A Survey on Secure Communication Protocols for IoT Systems

(Invited Paper)

Dan Dragomir*, Laura Gheorghe§, Sergiu Costea* and Alexandru Radovici*
*Computer Science and Engineering Department
University Politehnica of Bucharest
Bucharest, Romania
Email: {dan.dragomir, sergiu.costea, alexandru.radovici}@cs.pub.ro
§Research and Development Department
Academy of Romanian Scientists
Bucharest, Romania
laura.gheorghe@cs.pub.ro

***Abstract*—The Internet of Things (IoT) integrates a large number of physical objects that are uniquely identified, ubiquitously interconnected and accessible through the Internet. IoT aims to transform any object in the real-world into a computing device that has sensing, communication and control capabilities. There is a growing number of IoT devices and applications and this leads to an increase in the number and complexity of malicious attacks. It is important to protect IoT systems against malicious attacks, especially to prevent attackers from obtaining control over the devices. A large number of security research solutions for IoT have been proposed in the last years, but most of them are not standardized or interoperable. In this paper, we investigate the security capabilities of existing protocols and networking stacks for IoT. We focus on solutions specified by well-known standardization bodies such as IEEE and IETF, and industry alliances, such as NFC Forum, ZigBee Alliance, Thread Group and LoRa Alliance.**

***Keywords*-Internet of Things; security; standard; authentication; confidentiality; integrity.**

I. Introduction

The Internet of Things (IoT) represents the interconnection, through the Internet, of a large number of 'Things' – uniquely identifiable physical objects with sensing, communication and actuation capabilities. The term has been introduced by Kevin Ashton in 1999 in the context of chain supply management [1], [2], [3], [4], [5], [6].

There are currently 5 billion smart objects connected to the Internet, and it is expected that there will be 25 billion by 2020 [7]. The integration of 'Things' in the Internet is challenging because they may have characteristics such as limited memory, processing capacity and energy resources.

Most products were initially developed as closed proprietary solutions that were incompatbile with devices from other vendors [8]. The current trend however is towards standardized and interoperable protocols [9].

The number of IoT applications is growing. It includes smart home, healthcare monitoring, smart city, utilities, smart agriculture and animal farming, security and emergencies, smart water, industrial control, smart transportation, environment monitoring, etc. [10]. These IoT applications handle sensitive information regarding people and companies, which should not be disclosed to attackers and unauthorized persons.

As the field of IoT expands, attacks against IoT systems are growing in number and complexity [11]. Attacks against IoT systems aim to steal sensitive data, inject false information or disrupt the normal functionality of networks and services [3]. Recent attacks exploited vulnerabilities in smart refrigerators, in medical devices and smart cars [12]. Some attacks may involve considerable risk, for example, hacking medical devices may lead to the loss of human lives. Therefore it is important to ensure the security of critical IoT systems by providing protection against malicious attacks and failures.

In general, information security deals with confidentiality, integrity and availability (CIA) [1], [13], [14]. Schneier states that in the Internet of Things, attacks against integrity and availability are more important than attacks against confidentiality [12]. For example, in a smart home environment with a smart lock, it is more important to prevent an attacker from controlling the lock (to enter the house or block the door), than from finding out that someone has entered the house. In a similar manner it is more important to prevent an attacker from controlling your car, than from eavesdropping on your location. The main challenge in IoT security is to prevent attackers from obtaining control over the IoT system.

This paper presents a survey of the most used communication protocols for IoT and their security capabilities. Although many research solutions for IoT provide security, they are generally not standardized or interoperable. In this paper, we focus on standardized protocols and networking stacks because interoperability is important for the large-scale adoption of the IoT. We investigate solutions specified by industry alliances, such as LoRa Alliance, ZigBee Alliance, Thread Group, NFC Forum, and leading standardization bodies, such as IEEE and IETF.

This paper is organized as follows. Section 2 presents the

security requirements for IoT systems. Section 3 investigates the security capabilities provided by IoT communication protocols and networking stacks. Finally, Section 4 presents a comparative evaluation and Section 5 includes conclusions and future work.

II. Security Requirements

Vasilomanolakis et al. [9] classify security requirements in five categories: *Network Security*, *Identity Management*, *Privacy*, *Trust* and *Resilience*.

1) Network Security: Network security requirements include: *confidentiality*, *integrity*, *origin authentication*, *freshness* and *availability* [15].

In many IoT applications, such as healthcare or military applications, the sensitive data transmitted through the network should not be disclosed to unauthorized entities [11]. An attacker may eavesdrop on network traffic and extract sensitive information. Message *confidentiality* ensures that the contents of the message cannot be understood by anyone other than the desired recipients [15].

In critical IoT applications, the modification of sensitive data or the injection of invalid data may lead to the loss of human lives, for example in remote health monitoring systems [11]. An attacker may intercept network packets, modify their contents and inject them back into the network. In order to prevent the modification of messages by malicious or faulty devices, message *integrity* must be ensured.

Authentication refers to two different security requirements: entity authentication and data origin authentication. Data *origin authentication* ensures that a message originates from a certain entity [16]. In order to prevent the injection of invalid data by malicious external devices, the IoT system must provide data origin authentication.

An attacker may inject false information into the network by recording messages and replaying them. Message *freshness* ensures that attackers cannot reinject information.

Some IoT applications rely on real-time data collection and correct functionality of services. An attacker may disrupt the functionality of the IoT system by blocking network packets or by causing services to fail. *Availability* ensures that the devices and services are reachable and operating correctly whenever needed, in a timely manner [13]. Availability is directly related to resilience to attacks and failures.

This paper gives an overview of standardized and popular protocols that satisfy network security requirements.

2) Identity Management: Identity management represents an important challenge in IoT systems due to the complex relationships between entities (devices, services, service providers, owners and users) [9]. Identity management requirements include *authentication*, *authorization*, *revocation* and *accountability* [9].

Entity authentication refers to ensuring that an entity is who it claims to be [16]. More specifically, *device authentication* refers to verifying the unique and correct identity of the communicating devices in the IoT network [17].

Authorization allows authenticated entities to perform certain operations in the IoT system [11]. This means that each authenticated entity has permissions to perform specific operations. *Revocation* refers to removing the permission to perform an operation of a certain entity.

Accountability ensures that operations are clearly bound to authenticated entities. In large scale IoT systems, it is a challenge to provide accountability due to the considerable amount of devices, access delegation and multiple organizational domains [9].

3) Privacy: Privacy requirements refer to *data privacy*, *anonymity*, *pseudonymity* and *unlinkability* [9]. Privacy is an important challenge in IoT systems because users require the protection of their personal data because it provides information about their habits, interactions and location [18].

In IoT, the collected data may be Personally Identifiable Information (PII): data that identifies a person [19]. *Data privacy* implies that the collected data does not expose information about a person, for example its identity [9].

Anonymity means that a certain person cannot be identified as a source of data or action [20]. In some cases IoT applications need to comply to the data minimization laws [9].

Pseudonymity is a tradeoff between anonymity and accountability, as it links the data and actions to a pseudonym instead of a person [20]. *Unlinkability* means that the data or actions related to the same person cannot be linked together [9].

4) Trust: Trust requirements deal with *data trust* and *entity trust* [9]. Other dimensions of trust are *processing trust*, *connection trust* and *system trust* [19].

In IoT, data may be collected by potentially untrusted devices. *Trustworthy data* can be obtained by applying different algorithms like data aggregation or machine learning [9].

Entity trust refers to the expected behavior of entities, such as devices, services and users. *Device trust* relates to the interaction with reliable devices [19] and can be established through trusted computing [9].

5) Resilience: Large scale IoT systems are prone to attacks and failures due to the complexity and variety of hardware and software. Therefore, it is important to ensure resilience and robustness against malicious attacks and failures [9].

Intrusion detection and prevention systems provide protection against malicious attacks [15]. Failover and recovery mechanisms ensure resilience and help maintain normal operation [9].

III. Communication Protocols and Networking Stacks

This section provides in-depth investigation of IoT communication protocols and networking stacks and their secu-

Table I
SECURITY LEVELS IN IEEE 802.15.4

Security level	Protection
0	None
1	MIC-32
2	MIC-64
3	MIC-128
4	Encryption
5	Encryption and MIC-32
6	Encryption and MIC-64
7	Encryption and MIC-128

rity capabilities.

A. IEEE 802.15.4

The IEEE 802.15.4 standard [21] was designed as a basis for a protocol stack oriented towards short range, low data-rate and energy efficient communication. It was originally introduced in 2003, with several revisions and additions over the years, and defines the physical (PHY) and Medium Access Control (MAC) layers for short range communications at 250Kbps. The latest version of the standard was released in 2015 and includes previously released amendments that add additional PHY layers and modifications to the MAC layer which better support industrial markets.

While in IEEE 802.15.4 the PHY layer does not offer any security, the MAC layer provides multiple security levels, as described in Table I. All security services as based on the AES-128 block cipher [22] coupled with the CCM* mode of operation [23]. Although the standard specifies security as optional, the effect of having AES-128 in the specification is that most IEEE 802.15.4 compatible hardware platforms (the popular CC2420 [24] or the newer CC2520 [25] transceivers from Texas Instruments or the ATmega128RFA1 and variants SoC from Atmel [26]) implement some form of hardware acceleration for AES-128. This ensures that the energy cost of enabling security on these platforms is minimal. Having almost ubiquitous support for the AES-128 cipher in hardware means that higher layer protocols can also define their security services on top of AES-128 with minimal impact on the energy efficiency of the device. Even if MAC layer security is not employed in a given scenario, the device can still benefit from security at the network or application layers.

In IEEE 802.15.4 communications, secure MAC frames are identified by the *Security Enabled* flag inside the *Frame Control* field. This flag also signals the presence of the *Auxiliary Security Header* which contains information about how the frame is to be secured. The *Security Level* field selects one of the security levels from Table I applied to the frame. A security level of 0 means that the frame is sent unsecured, while security levels from 1 to 3 and 5 to 7 mean that the frames are protected by a Message Integrity Code of the given length, ensuring *integrity* and *origin authentication*. These properties apply to the entire MAC frame, except the *Frame Check Sequence*. Security levels 4 to 7 provide encryption for the payload part of the MAC frame, ensuring its *confidentiality*. All CCM* operations also use a nonce value of 13 bytes formed from the concatenation of the frame's *Source Address* and *Frame Counter* and *Security Level* fields from the *Auxiliary Security Header* which ensures *freshness*, as well as semantic security for the encrypted payload.

The IEEE 802.15.4 specification does not define how to do key management. The AES-128 block cipher uses 128 bit symmetric keys, but the generation, distribution and replacement of those keys is left for the upper layers. The standard does however include a key storing system inside the MAC *PAN Information Base* and a way of implementing a form of access control at the MAC layer, with pair-wise keys or group keys, through the use of the MAC PIB and the *Key Source* field inside the *Auxiliary Security Header*.

B. WiFi

WiFi communications are defined by the 802.11 family of standards, with the first one introduced in 1997 [27]. Popular older standards include 802.11a, 802.11b, and 802.11g. Most devices support the newer standards 802.11n (published in 2009) [28] and 802.11ac (published in 2014) [29].

WiFi networks often operate in congested wireless environments, which might lead to interference and degradation of performance. WiFi communications use frequency bands around 2.4 GHz and 5 GHz; devices operate on frequency ranges centered on preset channels located within those bands. The list of available channels depends on geographical regions. For example, in the 2.4GHz range, 11 channels exist in the US while 13 channels exist in Europe. The channels are 5MHz apart, with channel 1 centered at 2.412GHz.

WiFi standards use different channel widths. For example, 802.11n can use channels with a width of up to 40MHz while 802.11ac mandates the use of channels with a width of 80MHz (and can even reach 160MHz). If two WiFi network channels overlap, interference can lead to lower throughput or even loss of connectivity. Wireless devices often support dynamic selection of channels, but in severely congested industrial environments planning which frequencies are in use can lead to better performance.

WiFi networks support both open communication (i.e., in plain text) and encrypted communication. Due to the multi-access medium, messages are easily intercepted; encryption is preferable despite additional energy costs.

Possible security protocols include Wireless Equivalent Privacy (WEP) and Wi-Fi Protected Access version 1 or 2 (WPA or WPA2). Efficient attacks exist for WEP, and its use is discouraged. Vulnerabilities have also been identified in WPA [30]. WPA2 (introduced in 802.11i-2004) is the current recommendation, and is available on all WiFi certified devices.

Two authentication methods are supported by WPA2: personal, which uses pre-shared keys, and enterprise, which requires an additional RADIUS authentication server and can use multiple underlying authentication mechanisms through Extensible Authentication Protocols (EAP). For pre-shared keys, the overall security of the wireless network relies on choosing a secret key that is hard to guess.

When a client initially connects to an access point, WPA2 uses a secure challenge-based handshake to test whether both devices have the same pre-shared key. The handshake never reveals the pre-shared key. The purpose of the handshake is to derive a temporary secret key for encryption and integrity.

For encryption and integrity, WPA2 supports Counter-Mode CBC-MAC Protocol (CCMP) or Temporal Key Integrity Protocol (TKIP). CCMP uses AES-128 (128-bit keys and 128-bit blocks) in Counter-Mode with CBC-MAC (CCM) mode of operation. Attacks exist against TKIP [30] and it is no longer recommended as of standard 802.11-2012.

Setting WPA2 with CCMP is the recommended method for securing WiFi. Other security measures include: (1) hiding access point network announcements (SSID broadcasts, used to announce network names in an area), (2) allowing or denying station-access point associations based on the stations hardware address, (3) lowering transmit power of base station to reduce communication range. However, they can all be easily bypassed by a determined attacker.

WiFi is a secure choice for IoT communications. It allows for a throughput of up to 1Gbps (for 802.11ac) and security; IEEE proposed 802.11ac for use cases in which HD video streaming is a necessity, such as car cameras, factory floor automation or medical cameras. However, WiFi interfaces use more power compared to other communication technologies; this makes it undesirable for remote sensors with limited battery power.

C. NFC

Near Field Communication (NFC) is a set of short-range communication technologies, operating over electromagnetic fields at a frequency of 13.56MHz over distances of about 10cm. NFC specifications are developed by the NFC Forum, an association composed of companies with interest in NFC. NFC operation is described in standards ISO/IEC 14443 [31], ISO/IEC 18092 [32] and JISX6319-4 [33].

NFC devices communicate by generating electromagnetic fields. In an active communication, both devices generate their own fields. In a passive communication, one device transmits data by modulating the field generated by the active device.

NFC is used to read and write information stored in tags. Five types of tags are currently supported by the NFC forum, with types 1, 2 and 4 described in ISO/IEC 14443, type 3 in JISX6319-4 and type 5 in ISO/IEC 15693 [34].

Older NFC standards did not include any notion of communication security. This made NFC exchanges vulnerable to eavesdropping, data modification, and data insertion [35]. More complex attacks, such as man in the middle, were considered unfeasible due to the physical properties of the generated fields. Relay attacks – where an attacker makes devices that are far apart talk to each other – were also shown to be possible.

Methods of generating secure secrets were proposed [35], but are not standardized.

NFC security comes from the physical proximity. In practical scenarios, the two honest interacting parties need to be within a range of 10cm. Eavesdropping adversaries, for example, need to be closer than a few metres to reliably read exchanged information. The range decreases depending on how the signal is modulated and the type of communication (i.e. passive communications are harder to eavesdrop).

If applications require security, it needs to be implemented at higher levels. For example, SSL can be implemented on top of the low-level NFC protocol to provide secure communication channels.

In 2015, the NFC Forum introduced the Signature Record Type Definition (RTD 2.0) technical specification, which describes how NFC devices can supply authenticated data. This is an update to RTD 1.0, introduced in 2010. The specification lists the algorithms and message formats that NFC devices can use to provide authenticated information.

RTDs use certificates to sign information. A NFC party obtains a signed certificate from a certificate authority, and then uses the associated private key to sign NFC tag data. The result is a Signed NFC Data Exchange Format (NDEF) message, which contains both the signed data and the signature. NFC readers then read the NDEF message, verify the signature using a local list of trusted root certificates and accept the data only if the verification succeeds. This protects against forgery attempts where attackers attempt to supply information to NFC readers on behalf of another party. Currently, RTDs support X.509 and M2M certificates. The supported cryptosystems include RSA, DSA and ECDSA signatures with SHA-256 hashes, providing from 80 up to 128 bits of security.

D. LoRaWAN

Low-Power, Wide-Area Networks (LPWAN) are designed to integrate billions of devices in the Internet of Things [36]. LPWAN technologies complement short range and cellular networks, by providing long battery life (up to 10 years), large communication range and low cost devices [37].

Long Range Wide-Area Network (LoRaWAN) is a LPWAN optimized to have large capacity and range, and low energy consumption and cost. LoRa Alliance is an open, non-profit association of members that collaborate to develop LoRaWAN open standard [38].

LoRaWAN networks have a star-of-stars topology, in which end-devices send messages to gateways, which relay these messages to a central server [38]. End-devices use

single-hop LoRa communication with one of the gateways. The gateways communicate with the server through IP connections.

LoRaWAN standard includes two security layers: one for the network and one for the application. Network-layer security ensures device authentication and Application-layer security ensures the protection of the application data (confidentiality, integrity).

When an end-device is added to the LoRaWAN network, it needs to be personalized and activated [38]. Activation of an end-device can be performed either through Over-The-Air-Activation (OTAA) or through Activation by Personalization (ABP). OTAA is executed when the device is deployed or reset, and ABP includes both personalization and activation. OTAA enables devices to execute a joining procedure before sending any data messages in the network. For this, the end-device needs to be personalized with the following information before the join procedure: a globally unique device identifier (DevEUI), an application idenitifier (AppEUI) and an AES-128 key (AppKey).

AppKey is an AES-128 application key for a certain end-device. This key is allocated by the application owner to the device and is derived from the application-specific root key that is handled by the application provider. AppKey is used to derive the network session key (NwkSKey) and the application session key (AppSKey) when the end-device joins a LoRaWAN network through OTAA.

After the activation, the device stores a device address (DevAddr), an application identifier (AppEUI), a network session key (NwkSKey) and an application session key (AppSKey). NwkSKey is the network session key for a certain end-device. This is used by the end-device and the server to compute and verify the message integrity code (MIC). It is also used for encrypting and decrypting the payload of the MAC-only data messages. NwkSKey is used for encrypting and verifying network communication.

AppSKey is the application session key for a certain end-device. It is used by the end-device and the server to encrypt and decrypt the payload of application-specific data messages. It may also be used for computing and verifying an application-level MIC (which may be included in the payload of application-specific data messages). AppSKey is used for encrypting and verifying application data.

ABT means that DevAddr, NwkSKey and AppSKey are directly written on the end-device instead of DevEUI, AppEUI and AppKey. Each device has its unique set of session keys, so if a device is compromised, it does not affect the secure communication of other devices in the LoRaWAN network.

At the Media Access Control layer, the frame payload (FRMPayload) is encrypted and then a 4-byte MIC is computed [38]. The cryptographic algorithm used for encryption is based on the one used in IEEE 802.15.4 [21], AES with a key of 128 bits. The key used for encryption depends on the value of the Port field (FPort), if FPort is equal to 0, NwkSKey is used, and if FPort is between 1 and 255, AppSKey is used. FPort is equal to 0 when FRMPayload contains only MAC commands, and greater than 0 when it includes application-specific data.

For each data message, considering $payload = FRMPayload$, a number of blocks A_i are generated, where $i = 1..k$ with $k = ceil(len(payload)/16)$. Each block gets encrypted using AES-128: $S_i = \text{AES-128\_encrypt}(K, A_i)$ and all blocks are concatenated $S = S_1|S_2|..|S_k$. Then $(payload|pad_{16})$ xor S is computed and truncated to the number of octets of the initial payload ($len(payload)$). This way LoRaWAN provides *data confidentiality*.

After the frame payload is encrypted, a message integrity code (MIC) is computed over all the fields in the message. The algorithm used for computing the MIC is CMAC based on AES-128, as in Formula 1. The MIC ensures both *data integrity* and *data origin authentication*.

$$\begin{aligned} msg &= MHDR|FHDR|FPort|FRMPayload \\ cmac &= \text{AES-128\_cmac}(NwkSKey, B_0|msg) \\ MIC &= cmac[0..3] \end{aligned} \tag{1}$$

The join procedure consists of two MAC-layer messages exchanged between the joining end-device and the server (join-request and join-accept). The MIC for the join-request message is computed as in Formula 2, where DevNonce is a random value of 2 bytes (this provides *anti-replay protection* for join-request messages). The join-request message is not encrypted.

$$\begin{aligned} cmac = \text{AES-128\_cmac}(&AppKey, MHDR|AppEUI| \\ &DevEUI|DevNonce) \\ MIC &= cmac[0..3] \end{aligned} \tag{2}$$

The join-accept message includes information that is used for deriving the NwkSKey and AppSKey. The MIC for the join-accept message is computed as in Formula 3. The message is encrypted by the server using AppKey as in Formula 4 - the server uses an AES decrypt operation in ECB mode, in order to allow the end-device to use the associated encrypt operation. This is an optimization that allows the end-device to implement only the AES encrypt operation.

$$\begin{aligned} cmac = \text{AES-128\_cmac}(&AppKey, MHDR| \\ &AppNonce|NetID|DevAddr| \\ &RFU|RxDelay|CFList) \\ MIC &= cmac[0..3] \end{aligned} \tag{3}$$

$$\begin{aligned} \text{AES-128\_decrypt}(&AppKey, AppNonce|NetID| \\ &DevAddr|RFU|RxDelay|CFList|MIC) \end{aligned} \tag{4}$$

After the join-accept message is received from the server, the session keys can be generated as in Formula 5.

$$\begin{aligned} NwkSKey = \mathsf{AES\text{-}128\_encrypt}(AppKey, 0\text{x}01| \\ AppNonce|NetID|DevNonce|pad_{16}) \\ AppSKey = \mathsf{AES\text{-}128\_encrypt}(AppKey, 0\text{x}02| \\ AppNonce|NetID|DevNonce|pad_{16}) \end{aligned} \tag{5}$$

E. Z-Wave

Z-Wave is a low-power wireless communication protocol, designed by Sigma Designs, Inc., for remote control applications in residential and small-size commercial environments [39], [40], [41]. The protocol specification and software development kit are not open and are available only to the device manufacturers that signed a contract with Sigma Designs, Inc. [42]. Z-Wave is a complete protocol stack that covers all layers, from physical to application layer.

At the physical layer, Z-Wave operates in the Industrial, Scientific and Medical (ISM) radio frequency band, using low-bandwidth data communication frequencies: 868.42 MHz in Europe and 908.42 MHz in the United States [42]. It adheres to the ITU-T G.9959 PHY and MAC layer specification for sub GHz radio communications [43][44]. This way, it avoids interference with the wireless technologies in the 2.4 GHz range (Wi-Fi, Bluetooth, ZigBee, etc.) [42]. Z-Wave provides a range of 30 meters for point-to-point communications [41] and allows a transmission rate of up to 100 kbps [39].

A Z-Wave mesh network consists in a controller device and up to 232 nodes. Each controller device has a unique 32-bit Home ID, which is the identifier of the Z-Wave network. This ID is written by the manufacturer on the chip and cannot be changed in software. This prevents malicious controller devices from using a spoofed Home ID and collecting information from homes. In addition, controller devices do not support promiscuous mode, so they are not able to intercept all network traffic.

When secure transmission mode is enabled, the frame payload is encrypted and an 8-byte authentication header is added at the end of the frame, before the checksum. The checksum algorithm is described in the ITU-T G.9959 standard.

In the initial setup period (or re-installation), the controller device and the nodes exchange a network key (K_n). This key is generated by the controller using a hardware-based pseudo-random number generator (PRNG) and encrypted using a default key hardcoded in firmware. From this key, all devices derive two 128-bit keys: a data origin authentication key (K_m) and a payload encryption key (K_c). These keys are obtained by encrypting two 16-byte values hardcoded in firmware using AES in ECB operation mode and the network key, as in Equation 6 [42].

$$\begin{aligned} K_m = \mathsf{AES\text{-}ECB}_{Kn}(Passwd_m) \\ K_c = \mathsf{AES\text{-}ECB}_{Kn}(Passwd_c) \end{aligned} \tag{6}$$

Z-Wave computes a Message Authentication Code (MAC) using CBC-MAC with AES, the 8 bytes authentication header, in order to ensure *data origin authentication* and *data integrity*. It also uses 64-bit nonce values (generated using PRNG) when computing the MAC in order to provide *anti-replay protection* (*freshness*). According to Fouladi et al., the MAC is computed using Equation 7 [42].

$$\begin{aligned} \mathsf{MAC} = \mathsf{AES\text{-}CBC\text{-}MAC}_{Km}(IV, \\ SH||SRC||DST||LEN||C) \end{aligned} \tag{7}$$

IV is a 16-byte initialization value composed of 8 bytes generated using PRNG and 8 bytes representing the nonce received from the destination. SH is the security header, which is an one byte value that represents the type of message and is equal to 0x40 for nonce request, 0x80 for nonce reply and 0x81 for encrypted data. SRC and DST are the source and destination device IDs. LEN is the length of the encrypted payload and C is the actual encrypted payload.

The frame payload is encrypted using AES in OFB operation mode, according to Equation 8 [42], in order to provide *data confidentiality*. P is the plain text Z-Wave payload.

$$C = \mathsf{AES\text{-}OFB}_{Kc}(IV, P) \tag{8}$$

F. Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a technology introduced by the Bluetooth Special Interest Group (SIG) in the 4.0 version of the Bluetooth protocol specification. Also known as Bluetooth Smart, it reduces energy consumption and device costs when compared with classic Bluetooth. The Bluetooth specification [45] defines a complete communication stack for BLE composed of the physical layer, the link layer, the Logical Link Control and Adaptation Protocol (L2CAP), which multiplexes the upper layer protocols, the Attribute Protocol (ATT), which defines a way of discovering and transporting attributes (values) and the Generic Attribute Profile (GATT), which defines a framework based on ATT for defining services. The stack is split between the Controller, which implements the physical and link layers, and the Host, which implements the upper layers. These two components communicate with each other using the standardized Host Controller Interface.

The BLE specification defines two mechanisms for security, called *LE security mode 1* and *LE security mode 2*.

In security mode 1, security is applied at the link layer and supports encryption and data signing using the AES-128 [22] block cipher and the CCM [23] mode of operation with different levels of authentication:

- Level 1: no security is applied;
- Level 2: encryption and data signing are applied, but no authentication is performed during key exchange;
- Level 3: security is applied, but the pairing procedure which produces the shared key is performed with a vulnerable algorithm;
- Level 4: pairing no longer is vulnerable due to using Elliptic Curve Diffie-Hellman with the P-256 curve [46] for deriving the shared key.

The Message Authentication Code, called Message Integrity Check (MIC) in the Bluetooth specification, used for data signing has a length of 4 bytes and is computed over the link layer payload and the first byte of the link level header. The 13-byte nonce required by CCM is formed from a 39-bit packet counter, incremented for each packet sent, a 1-bit direction flag and an 8-byte initialization vector exchanged by the communicating devices.

In security mode 2, security is applied on the ATT layer and only supports data signing. The MIC for this security mode is 8-byte long and is computed using CMAC [47] and the AES-128 block cipher, from the concatenation of the transmitted data and a 32-bit counter. The counter is incremented for every message signed in order to protect against replay attacks. Because LE security mode 1 levels 2, 3 and 4 already offer data signing, LE security mode 2 is only applied over unencrypted connections.

BLE defines three shared keys for security purposes, all of them 128-bit long [48]:

- Long Term Key (LTK), used in deriving the key used by the link layer;
- Connection Signature Resolving Key (CSRK), used by the ATT layer for data signing;
- Identity Resolving Key (IRK), which is used to ensure *privacy*, after a connection is established between devices, by generating a new private device address which the other peer can resolve using the shared IRK.

The security of the entire communication depends not only on the security of the ciphers used, for which there are no known practical attacks, but also on the method used for establishing the shared keys [49]. In BLE, the shared keys are established in the pairing step, using 1 of the 4 defined association modes to first establish a *Temporary Key*.

The *Just Works* mode is insecure as the TK has a known fixed value and an attacker can perform a Man-In-The-Middle attack or can passively intercept the pairing procedure and obtain the rest of the shared keys. This mode is used by devices which have no input/output capabilities and cannot ask the user for a confirmation.

The *Out-of-Band* mode is secure if the out-of-band technology used to transfer the TK is secure in itself. Very few devices however are equipped with the necessary hardware to perform an association using this mode.

The remaining modes *Numeric Comparison* and *Passkey Entry* are secure starting with version 4.2 of the Bluetooth specification, which adds the ECDH key exchange algorithm for BLE devices [48]. Before this version, only the *Passkey Entry* method was available and it was using an algorithm different than Diffie-Hellman to establish the TK. With only $1,000,000$ possibilities for the passkey, the method could be easily brute-forced to extract the TK [49] and then the rest of the shared keys.

G. Thread

Thread is an open standard protocol stack that provides low-power, low-cost, wireless IPv6 communication for smart home devices [50]. It has been designed by Thread Group, which is an Internet of Things standards group that includes Google's Nest Labs, Samsung, ARM, Freescale, and others [40]. Thread protocol stack includes: IEEE 802.15.4 PHY and MAC layers, 6LoWPAN, Distance Vector Routing (DVR), UDP, and DTLS.

The Thread standard is based on IEEE 802.15.4 PHY and MAC layers [21], using the 2.4 GHz frequency band and 250 kpbs. The MAC layer provides message confidentiality and integrity protection based on keys that are obtained by the higher layers of the stack. The network layer is build on top of this MAC layer and provides reliable end-to-end communication [50]. Thread ensures *data confidentiality*, *integrity*, and *authentication*.

In Thread the IEEE 802.15.4 MAC layer secures frames using a network-wide key. This provides weak security and is not the only method of securing the messages. However, the network-wide key is used to differentiate between a Joiner device and an authenticated and authorized Thread device. This key will be delivered securely (using a Key Encryption Key) to a Joiner device.

Network authentication and key agreement is based on an elliptic curve variant of J-PAKE (EC-JPAKE): a password-authenticated key exchange with "juggling" using NIST P-256 elliptic curve [51]. This means that it uses elliptic curve Diffie Hellmann for key agreement and Schnorr signatures as Non-Interactive Zero-Knowledge (NIZK) proof mechanism [52] for the authentication between two peers and for establishing a shared key between them based on the passphrase [53]. Thread integrates EC-JPAKE with DTLS in order to provide security.

A Joiner device is initially untrusted and has to be verified using a policing mechanism: the Joiner node has to identify a Joiner Router and communicate only with it. In order to authenticate itself, the device must initiate a DTLS handshake. The Joiner Router will receive all network traffic from the Joiner node and forward it to the Commissioner in a controlled manner, in order to allow the DTLS handshake to be performed [53]. This means that the Joiner Router must relay the authentication traffic between the Joiner node and the Commissioner. This is achieved using a Commissioning relay protocol that provides encapsulation and relaying of the DTLS handshake.

The Commissioning protocol is called MeshCoP and is based on CoAP. It performs petitioning, relay, management and maintenance operations. The Commissioner will use the protocol to maintain a secure communication session alive and to change network parameters.

H. ZigBee

ZigBee is a wireless communication specification [54] defined by the ZigBee Alliance for use in sensor networks applications. It provides a complete protocol stack to foster interoperability between devices from different manufacturers. The ZigBee stack builds on top of the IEEE 802.15.4 standard, which provides the PHY and MAC layers. This makes the ZigBee specification compatible with all 802.15.4 hardware. ZigBee defines a network layer (NWK) which supports star, tree and mesh routing and a framework for building the application layer composed of the application support sub-layer (APS) and the ZigBee device objects (ZDO), which the application uses to build its own application objects.

As for the communication stack, the security of ZigBee is built on top of the security services provided by the IEEE 802.15.4 standard [55]. Although ZigBee doesn't directly use the MAC layer security defined in 802.15.4 [56], it uses the same AES-128 block cipher and CCM* mode of operation to secure transmissions at both the NWK and APS layers. The principle that "*the layer that originates a frame is responsible for initially securing it*" is employed, which means that if a NWK command frame needs protection, the protection will be applied at the NWK layer. Because the specification is targeted at low-cost devices, no security separation is assumed between stack layers. The consequence of this is that the same key can be shared by multiple layers, decreasing complexity and storage costs associated with security keys. The ZigBee specification also defines a set of security levels that mirror the security levels of IEEE 802.15.4 (Table I) and, depending and the level, provide the same protections (i.e *integrity*, *origin authentication*, *confidentiality* and/or *freshness*) as described in section III-A. Unlike the 802.15.4 standard, in ZigBee the security level is common to the whole network and cannot be changed on a per frame basis.

Unlike IEEE 802.15.4, ZigBee defines how key management is handled, making the specification complete and ready for deployment. For securing ordinary data communication ZigBee defines two types of keys: the *network key*, which is shared by all devices in a ZigBee network and offers protection against adversaries outside the network and multiple *link keys*, which are shared only between pairs of devices and offer end-to-end security between two devices. For specialized purposes there exists also a *master key*, used in establishing link keys via the Symmetric-Key Key Establishment (SKKE) protocol and *key-transport* and *key-load* keys, used in securing messages containing other keys. The key-transport and key-load keys are derived from the link key using HMAC [57] and the Matyas-Meyer-Oseas hash function instantiated with the AES-128 block cipher.

All keys used by ZigBee are 128 bits in length and the security of the specification rests on the secure initialization and installation of these keys. There are two mechanisms for obtaining a required key: the first is to have the key preinstalled on the device and the second is to obtain the key from a special node, the *Trust Center*, which is trusted by all nodes in the network. The *Trust Center* is also responsible with network management, accepting or rejecting nodes into the network.

For nodes joining the network that have no preinstalled keys, there is no way for the *Trust Center* to securely transport the required keys to the node in the join procedure. In this case the ZigBee specification allows for the key to be transported unsecured on the last hop to the joining node. For applications that allow such unsecured transportation of keys there is a small window of vulnerability when network security can be compromised by a passive attacker. The specification assumes that security is ensured through non-cryptographic means in this case, and recommends transmitting the key after external input to the nodes, only once, at low power to evade passive attacks.

I. 6LoWPAN

The IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [58] is a network layer protocol standardized by IETF for enabling Internet connectivity on devices with constrained resources. The initial 6LoWPAN specification describes the frame format and different header compression schemes for IPv6 packets when transmitted over IEEE 802.15.4 networks. Other specifications have been standardized for supporting IPv6 over Bluetooth Low Energy connections [59] or over ITU-T G.9959 networks [60] (used in the Z-Wave stack) and more documents are in the process of being standardized for using IPv6 over NFC [61], over DECT Ultra Low Energy [62] or over MS/TP [63] (a component in the building automation stack BACnet). Using 6LoWPAN resource constrained devices can achieve end-to-end connectivity with any other IPv6 enabled host.

The 6LoWPAN specification does not define any security mechanisms. It relies on securing communications with the mechanism available at the link-layer (e.g. IEEE 802.15.4 security - described in section III-A) or at the upper layer (e.g. CoAP - described in section III-L). An academic proposal exists for extending 6LoWPAN with IPSec using header compression techniques [64] [65] and comparisons with link-layer security [66] show that it scales better as data size and number of hops grow.

J. RPL

The IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [67] is a network layer protocol standardized

by IETF, complementary to 6LoWPAN (section III-I), that describes the way by which routing is done inside Low-Power and Lossy Networks (LLNs). The protocol defines the RPL messages exchanged between LLN nodes over ICMPv6 and how they are used to determine routing tables inside the LLN. Point-to-point traffic inside the LLN is supported as well as point-to-multipoint with the central node of the LLN.

The RPL specification provides a number of security mechanisms to protect against attacks on routing information. Three security modes are defined by the specification: unsecured, preinstalled and authenticated. A security bit inside the RPL packet distinguishes between secure and unsecure versions of the RPL control messages.

The unsecure version of the RPL control messages is designed to lower overhead when security is already provided by the link-layer (as possible with IEEE 802.15.4 - section III-I). If security is not provided at any layer, a number of attacks are possible against RPL [68].

The secure versions of the RPL control messages add a *Security Section* to the RPL header which identifies what type of security was used to protect a given packet. All combinations of security settings provide at least *data integrity* and *replay protection*, with optional *confidentiality* and *delay protection*.

In preinstalled mode, preinstalled keys are used by nodes to join an RPL network either as a host, or as a router. Authenticated mode is only partially defined, for the purpose of enabling future extensions, as it is not supported by the symmetric encryption algorithms allowed by the specification.

In authenticated mode a node can use preinstalled keys to join the network as a host, but then it has to request a second key from a key authority if it wants to become a router. This key authority can perform authentication of the node, however the details of how the authority is contacted and how authentication is performed are not specified.

The *Security Section* of a secure RPL message supports different security levels identified by a combination of the *Key Identifier Mode (KIM)* and *Security Level (LVL)* fields. The KIM identifies what type of key is used for the security algorithm. Symmetric group or per-pair keys are supported, as well as asymmetric keys. A packet can have either a MAC of 32 or 64 bits in length, computed using the AES-128 [22] block cipher with the CCM [23] mode of operation or a signature of 256 or 384 bits in length, computed using the RSA public-key cryptosystem [69] with 2048 or 3072 bit keys, respectively.

The signature calculation uses the RSASSA-PSS instantiation of the RSA algorithm with the SHA-256 secure hash function [70]. For each type of MAC or signature calculation encryption is also supported with AES-128 and the CCM mode of operation. In the cases where signatures are used, the CCM mode of operation is used in a way that only the ciphertext is computed without an associated MAC. In this case data integrity and replay protection are insured by the packet signature.

The MAC or signature for a message are computed over the entire packet content, including the IPv6 and ICMPv6 headers, while encryption excludes the headers and the Security Section of the RPL message. The RPL specification also includes the possibility of upgrading or replacing the encryption/authentication algorithm using the *Algorithm* field of the *Security Section*.

To protect against replay attacks and to provide semantic security, RPL nodes maintain counters that are used in the CCM nonce and that are sent in each secured packet. Optionally, the counters can represent the value of a timestamp, if it has a length of at least 6 bytes and a granularity of at least 1024Hz. This optional support for timestamp counters is intended to provide a simplified protection for delay attacks, inside LLNs that already provide a global synchronized time using other means. To mitigate attacks against the anti-replay protection the RPL specification also includes a *Consistency Check (CC)* message that can be sent by RPL nodes to check or synchronize their counters. The CC messages include a randomly generated nonce so they too cannot be replayed by an attacker.

K. IPSec

IPsec is a protocol suite for establishing secure channels over untrusted networks. The various parts of IPsec are defined in a large number of RFCs, with the main features of the suite described in RFC 4301 [71]. IPsec is available in all IPv6 networks, and is also compatible with IPv4 networks.

IPsec provides *integrity*, *confidentiality* and *authentication* at the Internet layer of the TCP/IP protocol stack. Compared to upper layer security protocols (e.g., TLS), IPsec is able to protect IP and transport layer header fields (e.g., IP addresses, port numbers).

Algorithms are negotiated during secure channel initialization depending on security and performance policies. IPsec channel setup typically consists of two phases. The first phase negotiates the security parameters and performs peer authentication; this step is handled by the IKEv2 protocol [72]. The first phase ends with a secure channel between the participating peers. The second phase uses this secure channel to negotiate the security parameters and shared secrets for actual data traffic. Once the negotiatiation ends successfully, data can be exchanged. The parameters are valid for a certain amount of time, after which a rekeying exchange runs to establish new secret.

Securing data with IPsec is more expensive than relying only on link layer security; packet size increases to account for the additional headers and the cryptographical algorithms might run on general purpose hardware, thus using a significant amount of processing time. However, as opposed to a strict link layer security implementation, IPsec traffic

can leave the local network and securely pass through the Internet.

The impact on processing power of running IPsec has been measured by Granjal et al. [73]. Optimizations on communication size have also been suggested [64]. Some mitigation for packet size could include disabling Data Link security protocols, since IPsec will achieve end-to-end security at the Internet layer of the protocol stack [64].

Additionally, depending on needs, IPsec can disable some security requirements; for example, if some application only needs integrity, IPsec can be configured to no longer encrypt packets.

IPsec protocols can operate in either *transport mode* or *tunnel mode*. In *transport mode* the initial IP header is left intact, with the notable exception of changing the Protocol field. The IPsec header is sandwiched between the IP header and payload (with the payload defined as transport and application layer data). Only some IP headers fields are protected, depending on whether they need to be modified during network transit. For example, IP addresses are protected while the *Time To Live* (TTL) field is not.

In *tunnel mode*, the IPsec header encapsulates the entire IP packet. All IP header fields from the encapsulated packet are protected. A new IP packet with new IP addresses then encapsulates the IPsec packet. Outer IP packet layer fields are also protected, similarly to *transport mode*. The encapsulation *tunnel mode* performs is useful in creating Virtual Private Networks over untrusted infrastructure.

IPsec uses two network protocols to communicate over the network: Authentication Header (AH) and Encapsulating Security Payload (ESP).

Authentication Header only provides *authentication*, *integrity* and *replay protection*. The lack of encryption decreases power consumption. Encapsulating Security Payload provides *authentication*, *confidentiality*, *integrity* and *replay protection*. Confidentiality protection can be disabled, making Encapsulating Security Payload perform similarly to Authentication Header.

IPsec implementations must support HMAC-SHA1-96, AES-GMAC with AES-128, and AES-XCBC-MAC-96 for authentication algorithms, AES-CBC and NULL for encryption algorithms, and AES-GCM for combined encryption and authentication [74]. IPsec implementations might support additional algorithms such as TripleDES-CBC, AES-CTR, etc. Unsafe algorithms must not be supported (e.g., DES-CBC).

L. CoAP

Constrained Application Protocol (CoAP) [75] [76] is an application layer protocol for IoT, created by an IETF working group called Constrained RESTful Environments (CoRE) [77] [41]. CoAP is practically a web transfer protocol based on REpresentational State Transfer (REST), that is more lightweight than HTTP and can be used on resource-constrained devices [8].

CoAP does not provide security in itself, but it uses Datagram Transport Layer Security (DTLS) [78] at the transport layer, in order to secure all CoAP messages. DTLS provides *data confidentiality* and *integrity*, *authentication*, *non-repudiation* and *anti-replay protection* for CoAP communication [79]. It is an adaptation of TLS that works over UDP (it has additional features that deal with the unreliable nature of UDP) and offers end-to-end security [80]. CoAP with DTLS support is also called *secure CoAP (CoAPs)* [81].

During the provisioning phase, the CoAP device receives all necessary security information, for example keys and access control lists [76]. After the provisioning phase the device will be in one of the four security modes, that are described further on. In the provisioning phase, the device identifiers are collected and stored on the data collection server. The list of identifiers can be used for performing access control: for example a device receives an access control list of device identifiers with which it can initiate DTLS sessions.

CoAP defines four security modes: *NoSec*, *PreSharedKey*, *RawPublicKey* and *Certificates* [76] [79] [82]. The specification states that at least two modes should be implemented, *NoSec* and *RawPublicKey*.

In the *NoSec* mode, CoAP messages are not secured using DTLS (DTLS is disabled). In this mode, packets are sent directly using UDP over IP.

In the *PreSharedKey* mode, DTLS is enabled and the device is pre-programmed with symmetric shared keys. A device may have a list of shared keys and each key can be used for the communication with a single node, or a group of nodes. In this mode, the system opens a DTLS session in a PSK mode with the destination node. The specification [76] states that at least *TLS_PSK_WITH_AES_128_CCM_8* cipher suite should be supported.

In the *RawPublicKey* mode, DTLS is enabled and the device is pre-programmed with an asymmetric key pair (that can be validated using an out-of-bound mechanism). The device has an identity computed from its public key and a list of node identities that it can communicate with. The specification [76] states that at least *TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8* cipher suite should be implemented for this security mode. The public keys should be ECDSA-capable, the curve secp256r1 must be supported and the hashing algoritm is SHA-256.

In the *Certificates* mode, DTLS is enabled and the device has an asymmetric key pair and an X.509 certificate that binds it to its subject and is signed by a trust root. The device must have a list of trust anchors for validating the certificates. The specification [76] states that at least *TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8* cipher suite should be supported for this security mode. The public key must be ECDSA-capable, the curve secp256r1

must be supported and the hashing algoritm is SHA-256. The certificate includes a field *SubjectPublicKeyInfo* that specifies the public key that should be used in the ECC calculations. It also includes a signature generated using ECDSA (secp256r1) and SHA-256. If the device has a shared key besides the certificate, it must also support *TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA*. It is expected that other cipher suites will be included in CoAP in the future versions.

An advantage of CoAP with DTLS support is that it supports Elliptic Curve cryptography (ECC) for the last two security modes. Device authentication is performed using ECDSA (Elliptic Curve Digital Signature Algorithm) and key agreement is achieved using ECDHE (Elliptic Curve Diffie-Hellman Algorithm).

CoAP does not include any key distribution and management schemes. It is assumed that keys are obtained from the initial DTLS authentication handshake. Also, DTLS does not support multiccast communications [79].

Raza et al. proposed Lithe, that uses a compressed version of DTLS for securing CoAP communication [81]. DTLS is compressed using 6LoWPAN compression mechanisms. This optimization has the purpose of improving energy efficiency on constrained devices and to avoid 6LoWPAN fragmentation, in order to increase CoAPs usage in constrained environments [83].

M. MQTT

Message Queue Telemetry Transport (MQTT) is a messaging protocol that was first presented in 1999 and was standardized in 2013. MQTT is used in IoT systems in fields such as health, energy, and social media such as Facebook [41].

MQTT is used to communicate between small devices having low capabilities over unreliable, low bandwidth networks. The protocol connects embedded devices and applications using a routing mechanism which works one-to-one, one-to-many and many-to-many [41].

MQTT is based on the publish subscribe model that uses a broker to route the messages to the clients. The three actors of the communication process are the publisher, the subscriber and the broker. In order to send a message, the publisher publishes the message having a certain topic name. Then, the broker places the message in the corresponding queue. Once it is placed there, all the subscribers to that topic automatically receive the message as a push notification. The clients can subscribe to multiple topics and there is also the possibility to use hierarchical topics (eg. "sensor/Room2/light") [84].

Each MQTT message contains a 2-byte header containing information such as the Message Type (CONNECT, CONNACK, PUBLISH, SUBSCRIBE), a DUP flag specifying whether the message is duplicated or not, and a Quality of Service level. There is also Retain field which tells the server to keep the message and retransmit it to new subscribers and the Remaining Length field storing the length of the rest of the message [41].

MQTT is built on top of TCP which takes care of ensuring message delivery and retransmits the packets if that is the case. On top of this, the messages being trasmitted are associated a Quality of Service level (QoS). There are three QoS levels available. For level 0, the sender sends the message only once without checking if it reached its destination. Level 0 relies solely on the TCP protocol, thus there is the possibility of the message not being delivered. For the next two levels, the protocol ensures the message reaches the destination. Messages having QoS level one are retransmitted until all the subscribers confirmed receiving them. This may result in some of the clients receiving the same message more than once. On the other hand, level 2 ensures that the same message reaches its subscribers only once [84].

The MQTT protocol requires each client to establish a connection with the broker before being able to publish messages or subscribe to topics. For this, a special message containing the Client ID is sent. This way, the broker can keep track of the messages received by each client, thus making possible the implementation of QoS level 2 [84] [85].

The choice regarding which QoS level to be used depends on the one implementing the protocol. The higher the level, the more bandwidth and energy are required. For instance, one could use level 0 for sending sensor data such as the humidity level, however, level 1 or 2 should be used for sending crucial messages such as alerts.

From the security point of view, the MQTT message can have a Variable Header, longer that the 2-byte header, which also contains a username and a password for authentication support. This way, the broker can deny connections. However, the values are not encrypted, making the communication insecure. There is SMQTT, an extension of MQTT which addresses the security issues, but it is still under research [85]. MQTT over TLS can be used for ensuring communication security (port 8883 is standardized for secure MQTT).

N. XMPP

eXtensible Messaging and Presence Protocol (XMPP, also known as Jabber) is a popular IETF protocols for instant messaging [86]. It is built over TCP and uses XML as data model. Some compatible implementations include Facebook Messenger, Google Hangouts, Jabber, etc.

The protocol is based on stanza exchanges. A stanza is a part of an XML document that is sent from one peer to another or to several others. XMPP relies on a set of standard stanzas and several extensions. All XMPP servers are required to know the standard set and may implement

several extensions. The most used ones are File Sending and Audio and Video Conference [41].

When it comes to security, XMPP implements authentication, authorization and secure communication.

XMPP provides secure communication using either a SSL connection or START-TLS. START-TLS initiates a normal TCP connection and switches to a secure one after a simple handshake. In this way, systems that have no TLS capability – such as constrained IoT devices – may still connect securely. Even if information is sent in plain text, the authentication method may still use encryption [87].

Authentication is done using a special SASL profile. This implies verifying a username, a fully qualified domain name, an identifier and a passphrase. Sending these can be done by choosing one of the following methods: PLAIN, DIGEST-MD5 [87].

The XMPP standard specifies that an XMPP server will have to authenticate users and servers. Contrary to other protocols, XMPP makes sure that when a server connects it provides a valid and verified Fully Qualified Domain Name (FQDN). Moreover, servers connect to each other directly so that information is sent directly to the recipient. The protocol specifies the possibility of having a multi-hop path, still servers on the path need to authenticate to each other.

As XMPP uses XML as its main data format, this generates significant overhead. This downside is overcome by the fact that it is one of the most secure protocols available for IoT device communication. The recommendation is to use a simple and lightweight protocol like MQTT for the internal network, where sensors communicate with a gateway, and XMPP when relaying information over a public network.

O. AMQP

Advanced Message Queuing Protocol (AMQP) is an open messaging protocol. In 2006, major companies such as JPMorgan Chase, Cisco Systems and RedHat joined forces into AMQP working groups, whose aim is to create an open, asynchronous messaging protocol to work at enterprise scale [88].

AMQP both offers a network protocol and also comprises a protocol model [88]. The network protocol specifies the entities taking part in the communication and the protocol model specifies how the messages should look like and which commands to use in order to make the implementation interoperable with others [89].

AMQP transmits messages by using the publish subscribe communication model or with point-to-point communication. The message passing is done via queues and require exchanges to route the messages to the corresponding queues. The routing is done based on rules that the clients subscribe to. AMQP allows customizations in what messages should be received, which the senders should be and even how to adapt the message exchange in order to keep the system secure and reliable [41].

The protocol is built on top of a TCP connection. In addition, similarly to MQTT, AMQP also supports primitives assuring at-most-once, at-least-once and exactly-once message delivery. On top of the transport layer, the protocol defines another layer called messaging. For this, we have two types of messages: bare and annotated. The bare messages are the ones fired by the sender and the annotated are the ones reaching the destination. The annotated message is built starting from the bare one to which extra information is added [41].

The bare message contains the body of the message together with System Properties, defined by the AMQP protocol and Application Properties, defined by each application implementing the protocol. The System Properties comprise properties such as *Message Id*, *To*, *Subject*, *Reply to*. Among the extra information specified by the messaging layer is the state of the message.

AMQP has TLS over TCP support. Thus, it is difficult to implement on IoT devices with limited resources. It provides authentication and secure communications through SASL and TLS.

IV. Discussion

IoT systems include technologies designed especially for low-power devices, such as IEEE 802.15.4, LoRaWAN, Z-Wave, etc. Table II presents a comparative evaluation of the IoT protocols and networking stacks investigated in this paper with regard to communication security requirements.

Some of them cover the physical and media access layers (IEEE 802.15.4, Wi-Fi, NFC and LoRaWAN), others cover the entire networking stack (BLE, Z-Wave, Thread). ZigBee is built on top of IEEE 802.15.4, covering network to application layers. IPSec, 6LoWPAN and RPL are network layer protocols, while CoAP, while XMPP, AMQP and MQTT are application layer protocols.

Thread includes a collection of standardized protocols in its networking stack: IEEE 802.15.4, 6LoWPAN, DVR, UDP and DTLS [50]. IETF proposes the network stack that includes: IEEE 802.15.4, 6LoWPAN, RPL, UDP and CoAP [79].

Some IoT technologies, like 802.15.4 and LoRaWAN, are more efficient and can be used on resource-constrained devices. Wi-Fi on the other hand consumes much more resources and is not adequate for low-power devices.

Some application-layer protocols, such as CoAP and MQTT, are more appropriate for running on constrained devices. Others, like XMPP, are recommended for the communication between gateways and servers, over the Internet.

Except for 6LoWPAN and NFC, all presented solutions provide communication security: data confidentiality, integrity, origin authentication and freshness. NFC provides only data integrity and origin authentication. CoAP is integrated with DTLS, MQTT, XMPP and AMQP can be integrated with TLS in order to ensure secure communication.

Protocol/stack	Layers	Data Confidentiality	Data Integrity	Data Origin Authentication	Data Freshness
IEEE 802.15.4	PHY - MAC	●	●	●	●
Wi-Fi	PHY - MAC	●	●	●	●
NFC	PHY - MAC	-	●	●	-
LoRaWAN	PHY - MAC	●	●	●	o
BLE	PHY - APP	●	●	●	●
Z-Wave	PHY - APP	●	●	●	●
Thread	PHY - APP	●	●	●	●
ZigBee	NET - APP	●	●	●	●
IPSec	NET	●	●	●	●
6LoWPAN	NET	-	-	-	-
RPL	NET	●	●	●	●
CoAP+DTLS	TR - APP	●	●	●	●
MQTT+TLS	TR - APP	●	●	●	●
XMPP+TLS	TR - APP	●	●	●	●
AMQP+TLS	TR - APP	●	●	●	●

Table II
COMPARATIVE EVALUATION OF IOT PROTOCOLS (●= YES, O = PARTIAL, - = NO)

V. CONCLUSIONS

As the Internet of Things continues to expand, the diversity and complexity of IoT applications increases. Such networks are vulnerable to attacks that aim to steal sensitive information, take control over devices and disrupt services.

Many protocols and networking stacks for IoT have been developed. Some of them are standardized, and provide interoperability between devices and connectivity over the Internet. They have been specified by standardization bodies such as IETF and IEEE or by industry alliances, such as LoRaWAN Alliance and Thread Group.

This paper analyzed the security requirements specific to IoT systems, by taking into consideration network security, identity management, privacy, trust, and resilience. Next, the standardized protocols and networking stacks for IoT, and the mechanisms they provide for satisfying communication security requirements are investigated. We presented the mechanisms that ensure data confidentiality, integrity, origin authentication and freshness for each IoT technology.

A large selection of IoT technologies was analyzed, from single-layer protocols (such as 6LoWPAN) to full protocol stacks (such as Thread). Their functionality and security capabilities are presented, and table II summarizes the protocol layers and security requirements that are covered by the investigated technologies.

As a future work, we would like to investigate standardized solutions for IoT that meet other security requirements, such as trust and resilience. Another interesting area of research would be how the security properties of the various specifications transfer to practical implementations, given the limitations of IoT devices and the possible variations inherent in a complete stack.

ACKNOWLEDGMENT

This work has been funded by program Partnerships in priority areas PN II carried out by MEN-UEFISCDI, project No. 47/2014, and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398.

REFERENCES

[1] M. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A Critical Analysis on the Security Concerns of Internet of Things (IoT)," *International Journal of Computer Applications*, vol. 111, no. 7, pp. 1–6, 2015.

[2] C. Kolias, A. Stavrou, and J. Voas, "Securely making 'things' right," *Computer*, vol. 48, no. 9, pp. 84–88, 2015.

[3] V. Bhuvaneswari and R. Porkodi, "The internet of things (IOT) applications and communication enabling technology standards: An overview," in *International Conference on Intelligent Computing Applications, ICICA 2014*, 2014, pp. 324–329.

[4] S. M. Sajjad and M. Yousaf, "Security analysis of IEEE 802.15.4 MAC in the context of Internet of Things (IoT)," in *2014 Conference on Information Assurance and Cyber Security (CIACS)*, 2014, pp. 9–14. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6861324

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.01.010

[6] K. Ashton, "That 'internet of things' thing," 2009. [Online]. Available: http://www.rfidjournal.com/articles/view?4986

[7] "Gartner says 4.9 billion connected "things" will be in use in 2015," 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2905717

[8] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman, and P. Demeester, *IETF Standardization in the Field of the Internet of Things (IoT): A Survey*, 2013, vol. 2, no. 2. [Online]. Available: http://www.mdpi.com/2224-2708/2/2/235/htm

[9] E. Vasilomanolakis, J. Daubert, and M. Luthra, "On the Security and Privacy of Internet of Things Architectures and Systems," in *Secure Internet of Things (SIoT 2015)*, 2015, pp. 49–57. [Online]. Available: https://www.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_TK/filesDownload/Published_Papers/SIoTpaper.pdf

[10] "50 sensor applications for a smarter world." [Online]. Available: http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/

[11] M. Abomhara and G. M. Koien, "Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks," *Journal of Cyber Security and Mobility*, vol. 4, no. 1, pp. 65–88, 2015. [Online]. Available: http://www.riverpublishers.com/journal_read_html_article.php?j=JCSM/4/1/4

[12] "Real-world security and the internet of things," 2016. [Online]. Available: https://www.schneier.com/blog/archives/2016/07/real-world_secu.html

[13] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of Things (IoT) Security : Current Status , Challenges and Prospective Measures," in *10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 336–341.

[14] Z. Z.-K., C. M.C.Y., and S. S., "Emerging security threats and countermeasures in IoT," in *ASIACCS 2015 - Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, 2015, pp. 1–6. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84942523308{&}partnerID=40{&}md5=3c4a207d8590001416184468e21e3d27

[15] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 2, pp. 2–23, 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4109893

[16] A. Barki, A. Bouabdallah, S. Gharout, and J. Traore, "M2M Security: Challenges and Solutions," *IEEE Communications Surveys & Tutorials*, no. c, pp. 1–1, 2016. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7373528

[17] L. Miller, *IoT Security for Dummies*. John Wiley & Sons, Ltd, 2016.

[18] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2014.11.008

[19] J. Daubert, A. Wiesmaier, and P. Kikiras, "A view on privacy & trust in IoT," in *2015 IEEE International Conference on Communication Workshop, ICCW 2015*, 2015, pp. 2665–2670.

[20] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management," Tech. Rep., 2009. [Online]. Available: http://dud.inf.tu-dresden.de/Anon_Terminology.shtml$\delimiter"026E30F$nhttp://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[21] *Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Std. 802.15.4, 2011. [Online]. Available: http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf

[22] *Advanced Encryption Sstandard (AES)*, NIST Std. FIPS-197, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[23] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, NIST Std. Special Publication 800-38C, 2004. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

[24] "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver (CC2420)," Texas Instruments, Dallas, USA. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2420.pdf

[25] "2.4 GHz IEEE 802.15.4 / ZigBee RF Transceiver (CC2520)," Texas Instruments, Dallas, USA. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2530.pdf

[26] "8-bit AVR Microcontroller with Low Power 2.4GHz Transceiver for ZigBee and IEEE 802.15.4 - ATmega128RFA1," Atmel, San Jose, USA. [Online]. Available: http://www.atmel.com/Images/Atmel-8266-MCU_Wireless-ATmega128RFA1_Datasheet.pdf

[27] IEEE, "802.11-1997 standard," Tech. Rep. [Online]. Available: http://standards.ieee.org/getieee802/download/802.11n-2009.pdf

[28] ——, "802.11n-2009 standard," Tech. Rep. [Online]. Available: http://standards.ieee.org/getieee802/download/802.11ac-2013.pdf

[29] ——, "802.11ac-2013 standard," Tech. Rep. [Online]. Available: https://standards.ieee.org/findstds/standard/802.11-1997.html

[30] E. Tews and M. Beck, "Practical attacks against wep and wpa," in *Proceedings of the second ACM conference on Wireless network security*. ACM, 2009, pp. 79–86.

[31] ISO, "ISO/IEC 14443," Tech. Rep. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=70171

[32] ——, "ISO/IEC 18092," Tech. Rep. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=56692

[33] Sony, "Felica," Tech. Rep. [Online]. Available: http://www.sony.net/Products/felica/business/tech-support/data/M830_NFC_FeliCa_1.1e.pdf

[34] ISO, "ISO/IEC 15693," Tech. Rep. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=39694

[35] K. Breitfuß and E. Haselsteiner, "Security in near field communication," in *Workshop on RFID security*, 2006.

[36] "LoRaWAN. What is it? A technical overview of LoRa and LoRaWAN," LoRa Alliance, Tech. Rep. November, 2015.

[37] "LPWA Technologies. Unlock New IoT Market Potential." LoRa Alliance, Tech. Rep. November, 2015.

[38] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, "LoRaWAN Specification," LoRa Alliance, Inc., Tech. Rep., 2015. [Online]. Available: https://www.lora-alliance.org/portals/0/specs/LoRaWANSpecification1R0.pdf

[39] "About z-wave technology." [Online]. Available: http://z-wavealliance.org/about_z-wave_technology/

[40] "Making Sense of IoT Standards," Redbend, Tech. Rep. March, 2015.

[41] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[42] B. Fouladi and S. Ghanoun, "Security Evaluation of the Z-Wave Wireless Protocol," in *Black Hat Conference*, 2013, p. 6.

[43] *G.9959 : Short range narrow-band digital radiocommunication transceivers - PHY and MAC layer specifications*, ITU-T Std. G.9959 (02/12), 2012. [Online]. Available: https://www.itu.int/rec/T-REC-G.9959

[44] J. D. Fuller, B. W. Ramsey, J. Pecarina, and M. Rice, "Wireless intrusion detection of covert channel attacks in ITU-T G.9959-based networks," in *Proceedings of the 11th International Conference on Cyber Warfare and Security, ICCWS 2016*, 2016, pp. 137–145.

[45] *Specification of the Bluetooth System*, Bluetooth SIG Core Specification, Rev. 4.2, Dec. 2014. [Online]. Available: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439

[46] *Recommended Elliptic Curves for Federal Goverment Use*, NIST Std., 1999. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

[47] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, NIST Std. Special Publication 800-38B, 2005. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38b.pdf

[48] 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. 11 734–11 753, 2012.

[49] M. Ryan, "Bluetooth: With low energy comes low security," Presented as part of the 7th USENIX Workshop on Offensive Technologies. USENIX, 2013. [Online]. Available: https://www.usenix.org/conference/woot13/workshop-program/presentation/Ryan

[50] "Thread Stack Fundamentals," Thread Group, Tech. Rep., 2015.

[51] F. Hao, "J-PAKE: Password Authenticated Key Exchange by Juggling," Internet Engineering Task Force, Internet-Draft draft-hao-jpake-04, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-hao-jpake-04

[52] ——, "Schnorr NIZK Proof: Non-interactive Zero Knowledge Proof for Discrete Logarithm," Internet Engineering Task Force, Internet-Draft draft-hao-schnorr-04, Jul. 2016, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-hao-schnorr-04

[53] "Thread Commissioning," Thread Group, Tech. Rep., 2015.

[54] *ZigBee Specification*, ZigBee Alliance Std. Document 053 474r17, 2008.

[55] G. Dini and M. Tiloca, "Considerations on security in zigbee networks," in *International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*. IEEE, Jun. 2010, pp. 58–65.

[56] K. Masica, "Recommended practices guide for securing zigbee wireless networks in process control system environments," Lawrence Livermore National Laboratory, Tech. Rep., 2007. [Online]. Available: http://cms.doe.gov/sites/prod/files/oeprod/DocumentsandMedia/Securing_ZigBee_Wireless_Networks.pdf

[57] *The Keyed-Hash Message Authentication Code (HMAC)*, NIST Std. FIPS-198-1, 2008. [Online]. Available: http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

[58] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, Internet Engineering Task Force (IETF) Std. RFC 4944, Sep. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4944

[59] Z. Shelby, J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, and C. Gomez, *IPv6 over BLUETOOTH(R) Low Energy*, Internet Engineering Task Force (IETF) Std. RFC 7668, Oct. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7668

[60] A. Brandt and J. Buron, *Transmission of IPv6 Packets over ITU-T G.9959 Networks*, Internet Engineering Task Force (IETF) Std. RFC 7428, Feb. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7428

[61] J.-S. Youn and Y.-G. Hong, "Transmission of IPv6 packets over Near Field Communication," Internet Engineering Task Force (IETF), Internet-Draft draft-ietf-6lo-nfc-04, Jul. 2016, Work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6lo-nfc-04

[62] P. B. Mariager, J. T. Petersen, M. van de Logt, D. Barthel, and Z. Shelby, "Transmission of IPv6 packets over DECT Ultra Low Energy," Internet Engineering Task Force (IETF), Internet-Draft draft-ietf-6lo-dect-ule-05, Mar. 2016, Work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6lo-dect-ule-05

[63] K. Lynn, J. Martocci, C. Neilson, and S. Donaldson, "Transmission of IPv6 over MS/TP networks," Internet Engineering Task Force (IETF), Internet-Draft draft-ietf-6lo-6lobac-05, Jun. 2016, Work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6lo-6lobac-05

[64] S. Raza, T. Chung, S. Duquennoy, T. Voigt, U. Roedig *et al.*, "Securing internet of things with lightweight ipsec," 2010.

[65] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing communication in 6LoWPAN with compressed IPsec," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*. IEEE, 2011, pp. 1–8.

[66] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, "Secure communication for the Internet of Things - A comparison of link-layer security and IPsec for 6LoWPAN," *Security and Communication Networks*, vol. 7, no. 12, pp. 2654–2668, 2014.

[67] A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, T. H. Clausen, and T. Winter, *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, Internet Engineering Task Force (IETF) Std. RFC 6550, Mar. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6550

[68] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LoWPAN in IoT," in *International Conference on Pervasive Computing (ICPC)*. IEEE, 2015, pp. 1–6.

[69] B. S. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, Internet Engineering Task Force (IETF) Std. RFC 3447, Feb. 2003. [Online]. Available: https://tools.ietf.org/html/rfc3447

[70] *Secure Hash Standard (SHS)*, NIST Std. FIPS-180-4, Aug. 2015. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[71] S. Kent and K. Seo, "Security architecture for the internet protocol," Tech. Rep. [Online]. Available: https://tools.ietf.org/html/rfc4301

[72] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet key exchange protocol version 2 (ikev2)," Tech. Rep. [Online]. Available: https://tools.ietf.org/html/rfc5996

[73] J. Granjal, R. Silva, E. Monteiro, J. S. Silva, and F. Boavida, "Why is ipsec a viable option for wireless sensor networks," in *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, 2008, pp. 802–807.

[74] D. McGrew and P. Hoffman, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)," Tech. Rep. [Online]. Available: https://tools.ietf.org/html/rfc7321

[75] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[76] C. Bormann, K. Hartke, and Z. Shelby, "The Constrained Application Protocol (CoAP)," RFC 7252, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[77] "Constrained restful environments (core)." [Online]. Available: https://datatracker.ietf.org/wg/core/charter/

[78] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," RFC 4347, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc4347.txt

[79] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things : A Survey of Existing Protocols and Open Research Issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[80] S. Raza, "Lightweight Security Solutions for The Internet of Things," Ph.D. dissertation, Swedish Institute of Computer Science (SICS), 2013.

[81] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the internet of things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, 2013.

[82] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, 2014.

[83] S. Raza, H. Shafagh, and O. Dupont, "Compression of Record and Handshake Headers for Constrained Environments," Internet Engineering Task Force, Internet-Draft draft-raza-dice-compressed-dtls-00, Sep. 2014, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-raza-dice-compressed-dtls-00

[84] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S A publish/subscribe protocol for Wireless Sensor Networks," *3rd International Conference on Communication Systems Software and Middleware and Workshops*, pp. 791–791, 2008.

[85] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," *Proceedings - 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015*, no. 4, pp. 746–751, 2015.

[86] P. Saint-Andre, K. Smith, and R. Tronon, *XMPP: the definitive guide*, 2009.

[87] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP) : Core," *Internet Engineering Task Force*, pp. 1–212, 2011.

[88] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, 2006.

[89] J. L. Fernandes, I. C. Lopes, J. J. P. C. Rodrigues, and S. Ullah, "Performance evaluation of RESTful web services and AMQP protocol," *International Conference on Ubiquitous and Future Networks, ICUFN*, pp. 810–815, 2013.