IoT: Internet of Trust Analysis of some security vulnerabilities in real IoT devices

Matteo Calore, Francesca Meneghello
Department of Information Engineering, University of Padova – Via Gradenigo, 6/b, 35131 Padova, Italy
Email: {calorema,meneghello}@dei.unipd.it

***Abstract*—In the fast expansion of e-World, the Internet of Things (IoT) paradigm is taking a leading role in people's everyday life. Starting from personal health care and environmental monitoring, to smart vehicle and buildings passing through the concept of industry 4.0 with automatic production and assembly line management, everything and everyone will be connected to the Internet network in a few years. In this scenario, security, privacy and related issues have become very hot topics for both researchers and manufacturers due to the fact that the first goal of IoT devices is to achieve trustful communications of sensible information and then they need to be resistant towards attacks. In this paper an overview of the actual implemented IoT protocols will be presented, focusing on security constraints and related issues. To better understand feasible security bugs and fails, some attacks made against real implementations of IoT devices available on the market will be analyzed. Actual protocols weaknesses will be reported and the importance of ensure security in IoT networks will be discussed.**

I. Introduction

Nowadays the heterogeneous nature of *Internet of Things* (IoT) architecture, that involves a very wide variety of entities each one with different resource capabilities, makes it challenging to provide *end-to-end* secured connections. The number of IoT devices has grown rapidly, with a recent estimate suggesting that there were 12.5 billions Internet attached devices in 2010 and a prediction of over 50 billions devices by 2020 [1]. These devices are becoming more and more useful and often essential to people's everyday lives: this is why everything that concern the topic of security and privacy is becoming day by day more discussed. For what concerns *end devices* in fact, by intentionally exposing pieces of their personal information, users could benefit from complex dedicated services and ad hoc enhanced interactions. However, as new technologies are deployed, a big portion of the users is still unsure of how to correctly and trustfully interact with the IoT world. Some of them don't even knows if they really want to. This new developing scenario since few years made security a very hot topic not only for academic researchers and scientists: IoT manufacturers too nowadays must take the correct time and diligence to embed security into the architecture, interfaces, and designs of their products. The caseworks analyzed by researchers have highlited that most of the IoT manufacturers have been aware of the privacy and security implications: up to now in fact, security in IoT devices production has been either neglected or treated as an afterthought. Two main causes have been individuated: first of all the very short time to market given to engineers to finish a product and then the imposed reduction of costs driving the device's design and development process due to the strong competition.
In the general IoT paradigm, security is not an easy and well defined field of analysis: the direct use of the traditional exchange schemes and security techniques in most of the cases may be unfeasible because of the heterogeneity of the new concept of IoT networks. As it will be treated below with some detailed examples, IoT devices have non negligible limitations and constraints. Due to the small batteries sizes, low power calculation (low power CPUs then low clock rate) and limited physical and RAM memories, the devices may not be able to support same levels of security that would be expected from the traditional Internet connected devices [2]: for obvious reasons these imply software limitations too. As well explained in [8], in addition to device's constrains there are network limitations too: mobility, scalability, multi-protocol networking are some of the main problems because of the day by day growing number of mobile entities.
More than limitations, in order to better understand security in IoT world, new challenges and main purposes at different layers of the communication [3] [5] have to be clarified. In a general overview of security, the first layer is the one related to the *Information level* and it takes care of:

- ***Integrity***: it ensures that any received data has not been altered in the transmission.
- ***Anonymity***: it hides the source of the data during the exchange.
- ***Confidentiality***: it refers to the preservation of the on-air information: a trustworthy relationship is established between IoT devices in order to exchange protected and appropriate information. The freshness of the message must be guaranteed too.
- ***Client privacy***: it contains all the necessary measures that only the information provider is able to infer. This is possible by observing the use of the lookup system related to a specific customer. At least, interference made by eavesdroppers should be very hard to conduct.

After the *Information level* there is the *Access level* which main security requirements are:

- ***Access control***: providers must be able to implement access control on provided data. It must be guaranteed that only valid users get access to the devices and to the networks for administrative tasks (e.g. remote reprogramming or controlling of the IoT devices and networks).

- ***Authentication***: it is highly relevant to IoT and is likely to be the first operation carried out by a node when it joins a new network, for instance, after mobility [6]: devices have to guarantee trusted authentications to multiple networks securely. To better understand how this topic is essential, it is possible to make an example easy to understand: if Windows came with a default login password for every system, it would create a security nightmare as many users would never change it and attackers would login as default user. So, as first and foremost instance, IoT systems must be able to authenticate its owners. [10]
- ***Authorization***: it ensures that only the authorized devices and the users get access to the network services or resources. Nodes access only what they are authorized and nothing more.

At the end, security requirements at the *Functional level* are:

- ***Resilience***: it could be defined as the network capacity to ensure security control over the devices belonging to it. More than that, in case of attack, the system has to avoid single points of failure and should adjust itself to node failures. For example, IoT entities have to be designed to operate autonomously in the field with no backup connectivity if primary connection is lost.
- ***Self organization***: IoT devices have to provide a minimum of service in the presence of power loss or failures; if for some reasons something fails or one runs out of energy, the remaining or collaborator devices should have the ability to reorganize themselves in order to guarantee a fixed minimum level of security.

Attacks against IoT devices are often simple and easy to conduct as will be shown further. The aims of the attacks could be very different one to another. They could be performed in order to break user privacy and leak personal sensible information: the collected information can range from a simple heart-rate sensor, to temperature and humidity data, to the location of the user himself and his living habits. As shown in [7], the common attack strategy is to compromise one device in the IoT network, use it as beachhead to other nodes within the network and perform fraudulent acts towards another connected object, impersonating the real one. A compromised IoT device can be utilized to further attack other units in an unsuspecting victim's network. To hackers, every IoT devices is a potential robotic soldier which they might be able to recruit into their bot-army. As explained, the IoT world requires special attention in security at every layer of the traditional communication stack, from the *Edge Layer* (physical) to the *Application Layer* passing through the *Access Gateway Layer* and the *Middleware Layer* that are the two layers that effectively link and intermediate between the IoT world and the standard Internet [9]. An accurate taxonomy of attacks in IoT world could be made according to these different layers:

- ***Edge Layer***: main attacks at this layer are *Hardware Trojan* and *Denial of Service* (DoS), attacks that attempt to make resources unavailable to its intended user (e.g. battery drain, sleep deprivation, outage attacks). Even the package must be tamper resistant because of at least two reasons: devices (e.g. environmental sensor) could deployed in remote regions and left unattended: attackers can extract the cryptographic secrets, modify programs, or replace them with malicious nodes (*camouflage*); on the other hand for everyday devices (e.g. fitness trackers) must be complex to apply reverse engineering.
- ***Middlware Layer***: at this level the main attacks are *eavesdropping* (also called *sniffing*) and injection of fraudulent packets and unauthorized conversations. Even Routing attacks have to be taken into account: an attacker may use this kind attacks to spoof, redirect, misdirect, or drop the packets at the communication layer.
- ***Application Layer***: attacks at application layer are quite different from the previous ones and they are more on the firmware side: integrity attacks against machine learning where attacker manipulate the training process and attacks at login and authentication phases could be performed.

Taking care of all this topics, in [3] and [4] authors present an in-deep clear analysis where they discuss some of the major vulnerabilities presented above proposing solutions at different layers, from the device side to the cloud services.

The rest of the paper is organized as follow: in II are analyzed four of the most implemented protocols in IoT devices with particular focus on the security aspects. In III are presented IoT security threats: some of them from a theoretical point of view and some others through the analysis of attacks accomplished against IoT devices now available on the market. Finally, in IV the conclusions of the work are drawn.

II. Protocols for IoT applications: general overview and focus on security

As discussed in I, for the IoT world, traditional protocols for Internet connection are not recommended due to the limited resources of the devices. Therefore some constrained protocols have been specifically developed for this task. In this section three of the most deployed protocols will be presented, focusing on their security aspects.

A. ZigBee - protocol and ZLL application profile

ZigBee is a two-way, wireless communication standard developed up to the two lower layers of standard IEEE 802.15.4-2003 by the ZigBee Alliance [11]. Thanks to its low-cost and low-power-consumption, ZigBee is one of the most used standard to connect IoT devices. ZigBee stack architecture is made up of four layers (Figure 1):

- ***Application Layer*** (APL): it is divided into two more specialized sub-layers:
  - *Application Support Sub-Layer* (APS): it provides data transmission service, security services and binding of devices between two or more application entities located on the same network.
  - *ZigBee Device Objects* (ZDO): it is responsible of the initialization of the APS, the Network Layer (NWK) and the security server provider.

- ***Network Layer*** (NWK): some of the functionalities provided by this layer are routing, security and configuration of new devices. Services of establishing connections, joining and leaving networks, addressing and neighbor discovery are guaranteed by this layer too.
- ***Medium Access Control Layer*** (MAC): MAC controls access to the radio channel using *Carrier Sense Multiple Access - Collision Avoidance* (CSMA-CA) mechanism. It provides reliable communications between a node and its neighbors.
- ***Physical Layer*** (PHY): PHY supplies the interface to the physical transmission medium. It operates in two separate frequency ranges: 868/915 MHz (European and USA band respectively) and 2.4 GHz. 2.4 GHz *Industrial Scientific Medical* (ISM) band is divided into 16 channels each one of 2 MHz. From the point of view of the Physical Layer, data rate is very different band to band: starting from 20 kbps at 868 MHz, bit rate moves up to 40 kbps at 915 MHz. It is also possible to achieve 250 kbps at 2.4 GHz. The coverage range varies from 10 to 75 m.

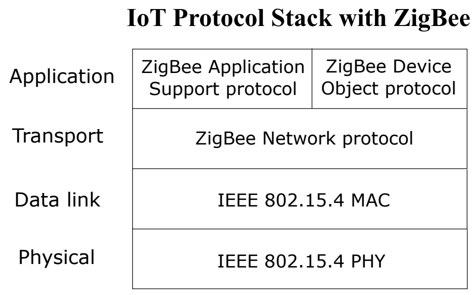
IoT Protocol Stack with ZigBee
Application
ZigBee Application Support protocol
ZigBee Device Object protocol
Transport
ZigBee Network protocol
Data link
IEEE 802.15.4 MAC
Physical
IEEE 802.15.4 PHY

Figure 1: ZigBee protocol stack.

ZigBee protocol includes different *application profiles*. The collection of descriptions from different devices allows the creation of a cooperative application. Application profiles establish agreements for messages, message formats and processing actions that developers have to respect in order to create interoperable and heterogeneous applications. In such a way devices from various vendors are able to communicate each other in a ZigBee network [11].

For what concern security, in addition to the security measures already improved by the IEEE 802.15.4-2003 standard at PHY and MAC Layers, ZigBee standard includes frame security services implemented at NWK and APS Layers. Frame security in ZigBee is based on two different type of keys: *link key* and *network key*, each of them of 128-bits. The first type of key is used to ensure secure unicast communications between APL entities, the second one for broadcast communications. For security purpose ZigBee network includes the *Trust Center*, a device trusted by all the other nodes in the network in order to manage keys control.

- ***Security at NWK Layer***: frame protection at NWK level is achieved by using *Advanced Encryption Standard* (AES) and *Counter with Cipher block Chaining Message* (CCM).
- ***Security at APS Layer***: frame security at APS level is ensured using both *link key* and *network key*. Keys can be acquired through three ways [11]:
  - *Pre-installation*: in general, it is made during the manufacturing process.
  - *Key-transport*: this service is offered by APS Sub-Layer and provides secured and unsecured means of transport keys to the devices. The secured transport-key command uses a key source for transport *link* or *network key*; on the other hand the unsecured transport-key command does not cryptographically protect the key exchanged and it is used for loading a device with an initial key.
  - *Key-establishment*: it is only for link key. Key-establishment services too are provided by the APS Sub-Layer. Key-establishment involves the Trust Center and one device of the network and it provides a *link key L_i* for secure communications between these two entities. This is the first step to achieve security in the network. The protocol is prefaced by the exchange of a trusted information, the *master key*, pre-installed during the manufacturing process. *Master key*, different for each application profile, is provided to ZigBee Alliance members how develop ZigBee products. After the preface phase, the devices exchange ephemeral data that are used to derive the *link key L_i*. When two devices *i* and *j* need to communicate each other, the Trust Center provides them with a *link key L_i,j*, protecting it using the *link keys LK_i* and *LK_j* respectively.

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*. Commissioning procedures are defined by the different application profiles standards. In addition to that, there is also a common procedure specified in the ZigBee standard that allows the connection between devices of different application profiles.

From now on, *ZigBee Light Link* (ZLL), the application profile used in lighting application domain [13] [12], will be taken into account. Differently from the standard protocol, in a ZLL network is not implemented a Trust Center so the *link key* is pre-installed by the manufacturers and it is bounded with a safekeeping contract. Devices in ZLL network use ZigBee NWK Layer security and they don't support APS Layer security. The ZLL node responsible for starting and allowing other devices to join a network trustfully is called *coordinator* [12]. When a device wants to join a network it starts the commissioning process by sending *beacon requests* on different channels. The coordinator of the ZLL network replies with a *beacon response* that contains the permission of joining and some information about the network. Then *network key*, encrypted using the device's *link key*, is sent to it. This process is called *classical commissioning*. In addition to that, in ZLL standard has been introduced the *touchlink commissioning* that offers also the possibility to manage network features with the touchlink commands. The initiator of the procedure is a ZLL device (already a member of the network, a controller or a bridge): it is characterized by a physical button that need to be pushed in order to start the commissioning procedure which leads to add a ZLL node to the network. The procedure starts with the transmission of a *scan request* by the initiator.

When the end device (i.e. the device that shall be added to the network) receives this message, it replies with a *scan response*. After that, the initiator sends to the device a *network join end device request* that contains the *network key* encrypted using the ZLL *master key*. Finally the end device replies with a *network join end device response*. Touchlink commissioning became insecure since 2015 when ZLL *master key* has been leaked on Twitter [14]. ZLL standard specifies also a fallback solution that allows the communication between ZigBee devices and other application profiles exchanging unencrypted messages: the login of the device in the network becomes in this case insecure. [12]. To ensure a higher level of security, the *network key* needs to be changed periodically otherwise attacks on the security of AES may be possible. For example with *plaintext attack* it is possible to obtain the key of a communication having both the encrypted and the decrypted messages [15]: if the *network key* never changes, an attacker can access whenever he wants in the network and eavesdrop sensible information.

B. BLE - protocol

Bluetooth is a short range connectivity protocol already used in 9 billion devices. *Bluetooth Low Energy* (BLE), the one adopted by the IoT world, is the latest tecnology defined by the *Bluetooth Special Interest Group* (SIG) WG in the Bluetooth Core Spec 4.0 [18]. The number of devices adopting BLE technology is expected to grow over 2.9 billion per year by 2016 [19]. BLE is a wireless protocol operating in the unlicensed 2.4 GHz ISM band (in detail, the frequency band is $2400 - 2483.5$ MHz) and it utilizes 40 RF channels with 2 MHz spacing. BLE must be robust against 2.4 GHz ISM band interference: Wi-Fi, 802.15.4, $X10$ (a specfic protocol defined for communication among electronic devices used for home automation) and all proprietary wireless are in this band. The Physical Layer uses a *Gaussian Frequency Shift Keying* (GFSK) with modulation index around 0.5: this scheme allows the use of fewer advertising channel and then lower power consumption. The Physical Layer data rate is 1 Mbps. The coverage range is typically over various tens of meters [25]. The BLE MAC Layer is split into two parts, [20] *advertising* and *data communication*: 37 of the available channels are used during the transmission of data and the remaining 3 (2402, 2426 and 2480 MHz in the channel 1, 6 and 11) are used by unconnected masters and slaves to broadcast device information and establish connections.

- ***Advertising***: as the name suggests, this is the phase responsible of the device advertisement, discovery and of the connection establishment. For example advertising channels are the ones that slave devices use to announce their presence to the master device. Advertising packets contain *beacon* information like the MAC address, connectivity modes, transmission power. A master device upon receiving the advertising packet, if interested in initiating a connection, sends other packets in order to continue the handshake and to establish connection.
- ***Data Communication***: this is the phase of the actual information data transfer. The communication is carried out over the 37 data channels. When two devices are connected, meaningful data are normally sent out in bursts in order to save energy. At the end of the day BLE defines two different roles at the Link Layer when a connection is created: master and slave. In order to save energy, slaves are normally in sleep mode and wake up periodically to listen at the channel for possible packet receptions from the master. The master determines the instants in which slaves are required to listen, and thus coordinates the medium access by using a *Time Division Multiple Access* (TDMA) scheme. Link Layer connections use a *Stop and Wait* (S&W) flow control mechanism based on cumulative acknowledgments, which at the same time provides error recovery capabilities.
On top of that other layers such as *Logical Link Control*, *Adaptation Protocol* (L2CAP) and *Low Energy Attribute Protocol* (ATT) are built in order to manage and apply the rules of connection and data flow communication (segmentation and reassembly techniques are not implemented since upper layers provide data units that fit the maximum L2CAP payload size of 23-bytes in BLE).

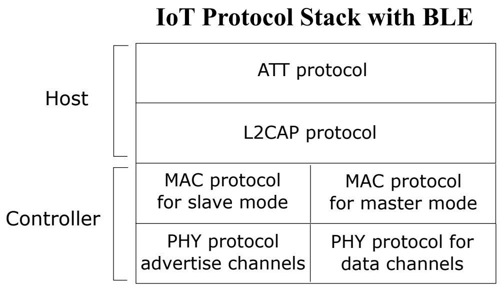
IoT Protocol Stack with BLE
Host
Controller
ATT protocol
L2CAP protocol
MAC protocol for slave mode
MAC protocol for master mode
PHY protocol advertise channels
PHY protocol for data channels

Figure 2: BLE protocol stack.

Whit regards to security, BLE supports two mutually-exclusive security models called *LE Security Mode 1* at the Link Layer and *LE Security Mode 2* at the ATT Layer. For encryption and authentication BLE uses AES-128 (symmetric encryption) and Key Distribution based on CCM algorithm to share various keys with public key exchange [21]. It is also possible to transmit authenticated data over an non encrypted Link Layer connection. In this case, a 12-bytes signature is placed after the data payload at the ATT Layer. If the receiver verifies the signature, it assumes that the data have been sent by the trusted source.
Another interesting security feature of BLE is the one that allows a device to use private addresses and frequently change them: as will be explained later on this paper with product ready available in the market, it is no actually used by the majority of manufacturers. A *paring* (key generation and distribution) procedure could be required. The *Security Manager Protocol* (SMP) operates on top of a fixed L2CAP channel and carries out the paring messages exchange. Despite that, as both [22] and [23] show, from the security point of view, BLE is vulnerable to several types of attacks at multiple layers: a vulnerability that currently exists in BLE is the fact that none of the actual pairing methods is protected against passive eavesdropping. In addition to that, conventional encryption protocols such as *Transport Layer Security* (TLS) or *HyperText Transfer Protocol over Secure Socket Layer* (HTTPS) are impracticable in wear-

able devices equipped with BLE: traditional protocols, in fact, relay on asymmetric cryptographic operation which require significant computation: experiments made in [24] show as the computation cost of a single ECDH-ECDSA key exchange required more than 6.000 times as much energy (236 mJ vs. 38 μJ) as encrypting a block using symmetric operations.

C. 6LoWPAN - protocol

The introduction of IPv6 allowed simple integration of IoT devices in a *Low-power and Lossy Network* (LLN). *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN) is an IPv6 adaptation protocol that defines mechanisms to make IP connectivity viable for tightly resource constrained devices that communicate over low power, lossy links such as IEEE 802.15.4. The Physical Layer data rate and the coverage area are the same of those already seen for ZigBee protocol in II-A. LoWPAN devices are characterized by short radio range, low data rate, low power and low cost: 6LoWPAN provides the connectivity between IPv6 network and resource constrained devices. It is based on pre-sheared keys for key exchange but it is not a feasible solution since sensors nodes should be able to communicate with external hosts without the need for prior authentication context [2]. Most of the work is carried out by the protocol at the *Adaptation Layer*: the IETF proposed as a solution to add a layer between the Network Layer (IPv6) and the Data Link Layer (802.15.4 MAC which provides access to media using CSMA-CA scheme). The main goal of 6LoWPAN is to optimize IPv6 packets by the fragmentation and reassembly mechanism and to remove duplicated information that can be derived from other layers. Even if IPv6 minimum MTU size is 1280-bytes, fragmentation avoids packet split in the way of having the maximum MTU size of only 127-bytes [31]. As will be treated below this could be a non negligible feature from the security point of view. 6LoWPAN provides the stateless auto configuration: inside a 6LoWPAN network, devices generates automatically their own IPv6 address with *Stateless Address Auto Configuration* (SSA) in the bootstrapping; algorithms such as *Duplicate Address Detection* (DAD) to avoid the assignment of the same IP to different devices are implemented too. Even if nodes can generate automatically IPv6 addresses, 6LoWPAN networks are connected to others network simply using IP routers. As shown in Figure 3, Transport Layer does not commonly use *Transmission Control Protocol* (TCP) because of performance, efficiency and complexity reasons: it is a connection-based protocol and has large overhead so it is not always suitable for IoT devices [29]. For what concern upper layers, a broadly used application on the Internet is HTTP but it runs over TCP. *Constrained Application CoAP* (defined by IETF in RFC 7252 [30]) is a valid alternative that runs over *User Datagram Protocol* UDP: it is easy to map to HTTP via proxies, defines retransmission, support sleepy devices and subscription, blocks transfers and resource discovery. The usage of UDP instead TCP lacks message reordering support and retransmission of lost packets.

6LoWPAN routing is based on *IPv6 Routing Protocol for Low-Power and Lossy Networks* (RPL) defined in RFC 6550 [38]; it is basically designed for the multi-point to point communication such as sensors of a *Wireless Sensor Networks* (WSNs) talking to router. However, it has to support also point to multi-point (sink broadcast) and point-to-point (leaf nodes communicating each other). This protocol has to allow different types of communication too: it has to be possible to communicate unicast, anycast and multicast. RPL builds a *Direct Acyclic Graph* (DAG) based on a root node called *Low power and lossy Border Router* (LBR), usually it is the border router responsible for management of a particular field of nodes and locate the junction between two networks. RPL topology forms the *Destination Oriented Direct Acyclic Graph* (DODAG) tree, which contain only one root and prevent any network loop in the tree constructed by *Distance Vector algorithm* (Figure 4).

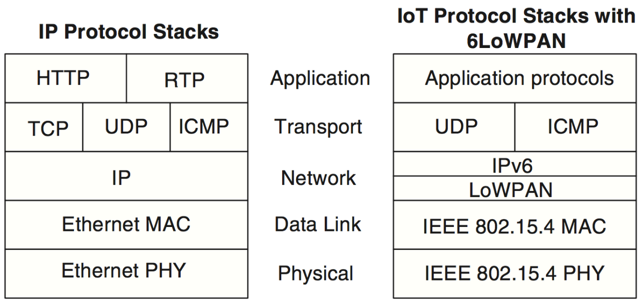

Figure 3: IP and 6LoWPAN protocol stacks.

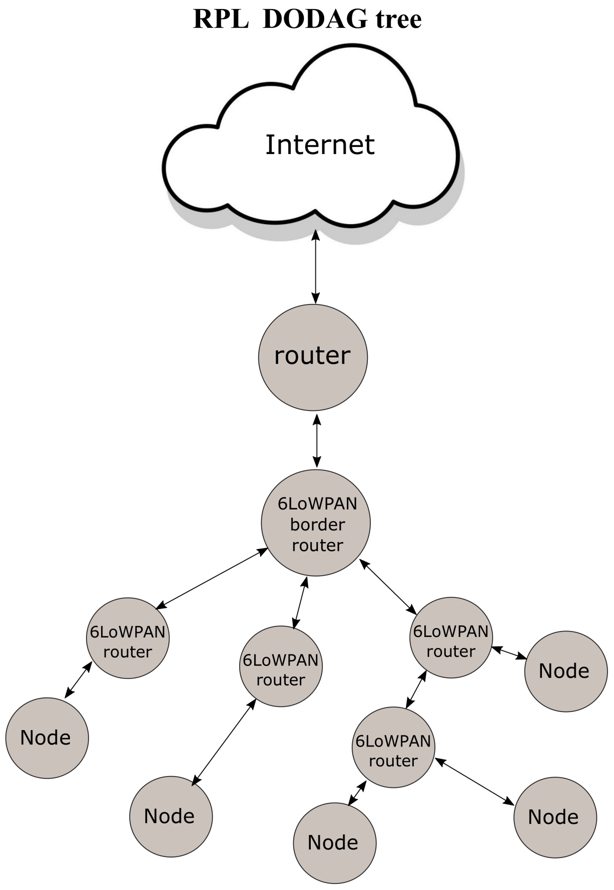

Figure 4: DODAG tree.

The choice of using a Distance Vector algorithm is logical as the use of *Link State algorithm* is almost impossible in this kind of networks because of power, memory and calculation constrains. Nodes in DOAG select and optimize the path using some node and link metrics and constrains called *instances* such as node state, throughput, latency, link reliability etc. The node uses the objective functions to point out particular metrics chosen for optimizing route, represented by *Objective Code Point* (OCP). The rank of the node is then calculated based on the objective functions and path costs toward the root. At the building phase, the DODAG root starts broadcasting its *DODAG Information Objective* (DIO) message, which contains information about its rank. All the neighbors of root node have direct connection: in this case their rank is set to 1 and root address is added as parent address. After update, rank 1 nodes continue broadcasting their own DIO: once a node receive a DIO it selects a preferred parent from the set of parent nodes. Iterating this algorithm to every node of the network, DODAG topology is created. A dual topology is obtained by the implementation of RPL protocol : the first one has a hierarchical structure thanks to the creation of a parental relationship; the second one is a mesh topology and allows routing between nodes that have same rank. *Global Repair* and *Local Repair* are used in case of a broken link: the first recalculate whole topology, the second is faster and the node who suffers broken link just sends the poison message to all its children informing that they need to update their parent.

D. *LoRaWAN - protocol*

Long Range Wide Area Network (LoRaWAN) was introduced in 2015 by LoRa Alliance. As the previous protocols, LoRaWAN is optimized for battery-powered end-devices. However, while ZigBee, BLE and 6LoWPAN protocols are designed specifically for short range communications in Local or Personal Area Networks, LoRaWAN is designed for long range communications in Regional or National Networks. The LoRa protocol is effective at IoT application where low data rate is required and poor power supply and network connectivity are expected. LoRaWAN presents a star-of-star topology that include end-devices, gateways and network server (Figure 5) [39].

Gateways are interconnected: in this way they belong to a common single network server via standard IP protocol. End-devices are connected to one or many gateways via a single-hop LoRa link. LoRa communication is spread out in different channels of the $868/900$ MHz ISM band. The data rate ranges from 0.3 kbps to 50 kbps and it is a trade-off between communication range (many kilometers can be achieved) and message duration. The communication is bidirectional and it is always initiated by the end device. After each *uplink window* the end device opens two *downlink windows* in different subbands, in order to protect the transmission against the fluctuations of the channel. ALOHA protocol is the one used for the transmission. At MAC Layer are defined three Classes: A, B and C. Class A functionality, the one described above, have to be implemented by all LoRaWAN devices. Class B and C offer some additionally options: Class B devices open extra receive windows at scheduled times thanks to the time synchronization made by beacon packets exchanged with the gateway; Class C devices have nearly continuously open receive windows. Beacon packet in Class B network is used also by the network server to know through which gateway a specific node can be reached.

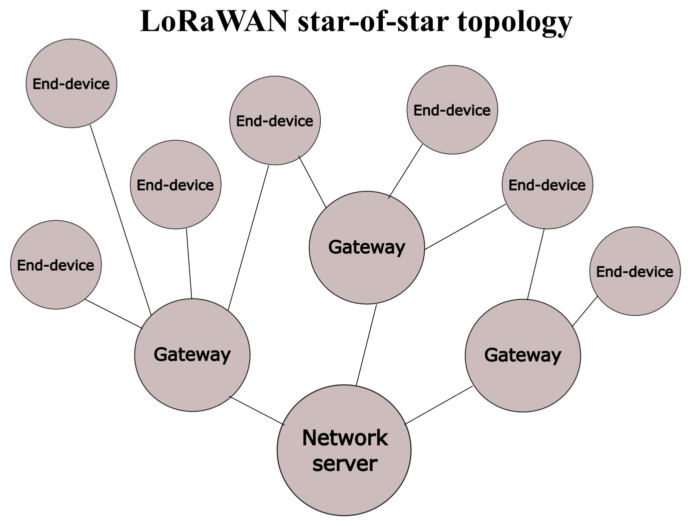

Figure 5: Star-of-star topology.

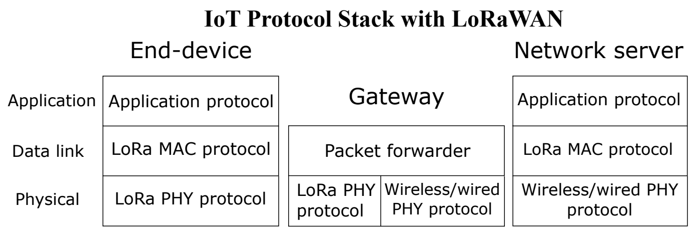

Figure 6: IP and LoRaWAN protocol stacks.

Over The Air Activation (OTAA) is the procedure by which a device can join a LoRaWAN. Three information have to be stored preliminarily in the device: the *End-device Identifier* (DevEUI), the *Application Identifier* (AppEUI) and the *Application Key* (AppKey). First two are global ID that uniquely identify the end-device and the application provider respectively. Application Key is an AES-128 key assigned by the application owner and specific for the end-device. The uniqueness of the key is needed for ensure security: if key is shared by more devices, it should be possible for an attacker to intercept traffic and eavesdrop it. The join procedure is initiated by the end-device by sending a *join request*, including DevEUI and AppEUI. If the device allows the permission 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). NetID and DevAddr are 32-bits long: the first uniquely identifies the network, 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). Session keys are specific for each end-device: the keys are used by them and

by the network server to encrypt and decrypt the payload data messages. NwkSKey is used for encrypt MAC commands, AppSKey for application specific messages. They are also used to verify the *Message Integrity Code* (MIC) to guarantee data integrity at MAC and Application Layers. MIC is a 4-bytes message obtained by the AES-128 encryption of the message with NwkSKey.
End-device can be activated also by a personalized procedure: NwkSKey, AppSKey and DevAddr are directly stored into the end-device differently from the DevEUI, AppEUI and AppKey. In this way joining procedure is by-passes.
Each end-device has also two frame counters: the *Frame Counter in the Uplink* (FCntUp) and the *Frame Counter in the Downlink* (FCntDown), incremented at each uplink or downlink communication respectively.

Once a end-node has joined the LoRaWAN all future messages will be encrypted using a combination of NwkSKey and AppSKey. Encryption of messages is performed using AES-128 standard. First of all, *A_i* blocks are created. The number of blocks is equal to the length of the payload divided by 16. Each of them is 128-bits long and contains the end-device address, the two counters and a byte for the stream direction. *A_i* blocks are then encrypted in the *S_i* blocks using the NwkSkey or the AppSKey according to the *Port Field* (FPort): 0 for NwkSKey, a value from 1 to 255 for AppSKey. Payload is finally encrypted by the XOR operation between the message itself and the sequence of blocks *S_i* [39].

III. Hacking the IoT world

In I a taxonomy of possible attacks against IoT devices has been presented: it has been treated from the point of view of the physical structure of the network. However, another analysis from a different point of view could be performed: looking at the way in which an attacker can exploit the IoT device in order to perform his attacks, four different approaches can be found [16]:

- ***Ignoring the functionality***: this class includes all the attacks in which the specific functionality of the IoT device is ignored. The attacker exploits only the ability of the device to connect to the *Local Area Network* (LAN) or to the Internet. For example, smart devices should be used to create a *bot-net* (a network completely controlled by attacker) or to penetrate the victim's home network and infect his computers. This type of attack is applicable to any device attached to the network, not only to IoT devices.
- ***Reducing the functionality***: the attacker tries to kill or limit the functionalities of the device, in order to annoy the victim or to create large scale chaos. The purpose of this type of attack is to infect the IoT device like smart TV or smart refrigerator, demanding for example the payment of money for restore their normal functionality.
- ***Misusing the functionality***: the normal functionality of the IoT device is used in order to create discomfort to its owner. For example an attacker could tamper a climate control turning off the heating in winter and on in summer. Another example could be the attack made infecting the smart light system: it is possible to switch on and off all the lights independently from the victim's actions.
- ***Extending the functionality***: the IoT device is used to achieve a completely different functionality.

In III-A III-B III-C III-D subsections some security bugs of IoT protocols presented above will be analyzed. The importance of ensure security in the IoT networks will be highlighted by reporting real attacks against purchasable devices.

A. ZLL - attack

Smart lights are LEDs connected to a LAN that allow users to manage colors and brightness using smartphone or computer applications. In 2012 LIFX and Philips presented their own first smart lights solutions and, afterwards, many other companies built their light systems to connect LEDs to the network. Many vendors, as Philips, for the system implementation use the ZLL application profile while others, such as LIFX, use 6LoWPAN protocol. In this section ZLL solution will be taken into account.

A general smart light system architecture is presented in Figure 7. A bridge, or a gateway or a hub, is used as ZigBee transceiver to allow the communication between bulbs and controllers. Smart bulbs can still be turned on and off using traditional light switches but it is preferable to command them remotely using the application installed on smartphones or computers.

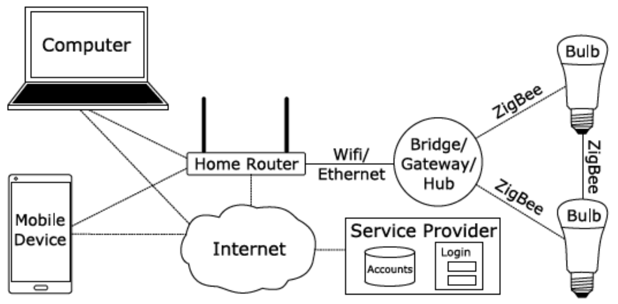

Figure 7: General architecture of a smart light system [14]

Many smart light systems security investigations have disclosed that designers and manufacturers implement only the essential security measurements that are necessary to obtain ZigBee Alliance's certificate. It may seem unnecessary to implement many precautions in a light system but, as explained above, it must be taken into account that all smart devices are connected to a network. Therefore they may be reached from an attacker with non-advisable purposes. To support this reasoning, the consequences of an attack against the connected lighting system of an hospital can be considered. This attack scenario in fact could force the interruption of surgeries or cause wrong treatments if the lights are use in order to exhibit medical parameters [14]. Attacks in different scenarios will be presented below.

In [14] the authors investigate the actual security in three different ZigBee smart light systems: *Osram Lightify*, *GE Link* and *Philips Hue*. The three systems have small differences one to another. Looking at Figure 7, while Lightify uses a gateway connected to the home router, Link uses a hub; however, both of them are connected to it via Wi-Fi. Differently from the previous two, Hue uses a bridge connected to the home router via Ethernet protocol. Authors evaluate vunerabilities of both bulbs and interconnected devices: this analysis is made up for seven different types of attack. The attacks are based on the *inter-PAN frames*: the frames used to transmit touchlink commissioning commands such as *scan request*, *scan response* an so on. These frames are neither secured nor authenticated: an attacker can send the same commands pretending to belong to the network.

- *Active device scan*. The scan searches for ZLL devices in wireless range achievable from the attacker, sending *scan requests* in different channels. Listening on the *scan response* in each channel, the attacker can obtain a complete overview of the devices connected to the network. The behavior of the three systems analyzed is different one to another. All light bulbs and the gateway from Lightify respond to the attacker's scan request; Link hub does not respond and finally Hue bridge responds only if its button is pushed within the last 30 seconds.
- *Blink attack*. This attack is established, after a device scan, by sending to the attacked device the inter-PAN command *identify request*. Such a command is implemented to allow the owner of the devices to identify one of them. The device indeed starts to blink for a default period. All the three lightbulb types are vulnerable to this attack.
- *Reset attack*. The attacker resets all setting of the ZLL device to the factory state, by sending the inter-PAN command *reset to factory new request* after having done the device scan. All devices are vulnerable to this attack.
- *DoS attack* and *hijack attack*. In these attacks the end user loses the control on the device. Two approaches are possible in order to make a DoS attack. The first is built up to force a device to change the transmission channel, sending a *network update request* inter-PAN command which includes the new channel. The purpose of the second approach is to force the device to join a non existing network, changing its network key with a series of arbitrary bytes. Hijack attack is similar to this second attack with the difference that it forces the device to join a new network chosen by the attacker. In this case the network key of the desired network is used. These two attacks are made up sending the inter-PAN command *network join end device request*: at the receiving of the command the device leaves its current network, changing its parameters according to the new configuration. All the smart light systems evaluated are vulnerable to DoS and hijack attacks but all of them integrate functions to regain control over attacked devices.
- *Network key extraction attack*. This attack allows the attacker to extract 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 a device from the network. After that, the attacked device will start a commissioning procedure in order to regain the access to the network. Therefore the attacker can extract the network key from the *network join end device request*. As seen in II-A, network key is in fact encrypted using the well known master key. Only Philips Hue lighting can be attack in this way because it is the only one that perform touchlink commissioning procedure.
- *Inject commands attack*. The purpose of this attack is to send commands to the devices of the network in order to control their actions. The knowledge of the current network key is needed. This is possible by a prior implementation of *network key extraction attack*, that extract the key, or *hijack attack*, that change the key of the network as the attacker wants. All the three smart light systems are vulnerable to this attack.

Philips Lux light system is the one used by Ronen and Shamir [16] to implement their attacks. Firstly they were able to establish a covert channel exploiting the existing interconnections both between different bulbs in the same network and then between the home router. Thankful to this covert channel, attackers can leak out sensitive data of the victims. The basic idea for the realization of the channel is to take advantage of the brightness of the bulbs in order to modulate and transport information. Philips Lux can provide the user with 256 brightness levels using *Pulse Width Modulation* (PWM). The signal sent from the controller (a bridge) to the bulb is a sequence of high and low voltage levels. Brightness is determined by the duty cycle duration (the portion of the total period in which the signal maintains high voltage). In order to not produce human detectable light fluctuations, PWM frequency is set at around 20 KHz. In color smart LED bulbs (like Philips Hue) a combination of different monochromatic LEDs is implemented: each one of them is controlled using the same PWM technique. At least two symbols are needed in order to modulate the information. Therefore to set up the covert channel the authors used two different PWM signals. The signals need to present similar duty cycles, in order to maintain the same perceivable brightness. In this way the user cannot detect that an attack is established in his network. However the difference between the two duty cycles has to be measurable. Looking at the implementation of the attack, two parts need to be built up: both the transmitting and receiving setup. The *transmitter* includes the smart light system (LEDs and bridge) and a PC used by the attacker to send command to the lightbulbs. In order to do that, the PC must be connected to the bridge. The connection can be achieved using a *network key extraction attack* as seen above in this subsection. In order to control the duty cycle and the frequency of the PWM signal, authors have found a specific command performing some tests to better understand how the different bytes change the behavior of the lightbulbs. For what concern the *receiver* side, the authors built a cheap, lightweight and portable setup using a light sensor (TAOS TC3200), the microcontroller Arduino and a telescope, with a total price of less than one thousand dollars. The optical receiver is able to detect slight changes in light intensity as small as 10 microseconds, using a sampling

rate of around 100 kHz. Employing this setup the authors were able to detect the two different symbols generated with different PWM signals, at a distance of 100 meters from the Philips Lux light. With this channel, more than 10 KB of data per day can be leaked. The following example should clarify the power of this attack. An organization can, in order to protect sensitive data, divide its internal network (Intranet) from the global Internet. Anyway, if it chooses to implement a smart light system connected to its internal network, the first security shrewdness became useless. Indeed an attacker should implement a covert channel and then eavesdrop sensitive information.
The second attack consists in the creation of an epileptic seizure with strobed light. The attack is implemented in the same way described above, forcing the smart bulb to flick between two close levels of brightness, at the top of the brightness range. This type of attack should be very dangerous and it could be directed at hospitals, schools and other public buildings that use connected LEDs.

In [15] the results of author's attacks against Philips Hue and *Osram Lightify* (and other ZigBee devices) show that link keys are not used or supported and key rotation too is not applied in this devices. This means that network keys are sent to the devices without encryptions and they are never changed. Therefore an attacker can easily obtain the access to the network. The procedure is the follow. As first step, the attacker causes the disconnection of a device from the network. At this point the device will try to regain the connection establishing fallback procedure (classical commissioning). During this operation the attacker can obtain the network key, sniffing the unencrypted information exchanged by devices. As seen, for this attack it isn't neither necessary physical access to the devices or knowledge of the secret keys.

Flynny et al. [17] have implemented an attack in which infections jump directly from bulb to bulb using only unmonitored and unprotected ZigBee communications. In this way it will be very difficult to detect that an attack is taking place and to locate its source. The attack spreads via physical proximity, disregarding the networking topology so it cannot be stopped by isolating infected subnetworks from the others. The process does not assume any prior knowledge about the attacked lights or about the ZLL master key and it can run simultaneously on all lights within range of the attacker. Authors have used RF transceiver CC2531 of Texas Instruments. The default software implemented on it is able only to sniff ZigBee messages. An ad hoc Python implementation of the ZLL stack provided them also a method to generate ZLL messages. The attack is based on the creation of a "worm" that can automatically spreads among physically adjacent lightbulbs in a chain reaction. First of all, the attacker have to impersonate his lightbulb, changing the version, MAC address and so on using the ZigBee *Over The Air* (OTA) upgrading cluster standard. An OTA image is created starting from a Python script, then translated in the hex code that the OTA standard requires to send over the air. OTA image should be stored in the external SPI flash by which the processor will read for the configuration. The worm attack is irreversible: there is no way to restore the initial funtionality of Philips Hue devices. The only possible solution is to replace the lightbulb with a new one. The second step of this attack is to generate a method by which the attacker's lightbulb is able to infect a lightbulb of the victim's network. For this purpose, the attacker's lightbulb must belong to the same network and share the same network key of the targeted lightbulbs. This can be achieve implementing a *network key extraction attack*, already presented above. The limit of the implementation is the proximity check of the touchlink commissioning. This procedure ensures that only a very close device (less than 1 m of distance) can force another device to reset to factory new or to join a new network. The control is implemented by a function that checks if the *Received Signal Strength Indication* (RSSI) of a request is above a certain manufacturer threshold, otherwise the request will be ignored. The authors discovered a bug in this procedure: the proximity check mechanism is performed only if the message is a *scan request* message, hence only if it has transaction or response IDs non zero. It is sufficient to set to zero the IDs of the sent message to ensure that it will be received and processed as a valid message. The attack is implemented in this way: infected attacker's bulb send a broadcast *reset to factory new request* message, using the network key of the victim's network extracted in the first step; bulbs that receive the message will undergo a factory reset and start scanning for a new network sending *scan request* message, such message is for sure sent by the infected bulb using its own network key; at the end of touchlink commissioning the attacked bulb joins the attacker's network. This process is replicates along all the channels available at 2.4 GHz range supported by ZLL in order to infect as much devices as possible. The attack works well at a distance of 50 m from the lightbulbs. Infected lights can be used to implement the attacks presented by Ronen and Shamir [16] described above in this subsection. In such a way their power increases.

Learning from this analysis, the design of a security architecture for smart light systems is not trivial. In everyday life, user often prefer to have devices with lower security protections rather than have incompatible devices that cannot be added to their home network. A critical point of the ZLL protocol is the commissioning procedure: the touchlink commissioning in fact become insecure after the leakage of the ZLL master key. Classical commissioning indeed looses its security if no link keys are used from the devices. The presence of a fallback mechanism is a critical point for security too: the messages exchanged during the procedure are unencrypted. In general the use of pre-installed key is not a good choice from a security point of view. In fact, such information can be leaked at every moment. The consequences of an attack against smart light should be not only to annoy users but also to limit or block some important activities by inducing a blackout. In addition to that, connecting smart lights to a critical network might expose this network to eavesdroppers. For this reasons, in the design of a ZLL network, a trade-off between security and functionality is needed. The design of the security architecture must be simple and straight at the same time. The coming ZigBee

3.0 standard is announced to replace ZLL profile, improving stronger security in connected bulbs networks.

B. BLE - attacks

Recent years have witnessed the introduction and use of several fitness tracker devices. Most of technological manufacturers and fitness brand companies have invested lot of money on this new products and the production of wearable devices in general is booming. While the specifications and the functionalities vary a lot among these devices, all of them measure some parameters. The main difference from the past generation is the ability to provide connection with others smart devices and Internet network in general.
Most of wearable devices and health care sensors implement BLE technology in order to communicate with master devices that, in the case of the wearable fitness tracker, are smartphones. In [20] authors found that fitness tracker are pretty always in advertise mode; this is due to the several times that master devices disconnect in order to preserve energy. When the app is quit or not running in the smartphone, the tracker terminates its communications and synchronization remaining in advertise mode until the next connection establishment. More than that, they found that 90% of the devices of their observation set (the ones from top leading manufacturers too) is not running the technology exposed in II-B of the changing MAC address: in this way there is possible to track strong correlation between the motion sensor and the pattern of the BLE traffic to its master device. In the literature analyzed in this paper only the *Apple Watch* randomizes the MAC address both when it is rebooted and in normal usage at an approximately 10 minutes intervals [27]. The continuous advertising by the fitness trackers using unchanged BLE addresses can be easily used by an attacker to track the movements of the BLE device owner (e.g. a mall visitor can be monitored as he move store to store and the shopping center is able to save location data or MAC addresses). The same feature should be used by a rubber as a simple motion sensor to see if there is anybody in a house. Even though the packets are encrypted, the volume of the data is a definite indicator of user's activity. The two plots in Figure 8, 9 will clearly explain this concept.

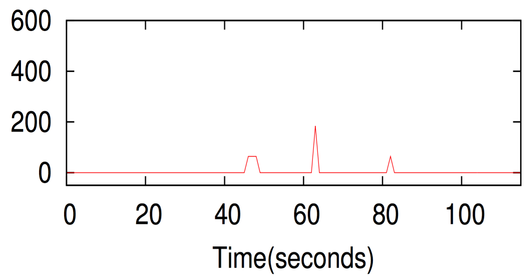

Figure 8: Data rate [bps] w.r.t time
Resting user [20]

In [26] they show another type of attack that could be conducted against wearable fitness devices. This one is based on the firmware's hacking and is possible if and only if an attacker has the physical access to the device. The analyzed device, a *Nike+ Fuelband*, contains a standard USB connector used

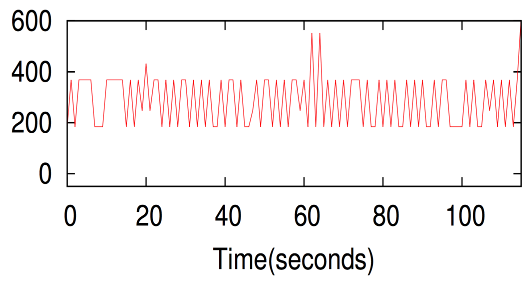

Figure 9: Data rate [bps] w.r.t time
Runnning user [20]

for charging and computer synchronization. This connector is used also by manufacturers to write firmware onto the device with special combination of pin connectors enabled. Using this mechanism, it is possible to send commands to the STM32 (the motherboard of the device) which allow read and write operations to the memory, changing memory protection modes and status retrieval. With the device's firmware management, it is possible to modify device's capabilities just flashing in the tampered custom version. If everyone is free to bypass any check on the update image, the protection mechanism becomes ineffective. Effects of firmware attack like the one to *Nike+ Fuelband* could rages from a simple back-door injection in order to leak user information or credentials, to the installation of rogue services with the aim of interrupt regular network operations (e.g. return false information from a fake server). *Address Resolution Protocol* (ARP) based attacks are possible with the compromised device acting like a router capturing and redirecting packets and network traffic.

Other studies in [27] show that in most of the cases fitness tracking applications transmit every logged fitness event over the Internet. This transmissions allows data analysis or sharing of personal and health data by eavesdropper and device selling companies too. In some case of analysis, is is unclear why transmission were occurring. On top of that, authors found that Garmin's devices don't use HTTPS protocol to everyday exchanges of data between the mobile device of the consumer and Garmins's servers (HTTPS is used only at the beginning in account creation). Taking that into account, with a *Man In The Middle* (MITM) *attack* is then possible to stole *sessionid*, *userid* and email address that are transmitted in clear-text. The same result is shown in [28] whit FitBit service logs: login credentials are forwarded in clear-text in a HTTP POST request sent from the base to the Internet server. All the sensitive data recorded by all these devices are vulnerable by third party surveillance or modification: for example, it is possible to manually change hart rate or the values of the distance traveled that are displayed on the smartphone application.

As exposed above, the actual vulnerabilities found on nowadays commercial IoT wearable devices such as fitness tracker are: clear-text login information and data processing, tracker data private capture, tracker injection attack and user account injection. Some attacks for DoS are possible too but they are not in our case study. In general, companies have the obligation to be clear with final customers on everything that concern privacy and security issues. To gain an higher level of security and privacy, BLE devices have to not be able to send sensitive and identifying information in clear-text and ensure that counters, when used, do not have the potential to be abused by a resourceful adversary. Mutual authentication of the fitness trackers and the web server has to be guaranteed as well as the randomize MAC address technique must be implemented in all devices even if battery duration is penalized. From the physical point of view micro-controllers and microchips must be inaccessible and any hardware debug interfaces must be detach or disabled. Micro-controllers must also be programmed before being placed in the commercial circuit board, to avoid adding unnecessary interfaces which could expose functionality.

C. 6LoWPAN - attacks

In this subsection some attacks against networks of IoT devices that implement 6LoWPAN protocol will be analyzed. As explained in II-C, in this protocol two different levels of connections should be taken into account: Internet and LLN [32]. Therefore an hypothetical attacker can operate in both of that two domains. Also Adaptation Layer and the specific routing protocol RPL seen above must be considered for the security analysis of the 6LoWPAN networks.

Attacks against the LLN. Attacks made in LLN can be threated at different layers, starting from the physical going up to the application layer [32]. Anyway, looking at security drawbacks of 6LoWPAN, routing is probably the weakest one. This is due to the fact that nodes are easy to be compromised while they are moving frequently in the network. For this reason most of the security threats concern the Network Layer. In the following taxonomy some attacks against the topology of the LLN will be reported [31] [33] [34].

- *Clone ID and sybil attacks*. In the clone ID attack the malicious node clones the identity of another node. In the sybil attack the attacker uses the identity of several entities at the same time. In this way the attacker can gain access to a large amount of network traffic. This types of attack can be detected keeping track of the number of instances of each identity or looking at the geographical location that should be different and unique for each identity.
- *Sinkhole attack*. The malicious node attracts to it a lot of traffic, by providing more efficient routing paths. After that it can achieved some other attacks to modify or drop the packets.
- *Selective forwarding and black hole attacks*. This attacks take place when a malicious 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 tries to pass through it. Possible solutions should be the creation of disjoint or dynamic paths inside the DAG or the usage of encryption techniques.
- *Hello flooding attack*. The message HELLO is used in the RPL network by a node to announce its presence. If a node receives an HELLO message, it understand that the sender node is in its neighbor so the link between them is available. An attacker can exploits this mechanism by sending a high power, broadcast HELLO message. In this way a large number of nodes consider it as a neighbor. However, when a node tries to use this new link, sent packets will be lost. In fact in order to be able to reach the imaginary neighbor, an higher power level than the one used by 6LoWPAN devices will be needed. This type of attack can be avoided using Link Layer acknowledgment to check the real presence of the connection.
- *Wormhole attack*. This attack takes place by the creation of an out-of-bound link between two malicious nodes, placed in the LLN network, that are not yet connected by the DODAG tree. Network topology is then disrupted. The traffic between the two malicious nodes will be transmitted through this new link, cutting out from the network the nodes that belonged to the correct path. A possible solution for this problem is to perform an authentication process by which each node is authenticated starting from the root.

All the previous attacks can be used in a general LLN. In the next part will be exposed some hacking procedures specifically designed for LLN that implement RPL routing protocol. This attacks in fact are based on the RPL service messages [31] [33] [34].

- *Rank attack*. In the DODAG tree the rank of the nodes strictly increases in the downstream direction. The nodes along the route have to check if this rule is observed. If a malicious node does not attempt at the condition of the rank value, some of the paths resulting from the routing should be not optimized: if this happens, the *Quality of Service* (QoS) decreases. Another consequence, for example, should be the creation of a loop that hinders the communications.
- *Local repair attack*. The attacker, a node without any problem of link connection, sends periodically *local repair message*. When a local repair happens, the network topology needs to be updated. Such operation has non negligible cost of resources and degrades all network operations.
- *Neighbor attack*. This attack is similar to the wormhole one but it uses specific messages of the RPL protocol. Malicious node that is not in the range of LLN, broadcasts high power DIO messages without any information about itself. The node who receives the message tries to select the attacker as parent node. The resulting link is then unavailable because the route is out of range.
- *Version number attack*. The *version number* is a field of DIO message incremented every time that a rebuilding of the DODAG have to be done. If an attacker forwards DIO messages to its neighbors after an illegitimate increasing of the version number, the whole DODAG graph should be unnecessary rebuilt. Successive unnecessary re-buildings cause the exhaustion of nodes resources and the congestion of the network.
- *Power exhaustion attack*. In RPL protocol there is no mechanisms to limit the actions that a node should do. In this way an attacker can reprogram a node in a way that it starts to process resource costing activities: that procedure

leads to the exhaustion of the resources of the node.

Attacks from the Internet side. Two threats derive from the connection to the Internet of a 6LoWPAN network. If no authentication mechanism is applied into the network, an attacker could access illegally to it and eavesdrop sensitive information from the data flow. On the other hand an attacker can gain the control of the sensor nodes, creating a bot-net [35]. The bot-net attack is established by the presence of a malicious node (bot) in the network that forges the data forwarded through it. The attack starts when an attacker from Internet sends to the bot a command to forge data. In this way the attacker can change a *non-alert* message into an *alert* message so an incorrect information arrives at the end user. For a practical example, it can be considered a LLN of temperature devices. If a bot-net is established the alert message can lead the end user to take an incorrect action with respect to the real temperature of the environment. This kind of attacks can be detected through an analysis of the node traffic: bot nodes have in fact to transmit more data than the other in order to maintain the bot-net. For the setup of the bot node will be needed physical access to the device. In [37] the authors have modified the default boot of a *Nest thermostat* by Google providing the capability of use it as a bot node.

Attacks at the Adaptation Layer. The translation of the packets between Internet and LLN is implemented at the border router by the fragmentation of IPv6 packets. The lack of authentication and the limited memory resource of the device in the LLN, make fragmentation mechanism vulnerable. Two attacks that can be achieved at this level, *fragment duplication* and *buffer reservation*, are presented in [36].
Duplication attack is based on the fact that a node cannot verify at the LoWPAN Layer if a received fragment belongs to the same IPv6 packet of the previous ones (this control is managed by higher layers). Therefore a malicious node inside the network can inject fragments with the same header of the legitimate LoWPAN packets in order to block the communication. The target node in fact cannot decide which fragments use during packet reassembly procedure since it cannot distinguish between legitimate and spoofed IPv6 fragments. Taking that into account, corrupted IPv6 packet will be dropped. In order to ensure the reliable delivery of some messages, higher layer protocols may retransmit lost packets: if fragmentation attack is implemented against the retransmission, large amount of device's energy resources will be wasted.
Buffer reservation attack targets the scarce memory of the LLN nodes. In the LoWPAN network, receiving nodes must reserve buffer space to reassembly the fragments that belong to the same IPv6 packet. When the reassembly buffer is assigned at one IPv6 packet, received fragments of other IPv6 packet are dropped out. The time reserved for the reassembly process of each IPv6 packet is 60 seconds. The attack is established transmitting one arbitrary fragment to a node: in this way the node is forced to allocate the buffer for a non-existing IPv6 packet. The attacker takes the buffer of the target node occupied and no additional fragmented packets can be processed. If the attack is continuously repeated, the communication will be definitely blocked.

In order to protect 6LoWPAN networks from the attackers, *Intrusion Detection System* (IDS) mechanisms are studied [31] [32] [33]. IDS is a network security approach based on the monitoring of network parameters able to identify signs of intrusions or attacks. The target of IDS that are involved in a 6LoWPAN network is to optimize features and computational work in order to save network resources. In this context IDS techniques should be implemented at the Internet and LLN sides. Therefore an hybrid architecture is needed in which centralized module (put on the border router) and distributed modules (installed on the sensor nodes) cooperate for attacks detection. In this way the detection of rank attack, sinkhole attack and selective forwarding attack will be possible.

At the end of the analysis of security in 6LoWPAN protocol should be clear that the amount of vulnerabilities at every layer force to make big efforts: an acceptable level of security has to be guaranteed. Most of the attacks exposed above can be used to establish DoS attacks against the network: this could be a very unpleasant, and in some cases dangerous, situation. Threats in a 6LoWPAN network arise both from Internet side and Local Network side. In detail, the characteristics of LLN networks such as resource constraints, dynamic topology and unreliable links make this kind of network particularly vulnerable. In an ideal context all the attack areas should be managed but the limited resources of a 6LoWPAN network force a trade-off between security and power save. However, the IDS must be chosen as the most harmful and dangerous attack: it must to be monitored in each specific implementation of 6LoWPAN network.

D. LoRa - attacks

The first weakness of the LoRaWAN protocol is the *key management* [40]. AppKey, NwkSKey and AppSKey are all stored in the end-devices. Using a *side channel analysis attack* it can be possible to eavesdrop the keys exploiting the variations in power consumption or electromagnetic emission from the transceiver during the encryption. NwkSKeys and AppSKeys are stored also in the Network Server, the one where they are generated: to ensure security, keys must be kept so that unauthorized parties cannot read or alter them. Another security threat arises when a node uses activation by personalization employing the same key that will be generated using the OTAA process. In this way an attacker, through reverse engineering on this node, can understand the way of generate the key: all the others communications would so be compromised. If the network server and nodes don't check correctly the FCntUp and FCntDown counters, *replay attack* can be built up: an attacker could record and play back multiple times the same message to the gateway. For example if the attacker is able to recognize a message of *alarm disable*, he can generate disruptions by continuously sending this type of message to the gateway. Moreover, if the counters are not updated, it means that the same AES-128 key is used for the encryption of more payloads. An attacker that know the plain-text of one message can derive the key through the XOR operation between the encrypted message and the plain-text. After this procedure, he is able to intercept in clear text all the other messages.

Also the *physical attacks* against LoRaWAN devices need to be considered. The aim of these attacks is to gain the control of a device through its tampering. In a WAN, physical attack can be achieved easier than in a PAN because the devices are installed in remote locations so owners cannot control them overall. A possible physical attack consists in the replacement of the microcontroller of the LoRaWAN node with a new one. In this case the attacker is not constrained to know the encryption keys: encryption process is managed by another component of the LoRaWAN node, the transceiver. In this way, an attacker is able to change the transmitted message of a node with a new one.

Some LoRaWAN solutions made the Network Server accessible by anyone who knows the IP address and the port used in order to be connected to the gateways. Therefore Network Server became an *Internet Facing Service* and an attacker can forge the traffic of gateway. Such attacks, that came from the server side, would be simpler to be detected using the IDS available for IT networks.

Class B networks introduce additional threats because of beacons and multicast messages [40]. Beacon messages are by default settings not encrypted: they represent a source of information and then a way to inject malicious data into the network. Beacon can also be generated by an attacker. The node in fact cannot distinguish between a malicious or a genuine beacon message: a beacon originated from the attacker is passed up at Application Layer and processed in the same manner of an authentic one. In actual implementations, nodes must share the same NwkSKey and AppSKey to exchange multicast messages. If an attacker is able to eavesdrop the keys from one node in the ways explained above, then all the multicast messages can be decrypted. In order to overcome the problems derived by this particular setting, the nodes need be able to differentiate between multicast and unicast communications. In the first case they must use the common key while for unicast communication they would use the secure NwkSKeys and AppSKeys.

IV. Conclusions

The aim of this work is to focus on actual security and privacy vulnerabilities of the latest IoT technologies and of devices already available on the market. Nowadays everything that goes around IoT security is becoming a very hot topic. This is due to the fact that nowadays, the Internet, could be compared to the central square of a big town: everyone, potentially, could see what you are doing, where you are going and people with which you are talking too. In the new e-world paradigm, IoT devices are the most powerful vehicles of personal and sensible data. More and more personal information are sent in clear-text continuously in user's everyday big data network flow and, in most of the cases, are sent without the total awareness of the end user. It is thus important, in the design and manufacturing process of a IoT device, to take into account security issues. Up to now, as shown in sections above, insufficient countermeasures have been taken, both in the protocol and device design. In the literature there are already some proposed improvements but not yet implemented in commercial devices: for example [24] proposes *CryptoCoP*, an encryption protocol for resource-constrained devices that supports energy-efficient symmetric key that could be implemented on BLE. In [41] is proposed another interesting solution in order 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 the constrained IoT devices under its purview. In [42] is shown how sometimes proposed bug fixes are not good at all: in this paper are discussed and analyzed some specific improvements of fitness tracker *Fitbit* proposed in other works. In [43] is proposed an algorithm for IoT connection establishment and key exchange between node and mediator based on the *timestamp*. The time-based secure key generation approach has the aim to efficiently manage, and renew, the keys to provide the trustful connection, while guaranteeing the integrity of data transmitted over an insecure channel. The timestamp option is also used to avoid fragmentation type of attacks in 6LoWPAN as explained in [44]: the value of timestamp field indicates the seconds passed since January 1, 1970 so it is unique for each packet. Author proposes also the *nonce* option to improve security: the field added to the message contains a random number selected by the transmitter.
In addition to everything that concerns with the privacy and information leakage of end users there are others big challenges in IoT networks: the extremely basic design that IoT devices require in every way and all the consequences of that make them very good targets for attackers. This requires big efforts aimed at the improvement of resistance and resilience of IoT sensors networks or even of traditional networks were IoT devices are employed. The *European Union Agency for Network and Information Security*, in [45], evaluates good practices to secure the life-cycle of IoT products and servers, looking at all the aspects of security 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.

The main objective of this paper is then to give the reader a wide-range analysis that takes into account both theoretical and practical points of view: somehow the reader has the opportunity to explore the IoT world from the protocol design to the real implementation. This analysis is built up for four of the most adopted actual technologies: differently from the first three (II-A, II-B, II-C) that are for the short ranges and already on the market, the last one (II-D) is for long range communications and is one of the latest proposed. For all of them some pro and cons have been discussed highlighting vulnerabilities, best features and possible future improvements. For all this reasons, this paper is addressed to industrial/academic research communities as well as manufacturers.

References

[1] Evans D., "The internet of things. How the Next Evolution of the Internet is Changing Everything", *Cisco Internet Business Solutions Group (IBSG)*, Apr. 2011. [Online]. Available: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[2] Yosra Ben Saied. "Collaborative security for the internet of things", *Economies and finances Institut National des Telecommunications*, 2013. English. <NNT : 2013TELE0013>. <tel- 00879790>

[3] Paul Fremantle, Philip Scott, "A security survey of middleware for the Internet of Things", PeerJ PrePrints 3:e1521, 2015. [Online]. Available: https://doi.org/10.7287/peerj.preprints.1241v1

[4] Arsalan Mohsen Nia, "A Comprehensive Study of Security of Internet-of-Things", in *IEEE Transactions on Emerging Topics in Computing*, vol. PP, Issue: 99, Sep. 2016.

[5] Rolf H. Weber, "Internet of Things New security and privacy challenges", *Computer Law & Security Review*, vol.26, pp.23-30, Jan. 2010.

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

[7] "Proofpoint uncovers Internet of Things cyberattack", Jan. 2014. [Online]. Available: http://investors.proofpoint.com/releasedetail.cfm?releaseid=819799

[8] Md. Mahmud Hossain, Maziar Fotouhi, Ragib Hasan, "Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things", *IEEE World Congress on Services*, Jun. 2015.

[9] Chiara Pielli, Daniel Zucchetto, Andrea Zanella, Lorenzo Vangelista, Michele Zorzi, "Platforms and Protocols for the Internet of Things", *EAI Endorsed Transactions on Internet of Things*, Oct. 2015.

[10] Intel. [Online]. Available: http://www.intel.it/content/www/it/it/homepage.html

[11] "ZigBee Specification - Document 053474r17", *ZigBee Alliance*, Jan. 2008.

[12] "Zigbee light link standard version 1.0 - Document 11- 0037-10", *ZigBee Alliance*, Apr. 2012.

[13] Zigbee light link. [Online]. Available: http://www:zigbee:org/zigbeefor-developers/applicationstandards/zigbee-light-link/

[14] Philipp Morgner, Stephan Mattejat, Zinaida Benenson, "All Your Bulbs Are Belong to Us: Investigating the Current State of Security in Connected Lighting Systems", *arXiv preprint* arXiv:1608.03732, Aug. 2016.

[15] Tobias Zillne, "ZigBee smart homes, a hacker's open house", *Cognosec*, Apr. 2016. [Online]. Available http://www.crestandiisp.com/wp-content/uploads/2016/03/TobiasZillner.pdf

[16] E. Ronen and A. Shamir, "Extended functionality attacks on iot devices: The case of smart lights", in *2016 IEEE European Symposium on Security and Privacy*, Mar. 2016.

[17] Eyal Ronen, Colin OFlynny, Adi Shamir, Achi-Or Weingarten, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction", 2016. [Online]. Available: http://iotworm.eyalro.net/

[18] "Bluetooth specification - Version 4.0 [Vol 0]", *Bluetooth SIG*, Jun. 2010.

[19] Mike Ryan, "Bluetooth with low energy comes low security", *USENIX, 7^th workshop on offensive technologies*, Aug. 2013.

[20] Aveek K. Das, Parth H. Pathak, Chen-Nee Chuah, Prasant Mohapatra, "Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers", in *HotMobile 16*, FL, USA Feb. 2016.

[21] Bluetooth.org. [Online]. Available: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=227336

[22] Sandhya S., Sumithra Devi K. A., "Analysis of Bluetooth Threats and v4.0 Security Features", in *2012 International Conference on Computing, Communication and Applications (ICCCA)*, Feb. 2012.

[23] John Paul Dunning, "Taming the blue beast: A Survey of Bluetooth Based Threats. Security & Privacy", *IEEE Security & Privacy*, vol. 8, Issue 2, Jan. 2010.

[24] Robin Snader, Robin Snader, Albert F. Harris III, "CryptoCoP: Lightweight, Energy-efficient Encryption and Privacy for Wearable Devices", in *WearSys '16 Proceedings of the 2016 Workshop on Wearable Systems and Applications*, pp. 7-12, Singapore, Jun. 2016.

[25] Carles Gomez, Joaquim Oller, Josep Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology", *Sensors 2012*, vol. 12, pp. 11734-11753, Aug. 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/9/11734

[26] Orlando Arias, Jacob Wurm, Khoa Hoang and Yier Jin, "Privacy and Security in Internet of Things and Wearable Devices", *IEEE Transactions on multi-scale Computing Systems*, vol.1 , no.2, Jun. 2015.

[27] Andrew Hilts, Christopher Parsons, and Jeffrey Knockel, "Every step you fake: a Comparative analysis of fitness tracker privacy and security", *Open Effect Report*, 2016. [Online]. Available: https://openeffect.ca/reports/Every_Step_You_Fake.pdf

[28] Mahmudur Rahman, Bogdan Carbunar, Madhusudan Banik, "Fit and Vulnerable: Attacks and Defenses for a Health Monitoring Device", *HotPETs 2013*, arXiv:1304.5672 [cs.CR], Bloomington, Apr. 2013.

[29] Jonas Olsson, "6LoWPAN demystied", *Texas Instrument*, 2014. [Online]. Available:http://www.ti.com/lit/wp/swry013/swry013.pdf

[30] "The Constrained Application Protocol (CoAP)", Jun. 2014. [Online]. Available: https://tools.ietf.org/html/rfc7252

[31] Pavan Pongle, Gurunath Chavan, "A Survey: Attacks on RPL and 6LoWPAN in IoT2, *International Conference on Pervasive Computing*, Jan. 2015.

[32] Anhtuan Le, Jonathan Loo, Aboubaker Lasebae, Mahdi Aiash, Yuan Luo, "6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach", *International journal of communication systems*, vol. 25, Issue 9, pp. 11891212, May 2012.

[33] Anass Rghioui, Anass Khannous, Mohammed Bouhorma, "Denial-of-Service attacks on 6LoWPAN-RPL networks: Threats and an intrusion detection system proposition2, *Journal of Advanced Computer Science & Technology*, vol. 3, no. 2, pp. 143-153, 2014

[34] Anthéa Mayzaud, Rémi Badonnel, Isabelle Chrisment2, "A Taxonomy of Attacks in RPL-based Internet of Things", *International Journal of Network Security*, vol.18, no.3, pp.459-473, May 2016.

[35] Eung Jun Cho, Jin Ho Kim, Choong Seon Hong, "Attack model and detection scheme for Botnet on 6LoWPAN", in *Management Enabling the Future Internet for Changing Business and New Computing Services*, Spinger, vol. 5787, pp. 515518, Sep. 2009.

[36] René Hummen, Jens Hiller, Hanno Wirtz, Martin Henze, Hossein Shafagh, Klaus Wehrle, "6LoWPAN Fragmentation Attacks and Mitigation Mechanisms", *WiSec13*, ACM, Budapest, Apr. 2013.

[37] Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin, "Smart Nest Thermostat: A Smart Spy in Your Home". [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home-WP.pdf

[38] "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6550

[39] "LoRa Specification", *LoRa Alliance*, 2015

[40] Robert Miller, "LoRa Security Building a Secure LoRa Solution", *MWR Labs Whitepaper*, Mar. 2016. [Online]. Available: https://labs.mwrinfosecurity.com/publications/lo/

[41] Riccardo Bonetto, Nicola Bui, Vishwas Lakkundi, Alexis Olivereau, Alexandru Serbanati, Michele Rossi, "Secure Communication for Smart IoT Objects: Protocol Stacks, Use Cases and Practical Examples", in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Jun. 2012.

[42] Wei Zhou, Selwyn Piramuthu, "Security / Privacy of Wearable Fitness Tracking IoT Devices", in *2014 9th Iberian Conference on Information Systems and Technologies*, Jun. 2014.

[43] Romeo Giuliano, Franco Mazzenga, Alessandro Neri, Anna Maria Vegni, "Security Access Protocols in IoT Capillary Networks", *IEEE Internet of Things Journal*, vol. PP, Issue: 99 Nov. 2016.

[44] HyunGon Kim, "Protection against Packet Fragmentation Attacks at 6LoWPAN Adaptation Layer", in *International Conference on Convergence and Hybrid Information Technology*, Aug. 2008.

[45] Dr. Cédric Lévy-Bencheton, Ms. Eleni Darra, Mr. Guillaume Tétu, Dr. Guillaume Dufay, Dr. Mouhannad Alattar, "Security and Resilience of Smart Home Environments", *European Union Agency for Network and Information Security*, Dec. 2015. [Online]. Available: https://www.enisa.europa.eu/publications/security-resilience-good-practices