Securing the Constrained Application Protocol (CoAP) for the Internet of Things (IoT)

by

Mohammed Hassan Alamri
B.Sc., Taif University, 2011

A Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Securing the Constrained Application Protocol (CoAP) for the Internet of Things (IoT)

by

Mohammed Hassan Alamri
B.Sc., Taif University, 2011

Supervisory Committee

Dr. Fayez Gebali, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Kin F. Li, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

The Internet of Things (IoT) technology has become a hot topic over the last decade around the world. The availability of inexpensive components of IoT devices enables a wide range of applications and provide smart environments. These devices perform actuating and sensing tasks and identified through unique addresses. The IoT devices are connected to the Internet and expected to use the Constrained Application Protocol (CoAP) at the application layer as a main web transfer protocol. This protocol is a general protocol and does not provide secure channels for transferred data. One of the possible ways to provide security to this protocol is by using Datagram Transport Layer Security (DTLS) protocol for authentication functionality as well as end-to-end security to protect the transmission of sensitive information. In this project, we provide a brief overview of the CoAP and DTLS protocols. We simulate the CoAP protocol without security using the Contiki operating system and then we simulate the DTLS protocol over CoAP. We make a comparison between the two experiments in terms of memory footprint and power consumption since these two factors are the most concern factors in constrained devices in any IoT environment.

Contents

Supervisory Committee **ii**

Abstract **iii**

Table of Contents **iv**

List of Tables **vi**

List of Figures **vii**

Acknowledgements **viii**

Dedication **ix**

1 Introduction **2**

1.1 Motivation 2

1.2 Project goal 3

1.3 Outline 3

2 Overview of Internet of Things **4**

2.1 IoT Overview 4

2.2 IoT vision and concept 6

2.3 IoT features and requirements 7

2.4 IoT architecture 8

3 The Constrained Application Protocol **9**

3.1 CoAP Overview 9

3.2 CoAP Messaging model 10

3.3 Message format 12

3.4 CoAP Requests and Responses 13

3.4.1 Request Method Definitions 13
3.4.2 Response Code Definitions 13

4 DTLS-Secured CoAP **16**
4.1 DTLS Overview 16
4.2 DTLS for CoAP security 17

5 Simulation and experimental results **20**
5.1 CoAP Simulation 20
5.2 DTLS over CoAP (CoAPs) 21
5.3 Experiment Results 22

6 Conclusions **26**

Bibliography **27**

List of Tables

Table 5.1 Classes of Constrained Devices 23
Table 5.2 Memory Footprint 23
Table 5.3 Data from Powertrace for CoAP 24
Table 5.4 Power consumption for CoAP (mW) 25

List of Figures

Figure 2.1 A new dimension to the telecommunication environment [1]. 5
Figure 2.2 IoT architecture 8

Figure 3.1 CoAP protocol layers 10
Figure 3.2 Reliable Message Transmission 11
Figure 3.3 Unreliable Message Transmission 11
Figure 3.4 CoAP Message format 12
Figure 3.5 Client-Server model 13
Figure 3.6 A GET Request with a Piggy-Backed Response 14
Figure 3.7 A GET Request with a Separate Response 15
Figure 3.8 A request and a Response Carried in NON-confirmable Message . . 15

Figure 4.1 DTLS Handshake Process 16
Figure 4.2 Abstract Layering of DTLS-Secured CoAP 18
Figure 4.3 CoAP protocol layers 19

Figure 5.1 Retrieving CoAP's resources from Copper's interface 21
Figure 5.2 Encrypting a CoAP packet using DTLS 22
Figure 5.3 Sending a DTLS decrypted to CoAP 22
Figure 5.4 RAM Footprint 24
Figure 5.5 ROM Footprint 24

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful

Alhamdulillah, all praises belongs to Allah the merciful for his blessing and guidance. He gave me the strength to reach what I desire.
I would like to thank:

My parents, my family, for supporting me at all stages of my education and their unconditional love.

My Supervisor, Dr. Fayez Gebali, for all the support, feedback, guidance, and encouragement during my study.

DEDICATION

To my parents, **Hassan and Zainh** for their love, prayers, and encouragement.

To my beloved wife, **Sahar** for always standing by me, and believing in me.

To my lovely son, **Ryan**, and to my siblings for their encouragement and prayers.

Acronyms

IoT..... Internet of Things
CoAP..... Constrain Application Protocol
DTLS..... Datagram Transport Layer Security
LLN..... Low power and Lossy Network
IETF..... Internet Engineering Task Force
TLS..... Transport Layer Securityl
IPsec..... Internet Protocol Security
ICT..... Information and Communications Technology
RFID..... Radio Frequency IDentification
HTTP..... Hypertext Transfer Protocol
M2M..... Machine to Machine
UDP..... User Datagram Protocol
TCP..... Transmission Control Protocol
SMS..... Short Message Service
SCTP..... Stream Control Transmission Protocol
URI..... Uniform Resource Identifier
URL..... Uniform Resource Locator

Chapter 1

Introduction

1.1 Motivation

Nowadays, billions of people use the Internet for surfing the web, using social networking applications, sending and receiving emails, banking, shopping online, playing games and many other tasks. In addition, the Internet has been used for controlling and managing our lifestyle and making it easier. Imagine a world or a time where everything around us, from smart tiny objects, smartphones, smart eyeglasses, smart homes, smart TVs, air-conditioners, light bulbs, refrigerators, vehicles are connected to each other and to the Internet. Also, they are controlled, monitored, and sometimes, actuating, interacting among themselves, this is the world of the Internet of things (IoT).

It is believed that the Internet of Things will connect many of the objects around us to the Internet. It will create an intelligent environment and will increase the quality of life since it could be deployed in everything and will allow access from anyplace anytime. The IoT will create smart environments including smart buildings, smart businesses, smart transportations and smart cities. Moreover, the IoT technology is recognized as one of the most significant areas of future technology, and it is grabbing the attention of governments and researchers as well as industries.

The IoT enables a wide range of applications with potentially critical functions of sensing and actuating which need to be secure. Sensitive data and information will be collected by sensors, cameras, and other devices from different users, for instance, companies, governments, hospitals and other users. Thus, security remains to be one of the most sensitive issues given the danger if the system falls under attack. Because of the combination of both objects and communication networks, the IoT applications can be vulnerable in physical and cyber space.

The security of the IoT is still a big concern as all new technologies and services depend on securing the information and protection of private data to success. Security challenges occur in IoT at various levels due to the idea of this technology for many reasons. First, most of the IoT communications are wireless, which could be vulnerable to attack. Second, a huge number of nodes and amount of data exchanged. Third, some of IoT applications will be vulnerable to physical attack. Finally, limited computation,

transmission and energy capabilities in most of the IoT components which lead to a limited implementation of complex security schemes.

1.2 Project goal

The interconnection of highly heterogeneous devices in a Low power and Lossy Network (LLN), as well as small devices that usually have 8-bit microcontrollers and a small amount of RAM and ROM is a big challenge in providing security for the IoT. The CoAP was designed by the Internet Engineering Task Force (IETF) [2] to be used in IoT constrained devices as a web transfer protocol at the application layer. CoAP is an LLN optimized version of the HTTP designed for constrained environments to assure efficient communication at the application level. The common approach to achieve security to the Internet communications is by the Transport Layer Security (TLS) protocol. However, TLS is not suitable for constrained networks as it needs a reliable messaging and high power consumption. Therefore, multiple alternatives have been proposed to provide the security requirements in constrained environments. The CoAP specification allows either the use of the DTLS protocol or Internet Protocol Security (IPsec) to achieve data integrity and authentication [2]. In this project, we will discuss the security of IoT using the Datagram TLS protocol over CoAP.

1.3 Outline

This report is organized as follows. In Chapter 1, we provide motivation and the project goal. In Chapter 2, an overview on the Internet of Things, the vision and concept of IoT, its features and requirements. Chapter 3 presents the Constrained Application Protocol (CoAP) providing its design and layers. Chapter 4 provides one of the methods of securing CoAP by deploying the Datagram Transport Layer Security DTLS protocol and explaining the handshake protocol and the method of securing the CoAP. Chapter 5 presents the simulation of CoAP and DTLS over CoAP (CoAPs) using Contiki-OS. The last chapter will presents the conclusion of this project.

Chapter 2

Overview of Internet of Things

2.1 IoT Overview

The general concept of the IoT is not new, as it is related to the internet and its applications, and to Information and Communications Technology (ICT), as well as many other technologies, for instance, smart devices, sensors, embedded systems, and wireless communications. The term Internet of Things was coined by Kevin Ashton in 1999 while working at Auto-ID Center at MIT [3]. The phrase IoT is used as an umbrella keyword that covers different aspects related to objects or things and the Internet. Many researchers and organizations have defined IoT from their point of view, and it is not easy to have a definition that everyone agrees on.

In 2001, members of the same MIT group used IoT concept and explained their vision of creating a smart world, defining IoT as: "an intelligent infrastructure linking objects, information and people through the computer network" in the context of identifying and tracking objects [4].

Later in 2005, the International Telecommunication Union (ITU) published a report about the IoT and the development of new technologies that enables new forms of communication between people and objects or things and between things themselves. The ITU explained its vision about IoT "a new dimension has been added to the world of information and communication technologies (ICTs): from anytime, anyplace connectivity for anyone, we will now have connectivity for anything. Connections will multiply and create an entirely new dynamic network of networks an Internet of Things". Connectivity will take on a new diminution "anytime, anywhere, by anyone and anything" as illustrated in Figure 2.1 [1].

The IoT definition regarding to Coordination and Support Action for Global RFID Related Activities and Standardization (CASAGRAS): "A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and

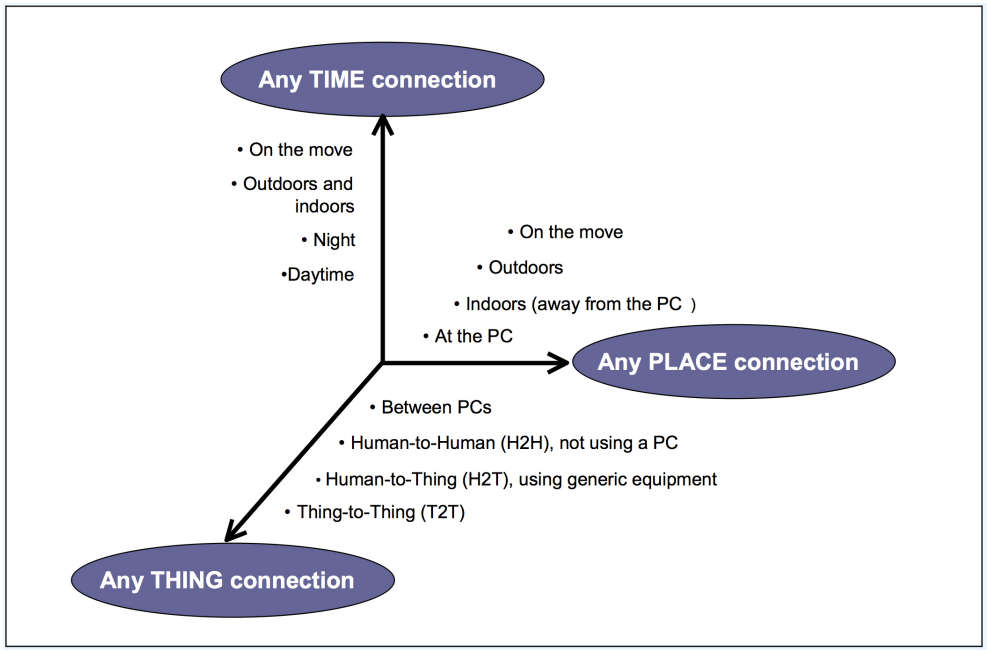

Figure 2.1: A new dimension to the telecommunication environment [1].

applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability" [5].

Another definition of IoT comes from the Cluster of European RFID Projects (CERP's): "a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities, use intelligent interfaces, and are seamlessly integrated into the information network" [6].

Based on the IoT concept, our vision of the IoT is a network infrastructure, that links physical objects or things through the Internet in order to make an action to the real world to reach common goals, and could be deployed in everything and will allow access from anyplace and anytime. The IoT is built on three main pillars related to the capability of objects which must have communication capability, computational capability and may have interaction capability:

(i) Communication capability

Objects in IoT must have a minimal set of communication capability. What we mean by this is not only a communication channel, but also everything related to it, in order to make an efficient communication, such as, an address, identifier, and

name. The objects may have all these features or some of them.

(ii) Computational capability

Objects must have some basic or complex computational capability, in order to process data and networks configurations. For instance, receive commands over the communications channel, manage network tasks, save the status from a sensor, activate an effector.

(iii) Interaction capability

The IoT technology may have an interaction capability in terms of sensing and actuating. This can be done either by, sensors and/or effectors (actuators). Sensors are things which sense or detecting the real world environment (e.g., light, humidity, temperature, movement, voice, etc.). Effectors are things that change or effect the real world such as, switches that allow you to trigger or to turn on/off anything that can change the real word such as motors, beepers, cameras, etc.

2.2 IoT vision and concept

From the above-mentioned definitions, there are some differences and different understanding of the IoT due to researchers background and organizations research interest. However, most of them agree on the concept of the IoT, which is to have a smart world by linking everything around us with the internet infrastructures to reach common goals. With the phenomenon increase of technologies nowadays we can see the shift from typical internet usage such as surfing the web and sending emails, to interconnecting physical objects that communicate with humans and with each other in order to offer a particular service.

The Internet of Things will create big businesses and non-businesses opportunities to users, companies, manufacturers, and governments. Besides, various IoT applications will be deployed in our lifestyle, for instance, health-care, home management, environmental monitoring, security and surveillance while ensuring the privacy of information and protection of exchanged contents. All these will bring tremendous benefits to the society, public and privet sectors, as well as increase jobs opportunities in the future.

2.3 IoT features and requirements

Since IoT will deal with different applications, devices and has a number of ICT fields, here are some of its significant characteristics that IoT need to support: [7] [8]

- Heterogeneity

  One of the most important features of IoT is to deal and manage such a high level of devices heterogeneity, which is expected to have different communication and computational capabilities.

- Scalability

  The IoT scalability refers to the ability to add new objects or devices, and services to the IoT system without affecting the quality of existing services. As thousands of objects are connected to the Internet every day, scalability issue is becoming bigger and more challenging. Such a huge number of new devices and services with heterogeneous resources will effect IoT scalability at communication level, naming and addressing level, managing and mapping services.

- Self-organizing capabilities

  IoT is expected to have self-organizing capabilities in order to minimize human intervention, including a configuration autonomy, self-organization, and adaptation to different scenarios, self-reaction, and processing of exchanged data.

- Energy-efficient solution

  Reducing the use of energy is one of the important technique for IoT applications. Since most of IoT objects will use wireless communication technologies, finding an efficient and reliable energy solutions is extremely important. The solution must be deployed in all IoT levels from designing level, to processing and communication level.

- Security and privacy

  Due to a high amount of data exchanged and the use of the Internet in all IoT applications, it is important to ensure the privacy of all the information and data. Securing devices and communications as well should be considered as a significant matter in order for this technology to be used everywhere and anytime.

2.4 IoT architecture

The standardization of IoT architecture is still an open issue. IEEE has started a new standard development project called IEEE P2413. This standard defines an IoT architectural framework, including several IoT domains and the relationship and communication between those domains. The work in this project is still in progress [9].

Many architecture models have been proposed from different organizations and researchers. Figure 2.2 provides the main and general model which consists of three-layer architecture perception layer, network layer, and application layer [10] [11]. The definitions of these layers are defined below:

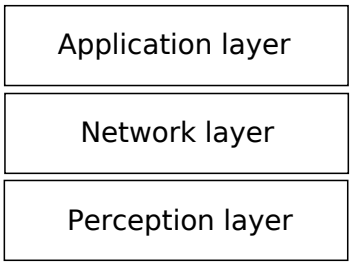

Figure 2.2: IoT architecture

1. Perception layer: the perception layer can also be called physical layer. This layer consists of physical objects or devices such as sensors, RFID system, meters, GPS system. This layer basically collect identification or information depends on the type of the sensor or the physical device.

2. Network layer: the network layer is also known as transmission layer. This layer is responsible for interconnection and communication functions. The transmission can be done through wired or wireless communication technologies such as Wi-Fi, Bluetooth, 3G, ZigBee. Many transmission and security protocols can be deployed in this layer such as: IPV4, IPV6, DTLS, IPsec. This layer should securely transfer the data from the physical layer to application layer.

3. Application layer: the application layer provides and manages application services for users needs. Many application protocols can be deployed in this layer such as CoAP, and HTTP depends on the type of application and the IoT devices.

Chapter 3

The Constrained Application Protocol

3.1 CoAP Overview

The Constrained Application Protocol (CoAP) is a web transfer protocol at the application layer intended to be used with constrained devices (e.g., low-power node, sensors, switches or actuators) and constrained (e.g., low-power, lossy) networks. The Internet Engineering Task Force (IETF) working group has designed this protocol to be used for M2M applications, IoT objects and suitable for constrained devices that have limited amount of ROM and RAM.

One design goal of CoAP is limiting the need for fragmentation by using small message overhead. Moreover, this protocol suitable for constrained networks such as 6LoWPAN which supports the fragmentation of IPv6 packets into small frames.

CoAP provides an interaction model similar to the client/server of HTTP. Also, M2M interactions result in implementing CoAP in both client and server rules. However, it is designed to interface easily with HTTP for integration with the web and ensuring other requirements such as low overhead, multicast support, and simplicity for constrained environments.

The main features of CoAP: [2]

- Constrained web protocol fulfilling M2M requirements.
- Asynchronous message exchanges.
- UDP binding with optional reliability supporting unicast and multicast requests.
- Low header overhead and parsing complexity.
- Simple proxy and caching capabilities.
- URI and Content-type support.
- A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP.

– Security binding to Datagram Transport Layer Security (DTLS)

Generally, CoAP architecture could be described as a two-layer approach: a message layer and a request/response layer shown in Figure 3.1. The lower layer deals with UDP and asynchronous technique, while the upper one manages the mapping between the requests and responses using method and Response Codes. The next two sections will present both layers.

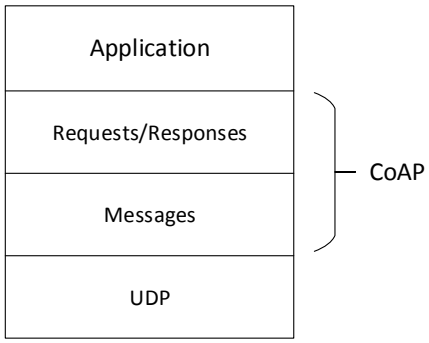

Figure 3.1: CoAP protocol layers

3.2 CoAP Messaging model

The messaging model of CoAP is based on the exchange of compact messages over User Datagram Protocol (UDP) between endpoints. Messages are transported, by default, over UDP. However, CoAP can also be used over DTLS and other transports such as TCP, SMS or SCTP. Messages are shared by requests and responses. Each message has a message ID used to detect duplicated messages and for optional reliability.

CoAP defines four types of messages:

– Confirmable (CON): the confirmable message requires an acknowledgement response from the receiver in order to provide a reliability functionality.

– Non-Confirmable (NON): the non-confirmable message does not require an acknowledgement from the receiver.

– Acknowledgement (ACK): acknowledges the confirmable message.

– Reset (RST): the reset is used instead of ACK in case that CON or NON cannot be processed.

Reliability functionality can be provided by CoAP by marking a message as Confirmable (CON). The idea is that a CON message is retransmitted using a timeout until the recipient sends ACK messages with the same message ID from the corresponding endpoint. Figure 3.2 illustrates a reliable message transmutation with a message ID example of 0x01d3. When a recipient is not able to process a CON message, it then replies with an RST message instead of an ACK message.

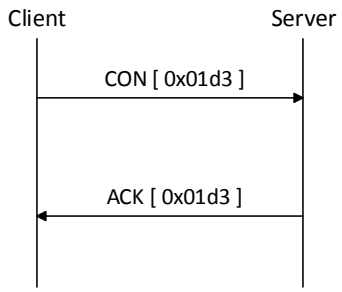

Figure 3.2: Reliable Message Transmission

Unreliable messages can be sent with NON type message. These do not need to be acknowledged, but have to contain message ID for detection in case of retransmission. When a recipient fails to process NON message, server replies with RST. Figure 3.3 shows unreliable message transmission.

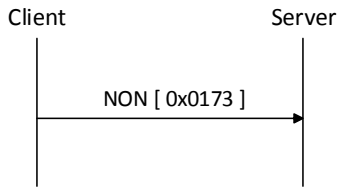

Figure 3.3: Unreliable Message Transmission

3.3 Message format

CoAP messages are encoded in a binary format. The message format is shown in Figure 3.4.

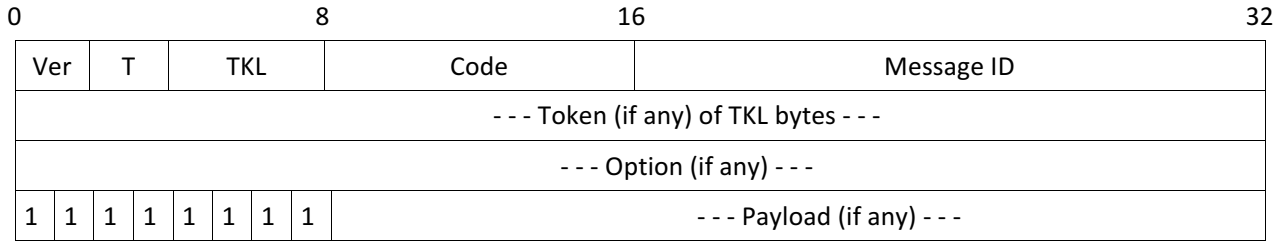

Figure 3.4: CoAP Message format

The message format starts with a fixed-size 4 bytes header that contains five fields and defined as follows:

- Version (Ver): 2-bit unsigned integer. Indicates the protocol version number.
- Type (T): 2-bit unsigned integer. Indicates the message type CON (0), NON (1), ACK (2), RST (3).
- Token Length (TKL): 4-bit unsigned integer. Indicates the number of bytes of the Token field.
- Code: 8-bit unsigned integer. Indicates the status of a client and server request and response.
- Message ID: 16-bit unsigned integer. Contains the Message ID that used to detect message duplication as well as to match ACK/RST to CON/NON message types.

The header is followed by a variable-length Token value, used to match or correlate requests and responses, which could be 0 to 8 bytes long, as given by the Token Length field in the header.

Header and Token fields are followed by zero or more CoAP options. An option may be followed by the end of the message or another option. Following the options, if any, the optional payload take place. The payload is prefixed by a one-byte payload marker (0xFF). The absence of this marker denotes a zero-length payload.

3.4 CoAP Requests and Responses

The CoAP interaction model is acting in both client and server similar to the HTTP client/server model as shown in Figure 3.5. CoAP requests and responses semantics are carried in CoAP messages that include either a Method code or response code. The CoAP requests are sent by a client to request an action on a resource on a server. The server then sends back a CoAP response with a response code.

Figure 3.5: Client-Server model

3.4.1 Request Method Definitions

The client requests an action using a Method code on a resource which identified by Uniform Resource Identifier (URI) on a server. The CoAP defines four different method codes:

GET: retrieves information from a specified resource identified by the requested URI.

POST: Submits information to be processed to a specified resource. The output result depends on the target resource, usually, results in the target resource being created or updated.

PUT: requests that the resource identified by the URI be created or updated with the carried information representation.

DELET: requests that the identified resource be deleted.

3.4.2 Response Code Definitions

The server sends back to the client a response code indicates the outcome of the request process. There are three classes of Response code:

– Success 2.xx : This class indicates that the request has been successfully received and processed. Here are some examples of response codes under this class: 2.01 Created, 2.02 Deleted.

– Client Error 4.xx : This class indicates that the request from the client was not valid or has an error. For example, 4.00 for a Bad request or 4.04 Not found this response code is like the common HTTP 404 which indicate that the request is correct but the server could not find the resource identified by the URI.

– Server Error 5.xx : This class indicates that the server has an error or incapable of processing the request. For example, 5.03 Service unavailable.

A request is carried in a Confirmable or Non-confirmable message, the response to the request has different scenarios:

1. Piggy-backed: when a request arrives at a server and if immediately available, the response is carried in the resulting ACK message. Therefore, there is no need for separate the acknowledgment and the piggy-backed response in order to save the network resources. An example of piggy-backed response is shown in figure 3.6.

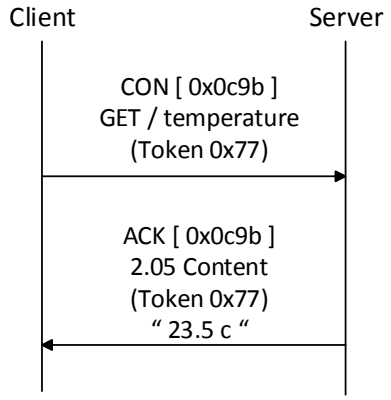

Figure 3.6: A GET Request with a Piggy-Backed Response

2. Separate response: when a CON request arrives and the server is not able to response immediately to the request, it responses with an empty ACK. Thus, the client can stop retransmitting the same request. An example of separate response is provided in Figure 3.7.

3. Non-confirmable request and response: When a client sends a Non-confirmable request, then the server response with a new Non-confirmable message. This type of exchange is shown in Figure 3.8.

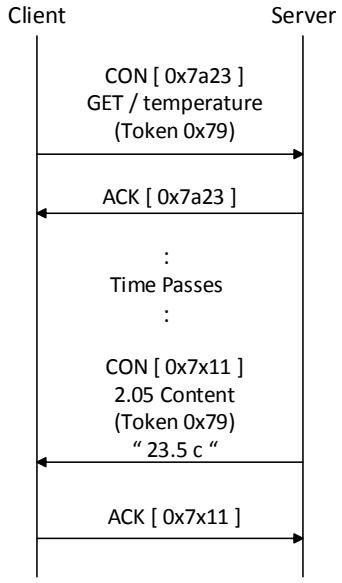

Figure 3.7: A GET Request with a Separate Response

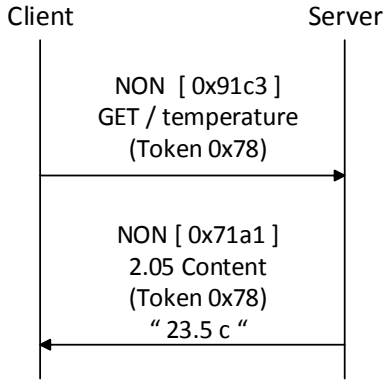

Figure 3.8: A request and a Response Carried in NON-confirmable Message

Chapter 4

DTLS-Secured CoAP

4.1 DTLS Overview

The Datagram Transport Layer Security (DTLS) is a protocol for securing network traffic, defined by IETF. DTLS is similar to the Transport Layer Security (TLS) with modifications to be used in datagram environments and to deal with unreliable transport protocols, e.g., UDP. DTLS was designed to be as close to the TLS as possible. The main changes due to lossy message transfer are: a) handling message loss, b) message reordering c) message size. As the TLS being used for securing HTTP, the DTLS is used for securing CoAP. The DTLS guarantees end-to-end security of different applications by operating between the application and the transport layers. It consists of two layers, the upper layer contains four subprotocols: Handshake, Change Cipher Spec, Alert, and Application protocols. The upper layer interacts with the lower layer which contains Record Protocol [12][13].

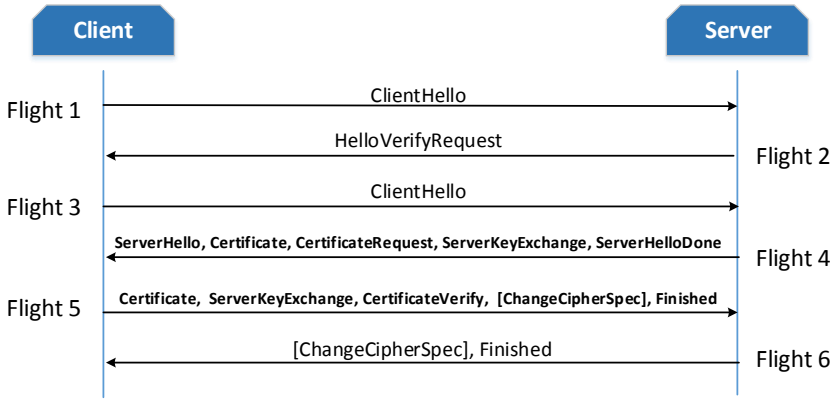

Figure 4.1: DTLS Handshake Process

- The Handshake Protocol

  The Handshake protocol is a series of exchanged messages between a client and a server before any data transmission take place. This protocol is responsible for the authentication and negotiation of security parameters to establish or resume secure

sessions. The TLS Handshake messages must be transmitted in a defined order. Therefore, this is incompatible with message loss and reordering. Whereas, DTLS can be deployed over unreliable transport layer protocols. The DTLS should then provide reliability to Handshake messages. This is done by adding two mechanisms: a retransmission timers to handle message loss, and a message sequence number within a handshake. This specific sequence number allows the receiver to quickly determine the correct order of handshake messages. DTLS messages are grouped in flights; each flight may contain a number of messages. Figure 4.1 illustrates the DTLS handshake process.

- The Change Cipher Spec Protocol

  The Change Cipher Spec on the upper layer of DTLS is used during the handshake process to indicate that all subsequent messages should be protected using the just-negotiated cipher spec and keys.

- The Alert Protocol

  The Alert protocol is used to carry information about errors or warnings messages between two peers. The Application protocol refers to the protocol on the application layer, e.g., CoAP.

- The DTLS Record Protocol

  The DTLS Record Protocol used to handle data transport and to apply security mechanisms. It protects application data by using security parameters and keys generated during the handshake. The outgoing messages are fragmented, compressed, and encrypted on this record layer and vice versa for the incoming messages.

4.2 DTLS for CoAP security

DTLS is used over CoAP to provide end-to-end security. Just as TLS being used for securing HTTP over TCP, Datagram TLS (DTLS) is used for securing CoAP over UDP. DTLS is implemented between transport layer and application layer as in Figure 4.2.

As DTLS operated on top of the UDP protocol, the complexity of its implementation increased which requires mechanisms to provide reliability. To design a DTLS version that can be used in constrained environments, it is important to minimize the

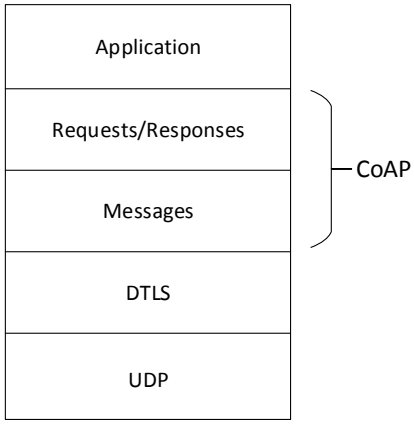

Figure 4.2: Abstract Layering of DTLS-Secured CoAP

code size, and the number of messages exchanged to get an optimized handshake protocol. As suggested in [14], where DTLS can be used in conjunct with CoAP to benefit from the block-wise transfer feature by CoAP, designed to support the transmission of large payloads which can be used to transport DTLS handshake messages. Figure 4.3 illustrates DTLS handshake protocol over CoAP with an example of a session ID (/session/1234abcd) to identify DTLS handshake session.

Figure 4.3 shows how the DTLS handshake works using CoAP, providing communication reliability by CON and ACK messages using CoAP block-wise transfer which contain DTLS handshake messages as a payload. When the DTLS handshake session has finished, the client can initiate the first CoAP request [2] [14].

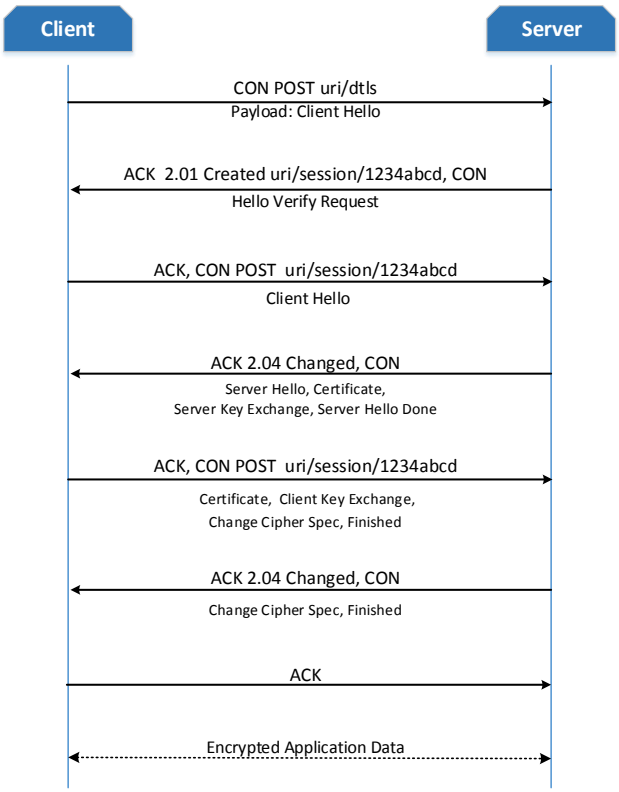

Figure 4.3: CoAP protocol layers

Chapter 5

Simulation and experimental results

The main goal of our simulation is to simulate the CoAP protocol without security and then simulating the DTLS protocol over CoAP (CoAPs). Providing a comparison between the two experiments in terms of memory footprint and power consumption since these two factors are the most concern factors in constrained devices. We simulate our experiment on Contiki-OS [15]. Contiki-OS is an open source operating system for IoT; it connects tiny low-power, low-cost microcontrollers to the Internet. Contiki-OS provides low-consumption Internet communication and supports many low-power wireless standards. We used Cooja simulator provided by Contiki. It is a network simulator where developers can test their applications on fully emulated devices before running it on real hardware. First, we simulate the Constrained Application Protocol CoAP without DTLS and then we simulate DTLS over CoAP (CoAPs) as shown in the following two sections.

5.1 CoAP Simulation

The CoAP implementation in Contiki-OS is based on Erbium (Er) [16]. Er is a low-power REST Engine for Contiki and become the official one for the operating system. The sensor mote we used for this simulation is a Wismote wireless sensor which is equipped with TI MSP430 low-power microcontroller, 16 KB RAM and 128 KB flash Memory [17]. The first scenario we implement two motes in Cooja, a CoAP server and CoAP client, we observe the motes output within Cooja.

The second scenario we implement a CoAP server and a Border Router mote. In this simulation, we used Copper (Cu) instead of CoAP client to retrieve all servers resources. Copper is a general browser for the Internet of Things (IoT) based on the Constrained Application Protocol (CoAP). It is a user-friendly management tool used for networked embedded devices. By integrating it into web browsers, which allows an intuitive interaction to debug CoAP devices. Figure 5.1 illustrate Copper interface in the web browser [18].

We can interact with CoAP server after entering the server URL and the CoAP port (5683), and then, discover the available resources of the CoAP server by using the

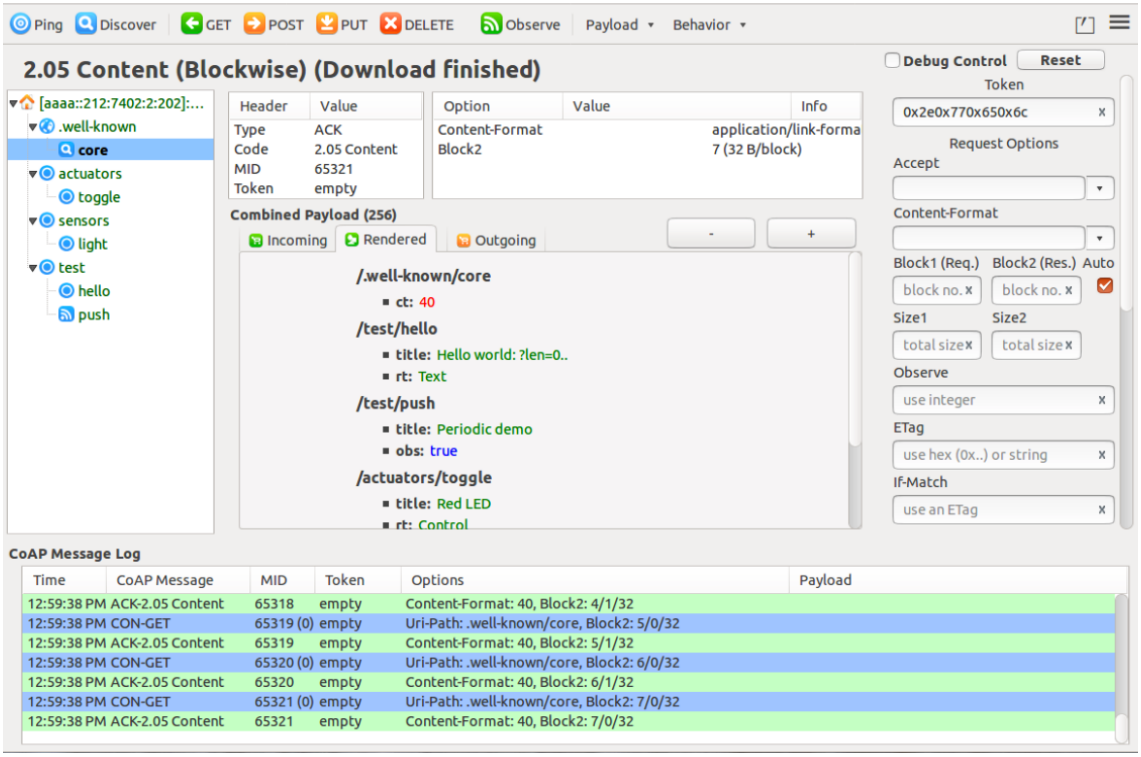

Figure 5.1: Retrieving CoAP's resources from Copper's interface

Discover button.

We can use the buttons GET, POST, PUT and DELET to interact with the server, these CoAP methods has different usage, GET: Retrieve a resource, POST: Create a resource, PUT: Update a resource and DELETE: Delete a resource. The REST engine calls each resource at the server by a handler function to serve the client request. The responses from the server will be displayed in the browser.

5.2 DTLS over CoAP (CoAPs)

DTLS protocol can be integrated with CoAP to provide end-to-end security. In this implementation, we used the open source TinyDTLS and CoAP libraries [19]. TinyDTLS supports the cipher suite based on Pre-shared keys (PSK) with the Advanced Encryption Standard (AES): TLS_PSK_WITH_AES_128_CCM_8. We implement two Wismote nodes in Cooja, a CoAP server and CoAP client, with an integration of TinyDTLS for both server and client. We observe the motes output within Cooja and compare the simulation result with the previous CoAP simulation.

The general interactions of data flow between DTLS and CoAP in both forward and

reverse directions are illustrated in figure 5.2 and figure 5.3 respectively. In the forward direction, CoAP packets are sent to DTLS module to add the security functionality. There are two interfaces in this operation: DTLS receives normal data packets from CoAP and then sends encrypted data to CoAP. Afterward, the encrypted packets are sent across to UDP as shown in Figure 5.2.

In the reverse direction, the secured packets received from UDP are sent across to DTLS for decryption then sending it back to CoAP as shown in Figure 5.3.

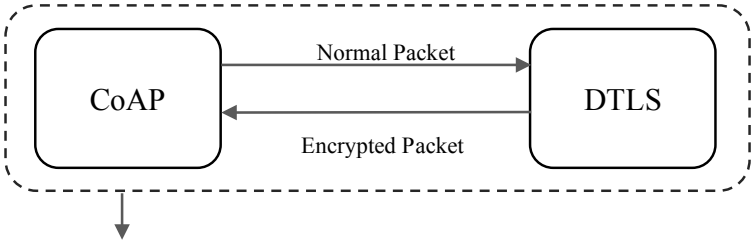

Figure 5.2: Encrypting a CoAP packet using DTLS

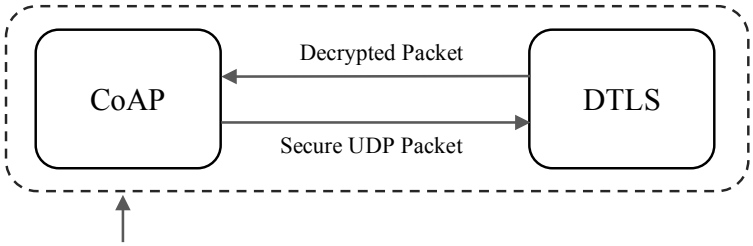

Figure 5.3: Sending a DTLS decrypted to CoAP

5.3 Experiment Results

IoT sensor devices have limited memory, CPU and power resources. In our experiment, we test the Constrained Application Protocol with and without security in constrained sensor nodes. In order to know the impact of deploying security mechanisms on constrained devices, we test and compare the memory footprint as well as the power consumption as shown below:

(a) Memory Footprint

The Internet Engineering Task Force group (IETF) classified constrained devices with consideration of code size and data size as shown in Table 5.1. Therefore, it

is important to know the code size and the data size for an application to measure the memory usage. The difficulty of providing a secure communication increase with limited resources. The classification of constrained devices is shown below:

Class 0 devices are very constrained nodes. They have limited memory and processing capabilities and may not have enough resources to communicate securely and directly to the Internet. Thus, they need a help of larger devices such as a proxy or gateway to participate in Internet communications. **Class 1** devices are quite constrained nodes. They cannot easily use full protocol stack such as HTTP or TLS. However, they are capable to use protocols that designed for constrained nodes such as CoAP over UDP. In our implementation, we use Wismote node which has 16 KB of RAM and 128 KB ROM and considered as a class 1 device. **Class 2** devices can support most protocols used in Internet communication [20].

Table 5.1: Classes of Constrained Devices

Name	Data size (e.g., RAM)	Code size (e.g., Flash)
Class 0	<< 10 KB	<< 100 KB
Class 1	~ 10 KB	~ 100 KB
Class 2	~ 50 KB	~ 250 KB

The memory footprint is provided by the MSP430-GCC compiler. We obtained the RAM and ROM by utilizing the MSP430-GCC size command of the firmware files. Table 5.2 shows the require memory size for both server and client with and without security implementation for Wismote sensor. Figures 5.4 and 5.5 show the impact of deploying security for CoAP. The difference of the RAM size between the two codes is about 30%. Whereas, the flash memory or ROM has increased by approximately 59%.

Table 5.2: Memory Footprint

Node type	RAM [bytes]	ROM [bytes]
CoAP Client	8210	44831
CoAP Server	8200	50224
CoAPs Client	11396	86583
CoAPs Server	11274	89737

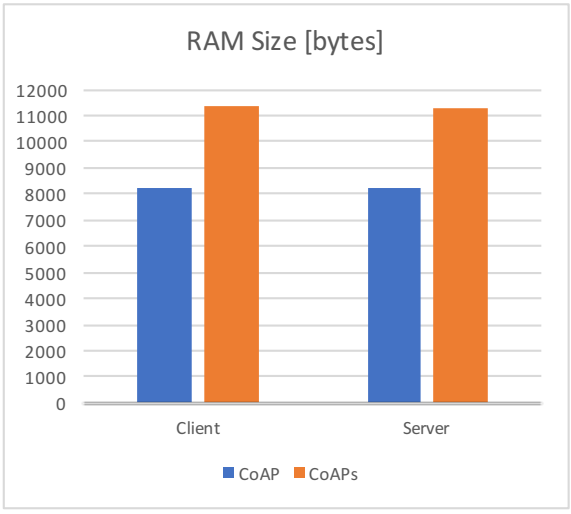

Figure 5.4: RAM Footprint

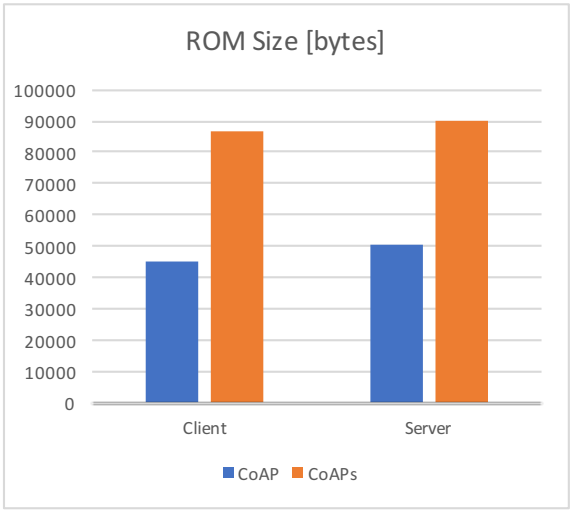

Figure 5.5: ROM Footprint

(b) Power Consumption

Power consumption of an IoT or constrained devices is a critical concern since most of the constrained devices have limited power resources. In our simulation, we use Powertrace function, which uses the CPU time measured in clock ticks unit, which is the time when the device used the CPU. This function has 94% accuracy compared with real measuring instrument[21].

Table 5.3: Data from Powertrace for CoAP

ALL CPU	ALL LPM	ALL TX	ALL RX
4878	1305479	68	1278364
6218	1631725	68	1606044
7560	1957968	68	1933724
9003	2284112	136	2261336
10355	2610345	136	2589016
11718	2936569	136	2916696
13079	3262795	136	3244376
14440	3589020	136	3572056

Powertrace provides the usage time of ALL_CPU (the total number of ticks for CPU in active mode), ALL_LPM (the total number of ticks in Low Power Mode), ALL_TX (Transmit) and ALL_RX (Receive).

Table 5.3 shows the printed values in four states of CoAP server node in the form of the number of clock ticks. The values from Powertrace were printed every 10 seconds of the simulation. The time value for each component can be converted to power with the following equation [21] [22]:

$$Power = \frac{V \times \sum_{i=1}^{4} I_i \times Ticks_i}{Ticks\ per\ Second \times Runtime}$$

Where Ticks mean the difference between the number of clock ticks in two-time intervals (e.g., in Table 5.3, to fined the Ticks value of ALL_CPU take the difference between 6218 and 4878). We got the values of the current for each component for CPU, LPM, TX and TR from Wismote datasheet which are 0.312mA, 0.026mA 33.6mA 18.5mA respectively. The voltage for this mote is 3V. The Ticks per Second is the value of the hardware timer resolution of Wismote 32768 Hz [17]. The Runtime is the time interval of Powertrace in which we perform measurements in every 10 seconds.

Table 5.4 provides the power consumption for each component and the average power consumption of CoAP without security which is around 55mW. We did the same measurements for CoAPs and the average power consumption is around 61mW. It is notable that the power consumption after deploying the security has increased by around 10 %.

Table 5.4: Power consumption for CoAP (mW)

CPU	LPM	TX	RX	Total
0.0038276	0.0077659	0.0000000	55.5000000	55.5115935
0.0038333	0.0077658	0.0000000	55.5000000	55.5115991
0.0041219	0.0077634	0.0209180	55.4884827	55.5212859
0.0038619	0.0077656	0.0000000	55.5000000	55.5116275
0.0038933	0.0077653	0.0000000	55.5000000	55.5116587
0.0038876	0.0077654	0.0000000	55.5000000	55.5116530
0.0038876	0.0077654	0.0000000	55.5000000	55.5116530
				55.5125678

From this simulation, the memory footprint showing a significant increase after deploying DTLS over CoAP. Where the RAM increased by 30 % and the ROM increased by 59 %. Therefore, the code need to be optimized to improve the memory footprint as well as the power consumption.

Chapter 6

Conclusions

The Internet of Things is considered as one of the largest improvement to the existing technology nowadays. It is extremely important to have a secure IoT system in order to develop and improve this technology to be used in a large scale.

In this report, we address the fundamentals of the Internet of things and the key features and requirements. Followed by the the Constrained Application Protocol (CoAP) as it will be a significant part of IoT. We present an overview of the Datagram Transport Layer Protocol (DTLS) which is one way of providing an end-to-end security for IoT applications.

To study the effect of added functionality to original CoAP protocol, two attributes were chosen, the memory footprint and power consumption. Simulation results show an increase in both attributes with a significant increase in the memory footprint almost doubled the size of the required memory. Thus, it is important to optimize the current implementation of this protocol to have an efficient and reliable IoT devices with better power consumption and memory footprint.

Bibliography

[1] Internal Telecommunication Union, "The internet of things?executive summary," *ITU Internet Reports*, 2005.

[2] Zach Shelby, Klaus Hartke, and Carsten Bormann, "The constrained application protocol (coap)," 2014.

[3] Kevin Ashton, "That internet of things thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[4] David L Brock, "The electronic product code (epc)," *Auto-ID Center White Paper MIT-AUTOID-WH-002*, 2001.

[5] RFID CASAGRAS, "the inclusive model for the internet of things report," *EU Project*, , no. 216803, pp. 16–23, 2011.

[6] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, M Eisenhauer, et al., "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, pp. 9–52, 2011.

[7] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[8] Eleonora Borgia, "The internet of things vision: Key features, applications and open issues," *Computer Communications*, vol. 54, pp. 1–31, 2014.

[9] IEEE Standards Association et al., "P2413-standard for an architectural framework for the internet of things (iot)," .

[10] Zhonggui Ma, Xinsheng Shang, Xinxi Fu, and Feng Luo, "The architecture and key technologies of internet of things in logistics," pp. 464–468, 2013.

[11] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan, "Future internet: the internet of things architecture, possible applications and key challenges," in *Frontiers of Information Technology (FIT), 2012 10th International Conference on*. IEEE, 2012, pp. 257–260.

[12] Eric Rescorla and Nagendra Modadugu, "Datagram transport layer security version 1.2," 2012.

[13] Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, and Thiemo Voigt, "Lithe: Lightweight secure coap for the internet of things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, 2013.

[14] Sandeep Kumar, Zach Shelby, and Sye Keoh, "Profiling of dtls for coap-based iot applications," 2013.

[15] Adam Dunkels, Oliver Schmidt, Niclas Finne, Joakim Eriksson, Fredrik Österlind, Nicolas Tsiftes, and Mathilde Durvy, "The contiki os: The operating system for the internet of things," *Online], at http://www. contikios. org*, 2011.

[16] Matthias Kovatsch, Simon Duquennoy, and Adam Dunkels, "A low-power coap for contiki," in *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011)*, Valencia, Spain, Oct. 2011.

[17] "Wismote specifications," *Web page: http://www.wismote.org/doku.php?id=specification*, 2017.

[18] Matthias Kovatsch, "Demo abstract: Humancoap interaction with copper," in *Proceedings of the 7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2011)*, Barcelona, Spain, June 2011.

[19] Olaf Bergmann, "Tinydtls," *Web page: http://tinydtls.sourceforge.net*, pp. 02–15, 2017.

[20] Carsten Bormann, Mehmet Ersue, and A Keranen, "Terminology for constrained-node networks," Tech. Rep., 2014.

[21] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes, "Powertrace: Network-level power profiling for low-power wireless networks," 2011.

[22] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors*. ACM, 2007, pp. 28–32.