UNIT - V: Service layer protocols and Security:

***Service Layer** -oneM2M, ETSI M2M, OMA, BBF – Security in IoT Protocols – MAC 802.15.4 6LoWPAN, RPL, Application Layer.*

Service Layer one M2M :

ONEM2M

oneM2M Architecture:

1. The oneM2M architecture is based on a horizontal layering approach that consists of four layers, as shown below:
    a. Application Layer: The application layer is responsible for managing the IoT applications and services that interact with the oneM2M platform.
    b. Platform Services Layer: The platform services layer provides a set of common services that are used by IoT applications and services, such as security, device management, and data management.
    c. Infrastructure Layer: The infrastructure layer provides the underlying network and computing infrastructure required to support IoT applications and services.
    d. Device Layer: The device layer consists of the IoT devices and sensors that are connected to the oneM2M platform.

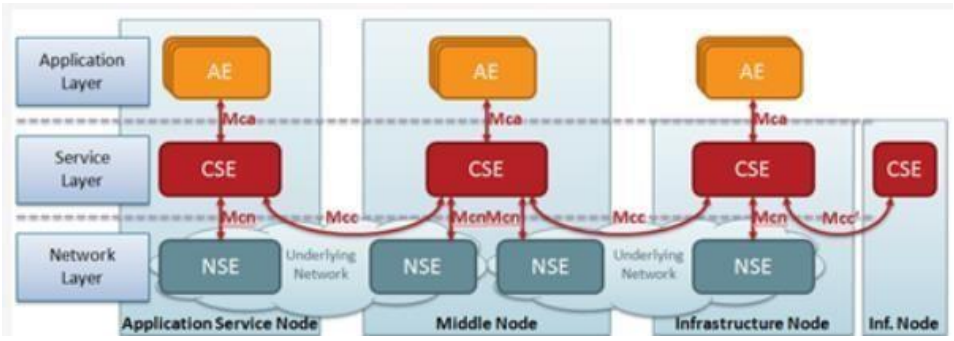

Figure 5.1 : OneM2m Functional Architecture

Benefits of oneM2M in IoT:

oneM2M offers several benefits when used in IoT applications. These include:

- Interoperability: oneM2M provides a common platform for IoT devices and services to communicate with each other, enabling interoperability between different IoT devices and platforms.

- Scalability: oneM2M can support a large number of devices and services, making it well suited for large-scale IoT deployments.
- Security: oneM2M provides a set of security mechanisms that can be used to secure IoT communications and data.
- Flexibility: oneM2M is a flexible standard that can be adapted to different IoT use cases and requirements.
- Global Reach: oneM2M is a global standard that can be used by IoT deployments worldwide, ensuring compatibility between different regions and countries.
    - oneM2M provides a common platform for IoT devices and services to communicate with each other, enabling interoperability between different IoT devices and platforms. The oneM2M architecture consists of four layers, providing a flexible and scalable framework for IoT deployments. The benefits of oneM2M include interoperability, scalability, security, flexibility, and global reach, making it a promising standard for the future of IoT.
    - OneM2M is a standard developed by a consortium of global organizations to provide a common platform for the interoperability of different IoT devices and services. It is a hierarchical architecture that enables seamless communication between different IoT devices and services, regardless of the underlying network technology.

- OneM2M has several features that make it well suited for IoT applications. These include:
    - Interoperability: OneM2M provides a common platform for different IoT devices and services to communicate with each other, regardless of the underlying network technology.
    - Scalability: OneM2M is highly scalable, enabling it to support a large number of devices and services.
    - Security: OneM2M provides built-in security features, including authentication, authorization, and encryption.
    - Device management: OneM2M provides a standardized way of managing IoT devices, including registration, discovery, and configuration.

Advantages of OneM2M in IoT:

OneM2M has several advantages when used in IoT applications. These include:

- Interoperability: OneM2M's common platform enables different IoT devices and services to

communicate with each other, regardless of the underlying network technology.

- Scalability: OneM2M's scalability makes it well suited for large-scale IoT deployments.
- Security: OneM2M's built-in security features provide a secure platform for IoT communications.
- Standardization: OneM2M provides a standardized platform that enables different IoT devices and services to communicate with each other, reducing the complexity of integrating different devices and services.

Limitations of OneM2M in IoT:

While OneM2M has several advantages, there are also some limitations to its use in IoT applications. These include:

- Complexity: OneM2M's hierarchical architecture can be complex to implement and maintain.
- Compatibility: OneM2M is not compatible with all IoT devices and services, requiring the use of gateways or translators to translate between OneM2M and other protocols.
- Adoption: OneM2M is a relatively new standard, and its adoption may be limited in some industries.

- In conclusion, OneM2M is a promising standard for IoT applications, offering several advantages such as interoperability, scalability, security, and standardization. However, it also has some limitations, particularly in terms of complexity, compatibility, and adoption. Overall, OneM2M is well suited for IoT deployments where interoperability, security, and scalability are critical, andwhere additional security measures can be implemented to ensure secure messaging.

ETSI :

- The Internet of Things (IoT) has gained significant attention in recent years as it has become increasingly popular to use interconnected devices to automate and optimize a variety of processes.
- To support this, a range of standards and protocols have been developed to ensure interoperability and facilitate the development of IoT applications.
- One of these standards is ETSI M2M. In this essay, we will discuss ETSI M2M in the context of IoT, outlining its features, advantages, and limitations.
- ETSI M2M is a standard developed by the European Telecommunications Standards Institute (ETSI) to provide a common platform for the interoperability of different IoT devices and services. It is a set of specifications and guidelines that define the requirements for M2M (machine-to- machine)

communication.

- Features of ETSI M2M:

ETSI M2M has several features that make it suitable for IoT applications. These include:

- Interoperability: ETSI M2M provides a common platform for different IoT devices and services to communicate with each other, regardless of the underlying network technology.
- Scalability: ETSI M2M is highly scalable, enabling it to support a large number of devices and services.
- Security: ETSI M2M provides built-in security features, including authentication, authorization, and encryption.
- Device management: ETSI M2M provides a standardized way of managing IoT devices, including registration, discovery, and configuration.

Advantages of ETSI M2M in IoT:

ETSI M2M has several advantages when used in IoT applications. These include:

- Interoperability: ETSI M2M's common platform enables different IoT devices and services to communicate with each other, regardless of the underlying network technology.
- Scalability: ETSI M2M's scalability makes it well suited for large-scale IoT deployments.
- Security: ETSI M2M's built-in security features provide a secure platform for IoT communications.
- Standardization: ETSI M2M provides a standardized platform that enables different IoT devices and services to communicate with each other, reducing the complexity of integrating different devices and services.

Limitations of ETSI M2M in IoT:

- While ETSI M2M has several advantages, there are also some limitations to its use in IoT applications. These include:
- Complexity: ETSI M2M's specifications and guidelines can be complex to implement and maintain.
- Compatibility: ETSI M2M is not compatible with all IoT devices and services, requiring the use of gateways or translators to translate between ETSI M2M and other protocols.

- Adoption: ETSI M2M is a relatively new standard, and its adoption may be limited in some industries.

- ETSI M2M is a promising standard for IoT applications, offering several advantages such as interoperability, scalability, security, and standardization. However, it also has some limitations, particularly in terms of complexity, compatibility, and adoption.

- Overall, ETSI M2M is well suited for IoT deployments where interoperability, security, and scalability are critical, and where additional security measures can be implemented to ensure secure messaging.

- The ETSI M2M architecture is designed to provide a standardized framework for machine-to-machine (M2M) communications in the Internet of Things (IoT). It is a layered architecture that defines the key components and interfaces necessary for M2M communications.

The ETSI M2M architecture comprises the following layers:

- Application Layer: This layer includes the M2M applications that use the ETSI M2M platform for communication. The applications communicate with the platform using standard protocols, such as HTTP or CoAP, and use ETSI M2M APIs to interact with other components of the architecture.

- Service Layer: This layer provides services to M2M applications, such as device management, data management, and security. The services are exposed as RESTful APIs, and applications can use them to manage and interact with M2M devices.

- Network Layer: This layer provides connectivity for M2M devices, including cellular networks, Wi-Fi, Ethernet, and others. The network layer also includes gateways that provide translation between different network technologies and protocols.

- Device Layer: This layer includes M2M devices that connect to the network layer. The devices can be sensors, actuators, or other types of devices that collect or act on data. The devices are managed by the service layer and can be configured, monitored, and updated remotely.

- The ETSI M2M architecture also includes several interfaces that enable communication between the layers. These interfaces include:

M2M Service Interface: This interface provides access to M2M services in the service layer.

- Device Management Interface: This interface enables the service layer to manage M2M devices in the device layer.

- Application Service Interface: This interface enables M2M applications to interact with the

service layer.

- Network Service Interface: This interface provides access to network services in the network layer.

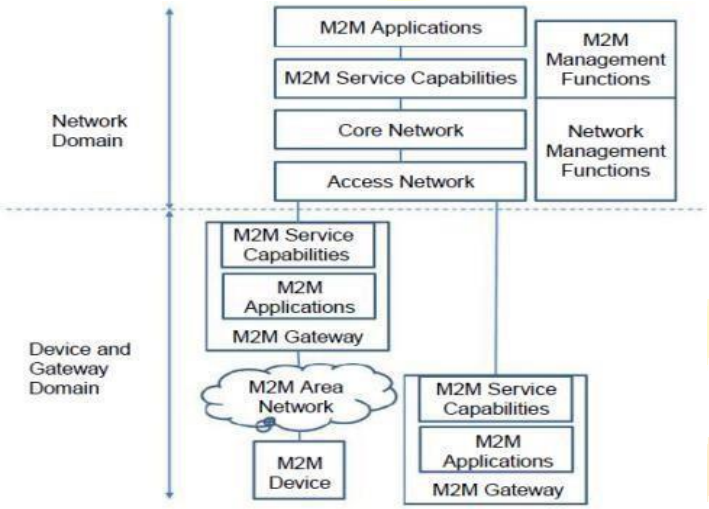

Figure 5.2 : ETSI Architecture

- The ETSI M2M architecture is designed to be modular and flexible, allowing different components to be added or removed as needed. It is also designed to be scalable and secure, with built-in security features such as authentication, authorization, and encryption.
- Overall, the ETSI M2M architecture provides a comprehensive framework for M2M communications in the IoT, enabling interoperability, scalability, and security across different devices, networks, and applications.

OMA

- OMA (Open Mobile Alliance) is a standards organization that develops specifications for mobile and IoT devices. In the context of IoT, OMA has developed several standards that define communication protocols, data models, and device management techniques. These standards are intended to enable interoperability between different IoT devices and platforms.
- Some of the key standards developed by OMA for IoT include:

  - Lightweight M2M (LwM2M): This is a device management protocol that enables remote management of IoT devices. It is designed to be lightweight and efficient, making it suitable for use in constrained environments. LwM2M defines a set of standard objects and interfaces for device management, and it uses CoAP as its underlying protocol.

    OMA DM (Device Management): This is a device management protocol that is used in mobile

devices and IoT devices. It enables remote management of devices over various types of networks, including cellular, Wi-Fi, and Ethernet. OMA DM defines a set of standard objects and interfaces for device management, and it uses HTTP or CoAP as its underlying protocol.

OMA Lightweight Machine-to-Machine (OMA LwM2M)

Enabler: This is a set of specifications that define a standardized framework for IoT device management. It includes specifications for device management, data management, and security.

- OMA LwM2M Enabler is designed to be scalable and flexible, allowing it to support a wide range of IoT devices and applications.
- OMA SensorThings API: This is a standard API for IoT sensors and devices that collect data. It provides a standardized way to access and manage sensor data, making it easier to integrate data from different sources. SensorThings API is based on RESTful principles and uses JSON as its data format.

- Overall, the OMA standards for IoT provide a comprehensive set of specifications that enable interoperability, security, and manageability in IoT devices and platforms. These standards are widely used in the industry and are supported by a large ecosystem of vendors and developers.
- OMA (Open Mobile Alliance) architecture for IoT is designed to enable interoperability, security, and manageability in IoT devices and platforms. The OMA architecture is based on a client-server model, where the IoT devices act as clients and the servers manage the devices and provide services.
- The OMA architecture consists of several layers, each with a specific function:
    - Application layer: This layer contains the applications and services that run on the IoT devices. The application layer interacts with the services provided by the server layer.
    - Device layer: This layer consists of the hardware and firmware that make up the IoT devices. The device layer interacts with the services provided by the server layer through the application layer.
    - Server layer: This layer provides the services that manage and control the IoT devices. The server layer consists of several components, including the device management server, data management server, and security server.
    - Gateway layer: This layer provides connectivity between the IoT devices and the server layer. The gateway layer may include devices such as routers, gateways, and access points.

- The OMA architecture uses standardized protocols and data models to enable interoperability between different IoT devices and platforms. Some of the key protocols and data models used in the OMA architecture include:
    - Lightweight M2M (LwM2M): This protocol is used for device management and enables remote management of IoT devices. LwM2M uses CoAP as its underlying protocol.
    - OMA DM (Device Management): This protocol is used for device management and enables remote management of IoT devices. OMA DM uses HTTP or CoAP as its underlying protocol.

OMA Lightweight Machine-to-Machine (OMA LwM2M)

Enabler: This set of specifications defines a standardized framework for IoT device management. It includes specifications for device management, data management, and security.

- OMA Sensor Things API: This API is used to access and manage sensor data from IoT devices.

Overall, the OMA architecture provides a comprehensive framework for building and managing IoT devices and platforms. It enables interoperability, security, and manageability, making it easier to develop and deploy IoT solutions.

BBF

1. BBF (Broadband Forum) is a global consortium of over 100 industryleading companies that develop and promote broadband network technologies, including those related to the Internet of Things (IoT).
2. The BBF's work in the IoT space is focused on defining standards and best practices for connecting IoT devices to broadband networks, enabling the development of more reliable, scalable, and secure IoT solutions.
3. The BBF's IoT work is organized around several key areas:

    3.1 Device management: The BBF's work in this area is focused on developing standards and best practices for managing large numbers of IoT devices connected to broadband networks. This includes defining protocols for device discovery, configuration, firmware updates, and diagnostics.

    3.2 Security: The BBF recognizes the importance of security in IoT deployments and has developed a range of security standards and best practices for IoT devices and networks. These include guidelines for securing IoT devices, as well as protocols for secure communication between

devices and the network.

3.3 Data analytics: The BBF is working to develop standards and best practices for collecting, analyzing, and using data generated by IoT devices. This includes developing standards for data formats, protocols for data transfer and storage, and best practices for data analytics and machine learning.

3.4 Interoperability: The BBF is committed to promoting interoperability between IoT devices and networks. This includes developing standards for device interoperability, as well as protocols for interoperable data exchange and communication.

3.5 One of the BBF's key contributions to the IoT space is its Open Broadband – IoT (OB-IoT) project, which aims to define a standardized architecture for IoT deployments on broadband networks. The OB-IoT architecture consists of several layers, each with a specific function:

3.6 Device layer: This layer consists of the IoT devices themselves, as well as the software and firmware that enable them to connect to broadband networks. The BBF has developed standards and best practices for device management, security, and data analytics in this layer.

3.7 Gateway layer: This layer provides connectivity between IoT devices and the broadband network. Gateways may be dedicated devices or integrated into broadband modems or routers. The BBF has developed standards and best practices for gateway management, security, and interoperability.

3.8 Network layer: This layer includes the broadband network infrastructure, including switches, routers, and other network devices. The BBF has developed standards and best practices for network management, security, and interoperability.

3.9 Application layer: This layer includes the applications and services that run on top of the broadband network, including cloud-based services and analytics platforms. The BBF has developed standards and best practices for application development, security, and interoperability.

4 The OB-IoT project is supported by a range of BBF members, including network operators, equipment vendors, and service providers. The project aims to accelerate the development and deployment of IoT solutions on broadband networks, promoting interoperability, security, and scalability.

5 In addition to the OB-IoT project, the BBF is involved in a range of other initiatives related to IoT, including the development of standards and best practices for smart home networks, smart cities, and

industrial IoT deployments.

6 Overall, the BBF's work in the IoT space is focused on defining standards and best practices that enable the development of more reliable, scalable, and secure IoT solutions. The OB-IoT project is a key part of this work, providing a standardized architecture for IoT deployments on broadband networks. With the support of its members, the BBF is well-positioned to drive innovation and promote interoperability in the IoT space.

7 Provide an overview of the BBF's work in IoT, focusing on the key standards and specifications developed by the consortium.

8 TR-069 is a specification developed by the BBF for the management of broadband networks. It provides a standard protocol for the remote management of network devices, including IoT devices.

9 The TR-069 specification enables the management of devices over a wide range of network types, including Ethernet, Wi-Fi, and cellular networks. The specification supports a range of management functions, including configuration, software updates, and performance monitoring.

10 The TR-069 specification has been widely adopted by the industry and is used by many service providers to manage their IoT networks. It has also been adopted by the European Telecommunications Standards Institute (ETSI) as a standard for the management of IoT devices.

11 The User Services Platform (USP) is a specification developed by the BBF for the management of IoT devices. USP is designed to provide a standard, secure, and scalable framework for the management of IoT devices, including those that are deployed in homes and businesses.

12 The USP specification is based on the TR-069 protocol and provides a standardized framework for the management of IoT devices, regardless of the type of network they are connected to.

13 USP includes a range of management functions, including device discovery, configuration, and software updates. It also includes support for remote diagnostics, troubleshooting, and security management.

14 USP has been adopted by the industry and is being used by many service providers to manage their IoT networks. It is also being promoted by the BBF as a standard for the management of IoT devices.

15 G.hn is a standard developed by the BBF for the provision of highspeed broadband services over any wired home network, including powerline, coaxial, and telephone line networks.

16 G.hn provides a standard protocol for the transmission of data over these networks, enabling the delivery of high-speed broadband services to IoT devices.

17 The G.hn standard has been widely adopted by the industry and is being used by many service providers to deliver broadband services to homes and businesses.

18 It is also being used to provide connectivity for IoT devices, enabling the deployment of IoT solutions over existing home networks.

19 FAN (Fixed Access Network) is a specification developed by the BBF for the management of broadband access networks. FAN provides a standardized framework for the management of broadband access networks, including those that are used for the delivery of IoT services.

20 FAN includes a range of management functions, including network discovery, topology discovery, and service discovery. It also includes support for network security, device management, and service management.

21 FAN has been adopted by the industry and is being used by many service providers to manage their broadband access networks. It is also being used to manage IoT networks, providing a standardized framework for the management of IoT devices.

22 Open Broadband - Broadband Access Abstraction (OB-BAA)-OBBAA is a specification developed by the BBF for the management of broadband access networks.

23 OB-BAA provides a standardized framework for the management of broadband access networks, including those that are used for the delivery of IoT services.

24 OB-BAA includes a range of management functions, including network discovery, topology discovery, and service discovery. It also includes support for network security, device management, and service management.

25 OB-BAA is being promoted by the BBF as a standard for the management of broadband access networks.

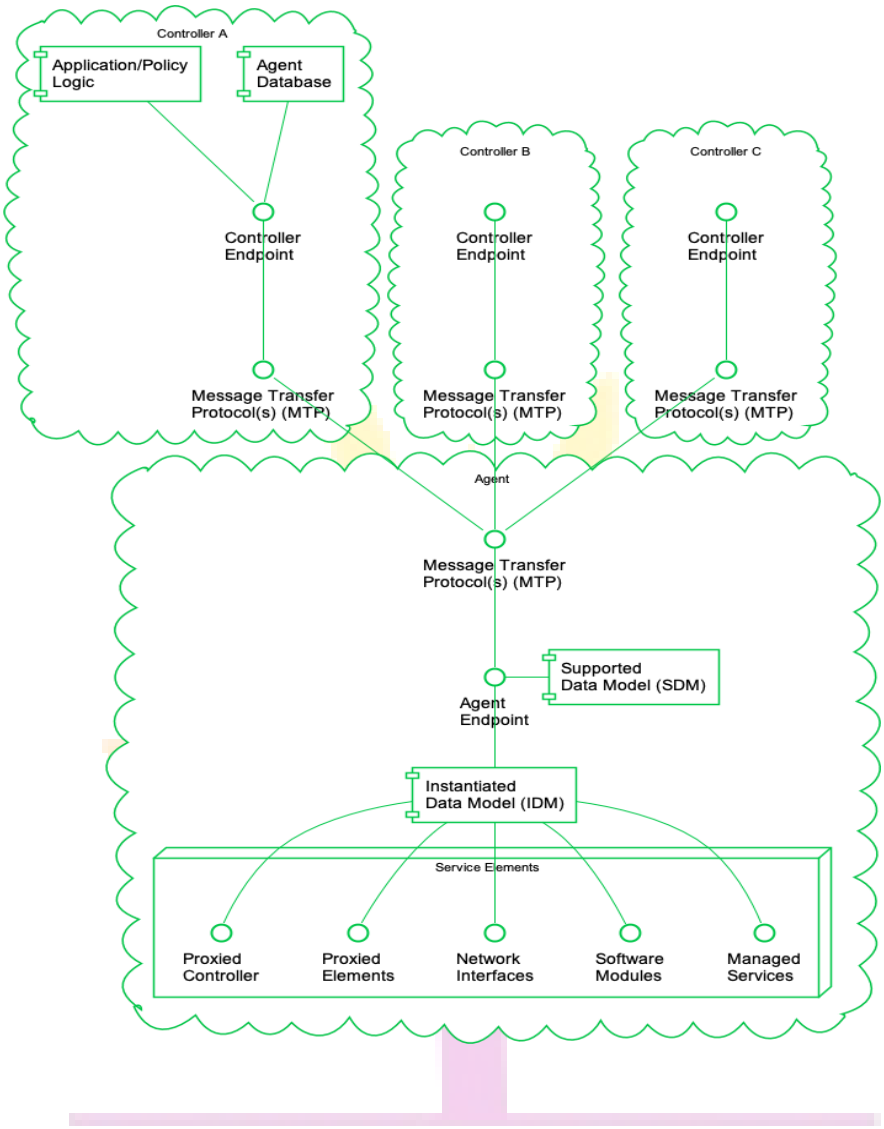

Figure 5.3 : USP Agent and controller Architecture

Security in IoT Protocols

Security is another aspect of IoT applications which is critical and can be found in all almost all layers of the IoT protocols. Threats exist at all layers including datalink, network, session, and application layers.

MAC 802.15.4

MAC 802.15.4 offers different security modes by utilizing the "Security Enabled Bit" in the Frame Control field in the header. Security requirements include confidentiality, authenticapapertion, integrity, access control mechanisms and secured Time-Synchronized Communications.

6LoWPAN

lo6LoWPAN by itself does not offer any mechanisms for security. However, relevant documents include

discussion of security threats, requirement and approach to consider in IoT network layer. For example, RFC 4944 discusses the possibility of duplicate EUI-64 interface addresses which are supposed to be unique [RFC4944]. RFC 6282 discusses the security issues that are raised due to the problems introduced in RFC 4944 [RFC6282]. RFC 6568 addresses possible mechanisms to adopt security within constrained wireless sensor devices [RFC6568]. In addition, a few recent drafts in [6Lo] discuss mechanisms to achieve security in 6loWPAN.

RPL

RPL offers different level of security by utilizing a "Security" field after the 4-byte ICMPv6 message header. Information in this field indicates the level of security and the cryptography algorithm used to encrypt the message. RPL offers support for data authenticity, semantic security, protection against replay attacks, confidentiality and key management. Levels of security in RPL include Unsecured, Preinstalled, and Authenticated. RPL attacks include Selective Forwarding, Sinkhole, Sybil, Hello Flooding, Wormhole, Black hole and Denial of Service attacks.

Application Layer

Applications can provide additional level of security using TLS or SSL as a transport layer protocol. In addition, end to end authentication and encryption algorithms can be used to handle different levels of security as required.

IoT Application Layer Protocols

When considering constrained networks and/or a large-scale deployment of constrained nodes, verbose web-

based and data model protocols, may be too heavy for IoT applications. To address this problem, the IoT industry is working on new lightweight protocols that are better suited to large numbers of con- strained nodes and networks. Two of the most popular protocols are CoAP and MQTT. Figure highlights their position in a common IoT protocol stack.

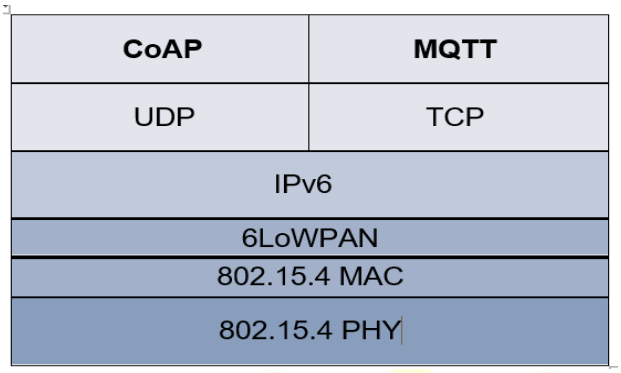

Figure 5.4 : Example of a High-Level IoT Protocol Stack for CoAP and MQTT

Constrained Application Protocol (CoAP) resulted from the IETF Constrained RESTful Environments (CoRE) working group's efforts to develop a generic framework for resource-oriented applications targeting constrained nodes and networks. (For more information on the IETF CoRE working group

The CoAP framework defines simple and flexible ways to manipulate sensors and actuators for data or device management. The IETF CoRE working group has published multiple standards-track specifications for CoAP, including the following:

- RFC 6690: Constrained RESTful Environments (CoRE) Link Format
- RFC 7252: The Constrained Application Protocol (CoAP)
- RFC 7641: Observing Resources in the Constrained Application Protocol (CoAP)
- RFC 7959: Block-Wise Transfers in the Constrained Application Protocol (CoAP)
- RFC 8075: Guidelines for Mapping Implementations: HTTP to the Constrained Application Protocol (CoAP) The CoAP messaging model is primarily designed to facilitate the exchange of messages over UDP between endpoints, including the secure transport protocol Datagram Transport Layer Security (DTLS). (UDP is discussed earlier in this chapter.) The IETF CoRE working group is studying alternate transport mechanisms, including TCP, secure TLS, and WebSocket. CoAP over Short Message Service (SMS) as defined in Open Mobile Alliance for Lightweight Machine-to-Machine (LWM2M) for IoT device manage- ment is also being considered. (For more information on the Open Mobile Alliance, see http://openmobilealliance.org.)

RFC 7252 provides more details on securing CoAP with DTLS. It specifies how a CoAP endpoint is provisioned with keys and a filtering list. Four security modes are defined: NoSec, PreSharedKey, RawPublicKey, and Certificate. The NoSec and RawPublicKey implementations are mandatory.

From a formatting perspective, a CoAP message is composed of a short fixed-length Header field (4 bytes), a variable-length but mandatory Token field (0–8 bytes), Options fields if necessary, and the Payload field. Figure details the CoAP message format, which delivers low overhead while decreasing parsing complexity.

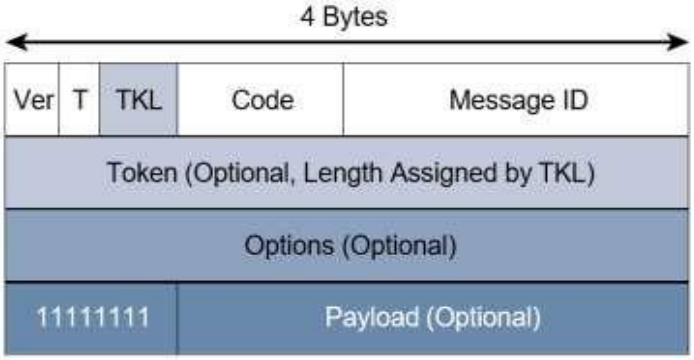

Figure 5.5: CoAP Message Format

In Figure, the CoAP message format is relatively simple and flex- ible. It allows CoAP to deliver low overhead, which is critical for constrained networks, while also being easy to parse and process for constrained devices.

Table 5.1 :CoAP Message Fields

CoAP Message Field	Description
Ver (Version)	Identifies the CoAP version.
T (Type)	Defines one of the following four message types: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), or Reset (RST). CON and ACK are highlighted in more detail in Figure 6-9.
TKL (Token Length)	Specifies the size (0–8 Bytes) of the Token field.
Code	Indicates the request method for a request message and a response code for a response message. For example, in Figure 6-9, GET is the request method, and 2.05 is the response code. For a complete list of values for this field, refer to RFC 7252.
Message ID	Detects message duplication and used to match ACK and RST message types to Con and NON message types.
Token	With a length specified by TKL, correlates requests and

responses.

Options	Specifies option number, length, and option value. Capabilities provided by the Options field include specifying the target resource of a request and proxy functions.
Payload	Carries the CoAP application data. This field is optional, but when it is present, a single byte of all 1s (0xFF) precedes the payload. The purpose of this byte is to delineate the end of the Options field and the beginning of Payload.

Message Queuing Telemetry Transport (MQTT)

At the end of the 1990s, engineers from IBM and Arcom (acquired in 2006 by Eurotech) were looking for a reliable, lightweight, and cost-effective protocol to monitor and control a large number of sensors and their data from a central server location, as typi- cally used by the oil and gas industries. Their research resulted in the development and implementation of the Message Queuing Telemetry Transport (MQTT) protocol that is now standardized by the Organization for the Advancement of Structured Information Standards (OASIS). (For more information on OASIS, see www.oasis-open.org.)

Considering the harsh environments in the oil and gas industries, an extremely simple protocol with only a few options was designed, with considerations for constrained nodes, unreliable WAN backhaul communications,

and bandwidth constraints with variable latencies. These were some of the rationales for the selection of a client/server and publish/subscribe framework based on the TCP/IP architecture, as shown in Figure

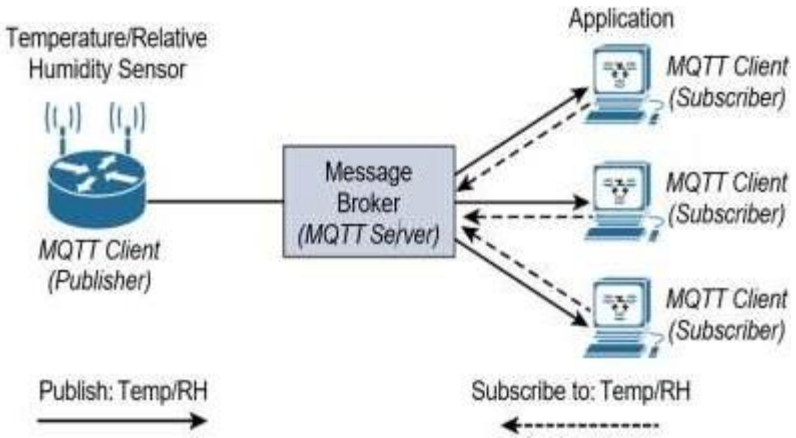

Figure 5.6: MQTT Publish/Subscribe Framework

An MQTT client can act as a publisher to send data (or resource information) to an MQTT server acting as an MQTT message broker. In the example illustrated in Figure, the MQTT client on the left side is a temperature (Temp) and relative humidity (RH) sen- sor that publishes its Temp/RH data. The MQTT server (or message broker) accepts the network connection along with application messages, such as Temp/RH data, from the publishers. It also handles the subscription and unsubscription process and pushes the application data to MQTT clients acting as subscribers.

The application on the right side of Figure is an MQTT client that is a subscriber to the Temp/RH data being generated by the publisher or sensor on the left. This model, where subscribers express a desire to receive information from publishers, is well known. A great example is the collaboration and social networking application Twitter. With MQTT, clients can subscribe to all data (using a wildcard character) or specific data from the information tree of a publisher. In addition, the presence of a message broker in MQTT decouples the data transmission between clients acting as publishers and sub- scribers. In fact, publishers and subscribers do not even know (or need to know) about each other. A benefit of having this decoupling is that the MQTT message broker ensures that information can be buffered and cached in case of network failures. This also means that publishers and subscribers do not have to be online at the same time. MQTT control packets run over a TCP transport using port 1883. TCP ensures an ordered, lossless stream of bytes between the MQTT client and the MQTT server. Optionally, MQTT can be secured using TLS on port 8883, and WebSocket (defined in RFC 6455) can also be used. MQTT is a lightweight protocol because each control packet consists of a 2-byte fixed header with optional variable header fields and optional payload. You should note that a control packet can contain a payload up to 256 MB. Figure provides an overview of the MQTT message format.

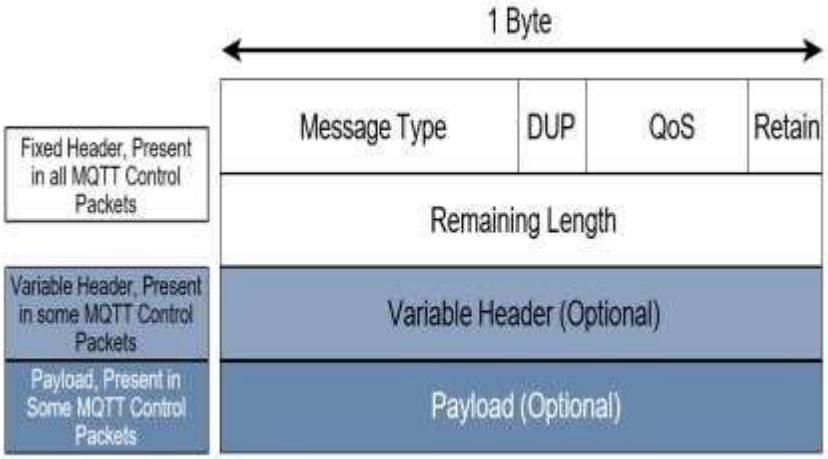

Figure 5.7: MQTT

Message Table : Format Comparison Between CoAP and MQTT

Factor	CoAP	MQTT
Main Transport Protocol	UDP	TCP
Typical Messaging	Request/response	Publish/subscribe
Effectiveness in LLNs	Excellent	Low/fair (Implementations pairing UDP with MQTT are better for LLNs.)
Security	DTLS	SSL/TLS
Communication Model	One-to-one	many-to-many
Strengths	Lightweight and fast, with low overhead, and suitable for constrained networks; uses a RESTful model that is easy to code to; easy to parse and process for constrained devices; support for multicasting; asynchronous and synchronous messages	TCP and multiple QoS options provide robust communications; simple management and scalabil- ity using a broker architecture
Weakness	Not as reliable as TCP-based MQTT, so the application must ensure reliability.	Higher overhead for constrained devices and networks; TCP connections can drain low-power devices; no multicasting support

Security

The IEEE 802.15.4 specification uses Advanced Encryption Standard (AES) with a 128-bit key length as the base encryption algorithm for securing its data. Established by the US National Institute of Standards and Technology in 2001, AES is a block cipher, which means it operates on fixed-size blocks of data. The use of AES by the US govern- ment and its widespread adoption in the private sector has helped it become one of the most popular algorithms used in symmetric key cryptography. (A *symmetric key* meansthat the same key is used for both the encryption and decryption of the data.) In addition to encrypting the data, AES in 802.15.4 also validates the data that is sent. This is accomplished by a message integrity code (MIC), which is calculated for the entire frame using the same AES key that is used for encryption.

Enabling these security features for 802.15.4 changes the frame format slightly and consumes some of the payload. Using the Security Enabled field in the Frame Controlportion of the 802.15.4 header is the first step to enabling AES encryption. This field is a single bit that is set to 1 for security. Once this bit is set, a field called the Auxiliary Security Header is created after the Source Address field, by stealing some bytes fromthe Payload field. Figure 4-8 shows the IEEE 802.15.4 frame format at a high level, withthe Security Enabled bit set and the Auxiliary Security Header field present.

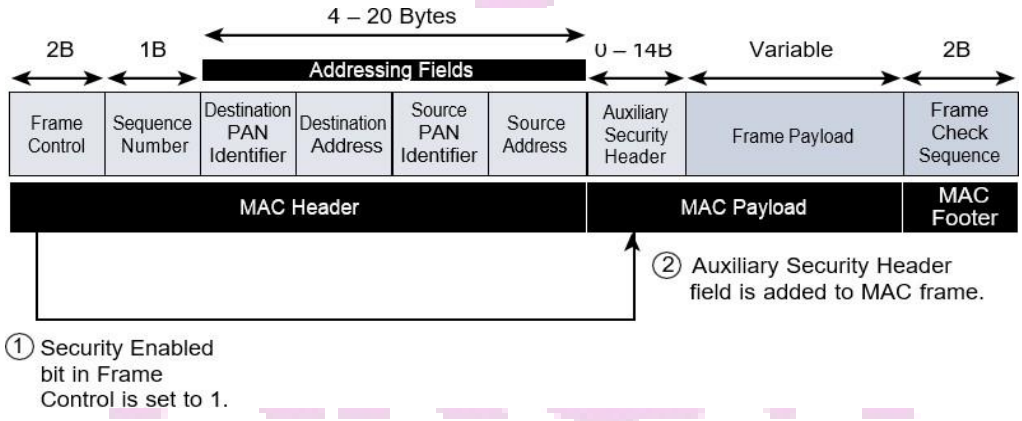

Figure 5.8 :Frame Format with the Auxiliary Security Header Field for 802.15.4-2006 and Later Versions

Security in IEEE 802.15.4 The IEEE 802.15.4-2011 standard provides security services at the MAC layer that, despite being designed to secure communications at the link layer, are valuable in supporting security mechanisms designed at higher layers of the protocol stack illustrated in Fig. This is motivated by the support of efficient symmetric cryptography at the hardware in IEEE 802.15.4 sensing platforms. For example, current sensing platforms employing the cc2420 single-chip RF transceiver from Texas Instruments, as the TelosB mote from Crossbow, support IEEE 802.15.4 security and symmetric cryptography at the hardware using the Advanced Encryption Standard (AES)

Security Modes in The IEEE 802.15.4 Standard

Security mode	Security provided
No Security	Data is not encrypted Data authenticity is not validated
AES-CBC-MAC-32	Data is not encrypted Data authenticity using a 32-bit MIC
AES-CBC-MAC-64	Data is not encrypted Data authenticity using a 64-bit MIC
AES-CBC-MAC-128	Data is not encrypted Data authenticity using a 128-bit MIC
AES-CTR	Data is encrypted Data authenticity is not validated
AES-CCM-32	Data is encrypted Data authenticity using a 32-bit MIC
AES-CCM-64	Data is encrypted Data authenticity using a 64-bit MIC
AES-CCM-128	Data is encrypted Data authenticity using a 128-bit MIC

Security Modes: The IEEE 802.15.4 standard support various security modes at the MAC layer, which are described in Table. The available security modes are distinguished by the security guarantees provided and by the size of the integrity data employed. Fig. illustrates the application of security to an IEEE 802.15.4 link-layer data frame. A protected frame is identified by the Security Enabled Bit field of the Frame Control field being set at the beginning of the header. The Auxiliary Security Header is employed only when security is used, and identifies how security is applied to the frame. In the Auxiliary Security Header, the Security Control field identifies the Security Level mode from the modes identified in Table I, and how the cryptographic key required to process security for the link-layer frame is to be determined by the sender and receiver. The standard employs 128-bit keys that may be known implicitly by the two communication parties, or on the other end determined from information transported in the Key Source and Key Index subfields of the Key Identifier field. The Key Source subfield specifies the group key originator, and the Key Index subfield identifies a key from a specific source. The various security modes require the transportation of security-related information in different configurations, as in Fig. In our following discussion we identify how fundamental security requirements are assured by security at the MAC. Confidentiality: Security as currently defined by IEEE 802.15.4 is optional, given that an application may opt for no security or for security at others layers of the protocol stack. For applications requiring only confidentiality of link layer communications, the transmitted data may be encrypted using AES in the Counter (CTR) mode, using the AES-CTR security mode. As with all the security modes available at the IEEE 802.15.4 MAC layer, 128-bit keys are used to support this requirement. Data Authenticity and Integrity: Applications requiring authenticity and integrity of link-layer communications may use one of the security modes employing AES in the Cypher Block Chaining (CBC) mode, which produces a Message Integrity Code (MIC) or Message Authentication Code (MAC) appended to the transmitted data. The security modes supporting this are

AES-CBC-MAC-32, AES-CBC-MAC-64 and AES-CBC-MAC-128, which differ on the size of the integrity code produced. This code is created with information from the 802.15.4 MAC header plus the payload data, and in such security modes the payload is transmitted unencrypted. Confidentiality, Data Authenticity and Integrity: The CTR and CBC modes may be jointly employed using the combined Counter with CBC-MAC AES/CCM encryption mode, which in IEEE

802.15.4 is used to support confidentiality as well as data authenticity and integrity for link-layer communications. This mode is supported in sensing platforms such as the TelosB in the CCM∗ variant, which also offers provides for integrity only and encryption-only security. This usage mode of AES provides confidentiality, message integrity and authenticity for data communications. The security modes are AES- CCM-32, AES-CCM-64 and AES-CCM-128, which again differ on the size of the MIC code following each message. AES-CCM modes require the transportation of all the security-related fields after the encrypted payload, as is illustrated in Figure.

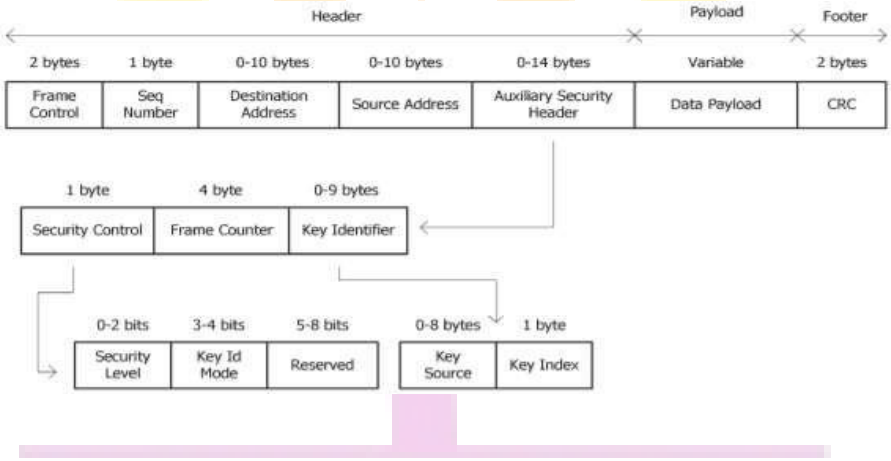

Figure 5.9: Security data and control fields in IEEE 802.15.4.

Security in 6LoWPAN No security mechanisms are currently defined in the context of the 6LoWPAN adaptation layer, but the relevant documents include discussions on the security vulnerabilities, requirements and approaches to consider for the usage of network layer security, as we proceed to discuss. Later in Section

- we analyze research proposals on approaches to 6LoWPAN security, as well as the open research challenges and opportunities. Identification of Security Vulnerabilities: The discussion regarding security on RFC 4944 is related to the possibility of forging or accidentally duplicating EUI-64 interface addresses, which may consequently compromise the global uniqueness of 6LoWPAN interface identifiers.
- This document also discusses that Neighbor Discovery and mesh routing mechanisms on IEEE 802.15.4 environments may be susceptible to security threats, and that AES security at the link-layer may provide a basis for the development of mechanisms protecting against such threats, particularly

for very constrained devices. Other interesting discussion is on the possibility of employing more powerful 6LoWPAN devices in order to support heavy securityrelated operations, also because such devices may support existing Internet security protocols, as such representing strategic places for the enforcement of security control mechanisms.

- The discussion concerning security on RFC 6282 focuses on the security issues posed by the usage of a mechanism inherited from RFC 4944, which enables the compression of a particular range of 16 UDP port numbers down to 4 bits. This document discusses that the overload of ports in this range, if employed with applications not honoring the reserved set for port compression, may increase the risk of an application getting the wrong type of payload or of an application misinterpreting the content of a message. As a result, RFC 6282 recommends that the usage of such ports be associated with a security mechanism employing MIC codes. Identification of Security Requirements and Strategies:
- The informational RFC 4919 discusses the addressing of security at various complementary protocol layers of the stack illustrated in Fig, considering that the most appropriate approach may depend on the application requirements and on the constraints of particular sensing devices. This document also identifies the possibility of employing security at the network layer using IPSec, together with the interest in investigating its applicability in the transport and tunnel usage modes.
- The discussion on security in RFC 6568 focuses on the possible approaches to adopt security in the light of the characteristics and constraints of wireless sensing devices. This document discusses threats due to the physical exposure of such devices, which may pose serious demands for its resiliency and survivability. It also discusses how IEEE 802.15.4 communications may facilitate attacks against the confidentiality, integrity, authenticity and availability of 6LoWPAN devices and communications. Rather than providing a specific approach to routing in 6LoWPAN environments, RFC 6606 provides guidelines that are useful in designing specific routing approaches. As with the previous standard documents, RFC 6606 identifies the importance of addressing security and the usefulness of AES/CCM available at the hardware of IEEE 802.15.4 sensing platforms.
- This document also discusses the importance of designing security mechanisms that are able to adapt to changes in the network topology and devices, rather than employing a static security configuration, given that many 6LoWPAN applications may employ networks that are dynamic in such respects.
- This document also discusses the importance of time synchronization, self-organization and security localization in providing security for data and multi-hop routing control packets. Other important security requirements identified are the support of authenticated broadcasts and

multicasts, and the verification of bidirectional links. RFC 6775 focuses on optimizations to enable

- Neighbor Discovery (ND) operations in 6LoWPAN environments, and also on the application of the threat model for ND operations defined in RFC 4861 to 6LoWPAN environments. Other possibilities discussed in this document consists in the adaptation of the SEcure Neighbor Discovery (SEND) and cryptographically generated addresses [36] mechanisms to 6LoWPAN environments.

Security in RPL : The RPL specification defines secure versions of the various routing control messages, as well as three basic security modes. In Fig. illustrate the format of a secure RPL control message, transporting a Security field after the 4-byte ICMPv6 message header.

- The high order bit of the RPL Code field identifies whether or not security is applied to a given RPL message, which may thus be a secure DIS, DIO, DAO or DAOACK message. The format of the Security field is illustrated in Figure.
- The information in the Security field indicates the level of security and the cryptographic algorithms employed to process security for the message. What this field doesn't include is the security—related data required to process security for the message, for example a Message Integrity Code (MIC) code or a signature. Instead, the security transformation itself states how the cryptographic fields should be employed in the context of the protected message. Support of Integrity and Data Authenticity: The current RPL specification defines the employment of AES/CCM with 128-bit keys for MAC generation supporting integrity, and of RSA with SHA-256 for digital signatures supporting integrity and data authenticity.
- The LVL (Security Level) field indicates the provided packet security and allows for varying levels of data authentication and, optionally, of data confidentiality. RFC 6550 also defines various values to identify the presence of confidentiality, integrity and data authenticity with MAC- 32 and MAC-64 authentication codes, as well as of 2048 and 3072- bit signatures using RSA. Support of Semantic Security and Protection Against Replay Attacks:
- A Consistency Check (CC) control message enables a sensing node to issue a challenge-response with the goal of validating another node's current counter value, for example in situations when a received message has an initialized (zero value) counter value and the receiver has an incoming counter currently maintained for the message originator. In this case the receiver initiates counter resynchronization by sending a CC message to the message source. Semantic security and protection against packet replay attacks is provided with the help of the Counter field, which may be used to transport a timestamp, as indicated by the T in Fig. The next byte in the Security section of the RPL control message identifies the security suite employed to provide security, while the Flags field is currently reserved.
- Support of Confidentiality: The secure variant of the various RPL control messages may also support

confidentiality and delay protection. Regarding the employment of cryptographic algorithms in RPL, AES/CCM is adopted as the basis to support security in the current specification, while we note that other algorithms may be adopted in the future and appropriately identified in the Security section of a secure RPL control message.

- RPL control messages may be protected using both an integrated encryption and authentication suite, such as with AES/CCM, as well as schemes employing separate algorithms for encryption and authentication. The entire RPL message is within the scope of RPL security. MAC codes and signatures are calculated over the entire unsecured IPv6 packet, with the mutable fields of the packet zeroed.
- When a RPL ICMPv6 message is encrypted, encryption starts at the first byte after the Security section and continues to the last byte of the packet. The IPv6 header, the ICMPv6 header and the RPL message, up to the start of the Security field, are not encrypted, since those fields are required to correctly decrypt the packet.

Support for Key Management: The KIM (Key Identifier Mode) field of the Security section illustrated in Figure indicates whether the cryptographic key required to process security for this message may be determined implicitly or explicitly. RFC 6550 currently defines different values for this field to thus supports different key management approaches, namely group keys, keys per pair of sensing devices, and digital signatures. This field supports various levels of granularity of packet protection, and is divided in a key source and key index subfields. The key source subfield indicates the logical identifier of the originator of a group key, while the key index subfield, when present, allows unique identification of keys with the same originator.

Security Modes in RPL: As previously discussed, RPL defines how security is applied to routing control messages, and the current specification also defines the following three security modes:

- **Unsecured:** in this mode no security is applied to routing control messages, and this is the default usage mode of RPL.
- **Preinstalled:** this security mode may be employed by a device using a preconfigured symmetric key in order to join an existent RPL instance, either as a host or a router. This key is employed to support confidentiality, integrity and data authentication for routing control messages.

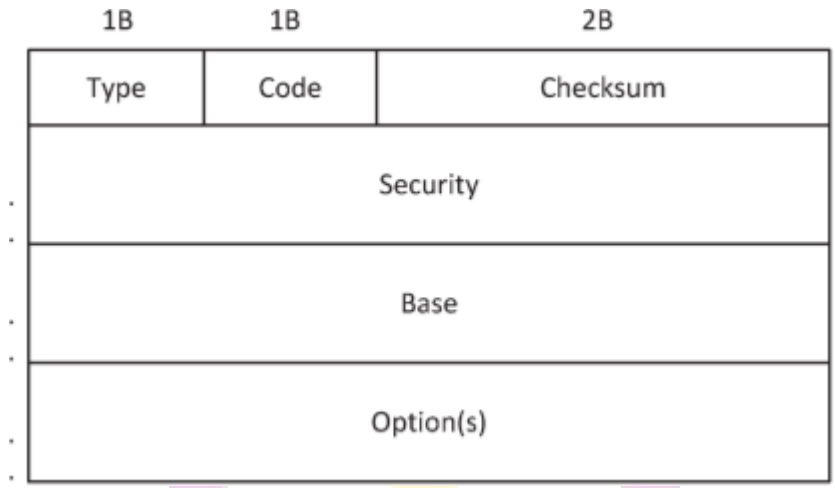

Figure: Secure RPL control message.

Figure 5.10: Security section of a secure RPL control message

Authenticated: this security mode is appropriate for devices operating as routers. A device may initially join the network using a preconfigured key and the preinstalled security mode, and next obtain a different cryptographic key from a key authority with which it may start functioning as a router. The key authority is responsible for authenticating and authorizing the device for this purpose.

The RPL specification currently defines that the authenticated security mode must not be supported by symmetric cryptography, although it doesn't specify how asymmetric cryptography may be employed to support node authentication and key retrieval by the device intending to operate as a router. A more clear definition of such mechanisms is thus required, and future versions of the RPL standard may more clearly define how to support them. While not introducing additional security mechanisms, other documents relevant to RPL also include analysis on security aspects. This is the case of the informational RFC documents discussing routing requirements for the various application areas. Such documents discuss the importance of protecting routing control messages with appropriate confidentiality, authentication and integrity. RFC 6551 specifies a set of link and node routing metrics appropriate to the characteristics and constraints of 6LoWPAN environments, and discusses the necessity of handling such metrics in a secure and trustful manner, including protection against nodes being able to falsify or lie in the advertisement of metrics, as a way to protect against attacks on routing operations.

SECURITY FOR IOT APPLICATION-LAYER COMMUNICATIONS

Application-layer communications are supported by the CoAP protocol, currently being designed by the Constrained RESTful Environments (CoRE) working group of the IETF. We next discuss the operation of the protocol as well as the mechanisms available to apply security to CoAP communications. A. Application-Layer Communications With CoAP The CoAP protocol implements a set of techniques to compress application- layer protocol metadata without compromising application inter-operability, in conformance with the representational state transfer (REST) architecture of the web. CoAP is currently defined only for UDP communications over 6LoWPAN, although the adoption of transport-layer approaches with characteristics more close to protocols such as the Transmission Control Protocol (TCP) is still open to debate, with ongoing research addressing the adaptation of TCP for 6LoWPAN environments. Application-layer communications may enable IoT sensing applications to interoperate with existing Internet applications without requiring specialized application oriented code or translation mechanisms. CoAP restricts the HTTP dialect to a subset that is well suited to the constraints of 6LoWPAN sensing devices, and may enable abstracted communications Fig. 9. Format of a CoAP message header. between users, applications and such devices, in the context of IoT applications. The CoAP protocol provides a request and response communications model between application endpoints and enables the usage of key concepts of the web, namely the usage of URI addresses to identify the resources available on constrained sensing devices. The protocol may support end to-end communications at the application-layer between constrained IoT sensing devices and other Internet entities, using only CoAP or in alternative by translating HTTP to CoAP at a reverse or forward gateway. Messages in the CoAP protocol are exchange asynchronously between two endpoints, and used to transport CoAP requests and responses. Since such messages are transported over unreliable UDP communications, CoAP provides a lightweight reliability mechanism. Using this mechanism CoAP messages may be marked as Confirmable, for which the sender activates a simple stop-and-wait retransmission mechanism with exponential backoff. The receiver must acknowledge a Confirmable message with a corresponding Acknowledge message or, if it lacks context to process the message properly, reject it with a Reset message. Acknowledge or Reset messages are related to a Confirmable message by means of a Message ID, along with the address of the corresponding endpoint. CoAP messages may also be transmitted less reliably if marked as Non-Confirmable, in which case the recipient does not acknowledge the message. Similarly to HTTP, CoAP defines a set of method and response codes available to applications. Other than a basic set of information, most of the information in CoAP is transported using options. Options defined for the CoAP Protocol may be critical, elective, safe or unsafe. A critical option is one that an endpoint must understand, while an elective option may be ignored by an endpoint not recognizing it. Safe and unsafe options determine how an option may be processed by an intermediary entity. An unsafe

option needs to be understood by the proxy in order to be forwarded, while a safe option may be forwarded even if the proxy is unable to process it. The CoAP header and message format is illustrated in Fig. The message starts with a 4-byte fixed header, formed by the Version field (2 bits), the T (message type) field (2 bits), the TKL (Token Length) field (4 bits), the Code field (8 bits) and the Message ID (16 bits). The token in practice enables a CoAP entity to perform matching of CoAP requests and replies, while the message ID supports duplicate detection and optional reliability. The options adopted in CoAP are defined in the Type-lengthvalue (TLV) format, by specifying its option number followed by its length and value. CoAP currently defines the Uri-Host, Uri-Port, Uri-Path and Uri-Query options enabling the identification of the target resource of a request, Content-Format to specify the representation format of the message payload, and Max-Age to indicate the maximum time a CoAP response may be cached before being considered not fresh, among others. Regarding security, rather than designing mechanisms to support (object) security directly in the context of application layer communications, CoAP adopts DTLS at the transport layer to transparently apply security to all CoAP messages in a given communications session. The protocol also defines four security modes.

Security in CoAP The CoAP Protocol defines bindings to DTLS (Datagram Transport-Layer Security) [45] to secure CoAP messages, along with a few mandatory minimal configurations appropriate for constrained environments. Support for Confidentiality, Authentication, Integrity, NonRepudiation and Protection Against Replay Attacks: The adoption of DTLS implies that security is supported at the transport-layer, rather than being designed in the context of the application-layer protocol. DTLS provides guarantees in terms of confidentiality, integrity, authentication and non-repudiation for application-layer communications using CoAP. DTLS is in practice TLS with added features to deal with the unreliable nature of UDP communications. Fig.illustrates the availability of payload space for applications in IEEE 802.15.4 and 6LoPWAN communication environments in the presence of CoAP and DTLS. Once the initial DTLS handshake is completed, DTLS adds a limited per-datagram overhead of 13 bytes, not counting any initialization vectors, integrity check values or the padding that may be required by the cipher suite employed. As considered in Fig. 10, shared-context 6LoWPAN header compression requires 10 bytes for an UDP/IPv6 header, while the CoAP fixed header requires 4 bytes. The impact of DTLS on constrained wireless sensing devices is due to the cost of supporting the initial handshake plus the processing of security for each exchanged CoAP messages. The impact of DTLS on constrained wireless sensing devices is due to the cost of supporting the initial handshake plus the processing of security for each exchanged CoAP messages. Similarly to other approaches to security in 6LoWPAN environments, AES/CCM is adopted as the cryptographic algorithm to support fundamental security requirements in the current CoAP specification. Security against replay attacks may also be achieved in the context of DTLS, using a different nonce value for each secured CoAP packet. Security Modes in

CoAP: In addition to the adoption of DTLS, CoAP currently defines four security modes that applications may employ. Those security modes essentially differ on how authentication and key negotiation is performed, as follows: • NoSec: this mode in practice provides no security, and CoAP messages are transmitted without security applied. • PreSharedKey: this security mode may be employed by sensing devices that are pre-programmed with the symmetric cryptographic keys required to support secure communications with other devices or groups of devices. This mode may be appropriate to applications employing devices that are unable to support public- key cryptography, or for which it is convenient to employ security preconfiguration. Applications may use one key per destination device or in alternative a single key for a group of destination devices. • RawPublicKey: this security mode is appropriate for devices requiring authentication based on public keys, but which are unable to participate in public-key infrastructures. A given device must be preprogrammed with an asymmetric key pair that may be validated using an outof-band mechanism and possibly programmed as part of the manufacturing process, while without a certificate. The device has an identity calculated from its public key and a list of identities and public keys of the nodes it can communicate with. This security mode is defined as mandatory to implement in CoAP. • Certificates: this security mode also supports authentication based on public-keys, but for applications that are able to participate in a certification chain for certificate validation purposes. This security mode thus assumes the availability and usage of a security infrastructure. The device has an asymmetric key pair with an X.509 certificate that binds it to its Authority Name and is signed by some common trusted root. The device also has a list of root trust anchors that can be used for certificate validation.

An important aspect of CoAP security using DTLS is that Elliptic Curve Cryptography (ECC) [48] is adopted to support the RawPublicKey and Certificates security modes. ECC supports device authentication using the Elliptic Curve Digital Signature Algorithm (ECDSA), and also key agreement using the ECC Diffie-Hellman counterpart, the Elliptic Curve Diffie-Hellman Algorithm with Ephemeral keys (ECDHE). The NoSec security mode corresponds to a device sending packets without security, using the “coap” scheme in URI addresses identifying resources available on CoAP servers. On the other end, accesses to resources with DTLS use the “coaps” scheme, and in this case a security association at the transport-layer using DTLS must exist between the CoAP client and the CoAP server.

The current CoAP specification defines a mandatory-toimplement cipher suite for each security mode, based on the usage of AES/CCM and ECC cryptographic operations, as follows:

• Applications supporting the PreSharedKey security mode are required to support at least the TLS_PSK_WITH_AES_ 128_CCM_8 suite, which supports authentication using pre-shared symmetric keys and 8-byte nonce values, and encrypts and produces 8-byte integrity codes.

• Applications supporting the RawPublicKey CoAP security mode are required to support the TLS_ECDHE_ ECDSA_WITH_AES_128_CCM_8 security suite using ECDSA-capable public keys. This security mode also employs SHA-256 to compute hashes.

• Applications supporting the Certificates security mode are also required to support the TLS_ECDHE_ECDSA_ WITH_AES_128_CCM_8 cipher suite. Regarding the usage of public-keys transported in X.509 certificates, the SubjectPublicKeyInfo field in a X.509 certificate defines how the corresponding public key must be employed for ECC computations. The certificate must also contain a signature created using ECDSA and SHA-256. Applications using devices with a shared key plus a certificate must also support TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA.

In addition to the cipher suites it may be expected that further security suites may be adopted in future versions of CoAP, as this would enable a better adaptation of the various security modes to different applications and types of sensing platforms. CoAP also doesn't currently define or adopt any solution to address key management, other than the assumption that initial keys are available resulting from the DTLS authentication handshake.

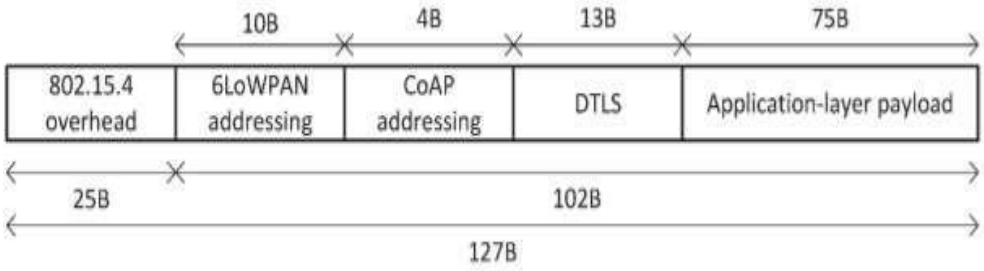

Figure 5.11: Payload space with DTLS on 6LoWPAN environments.