests include analysis and development of protocols and architectures for the next generation of cellular networks (5G), in particular for millimeter-wave communication, and in the performance evaluation of complex networks.

Mr. Polese was a recipient of the Best Journal Paper Award of the IEEE ComSoc Technical Committee on Communications Systems Integration and Modeling 2019 and the Best Paper Award at WNS3 2019.

Andrea Zanella (S'98–M'01–SM'13) received the Laurea degree in computer engineering and the Ph.D. degree in electronic and telecommunications engineering from the University of Padova, Padua, Italy, in 1998 and 2001, respectively.

In 2000, he was a visiting scholar with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA, USA. He is an Associate Professor with the Department of Information Engineering, University of Padova. He is one of the coordinators of the SIGnals and NETworking (SIGNET) Research Group. His long-established research activities are in the fields of protocol design, optimization, and performance evaluation of wired and wireless networks.

Dr. Zanella is a Technical Area Editor of the IEEE INTERNET OF THINGS JOURNAL, and an Associate Editor of the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and *Digital Communications and Networks*.