CHALMERS

Enabling Multicast IPsec for Internet of Things

Master's Thesis in Communication Engineering

ARGYRO LAMPROUDI

SICS, Swedish ICT AB
Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2017
Master's Thesis 2017

RI.
SE

Abstract

The efforts of making our life simpler and easier, with sustainability and environment protection being our main drives, brought about the so called Internet of things (IoT). One of the biggest challenges in IoT, is secure and seamless interconnection of devices with limited and non-limited capabilities over the Internet, while utilizing minimum of resources. Group communication has been proven to be more efficient in terms of resource utilization compared with pairwise communication. However, it does not fulfil the security requirements in the network layer of IoT as yet. This thesis provides a solution that contributes towards security of group communication in the network layer of IoT, using the multicast extension to the security architecture for the Internet protocol (IP) and a group key management protocol that enables dynamic establishment of group security associations and keys, called Group-IKEv2 for IoT.

This thesis work presents the design and implementation of Multicast IPsec with Group-IKEv2 for IoT, discussing vital components that should be taken into consideration during design and implementation phases. Experimental results that indicate the performance and security limitations of the considered system are presented as well.

Acknowledgements

First of all, I would like to thank my supervisor at SICS Swedish ICT, Shahid Raza, for trusting me to take over this project and for his overall assistance. I am also grateful to Marco Tiloca for the interesting discussions and his guidance. Moreover, I would like to thank my examiner at Chalmers University of Technology, Tommy Svensson for the fresh ideas, the motivation and the support he provided thoughout the duration of this project. I would also like to thank my professors at Chalmers for widening my perspective of approaching problems. Last but not least I would like to express my gratitude to my father, my mother, my brother, and my friends for believing in me and for supporting me all this time during my master.

Argyro Lamproudi, Stockholm 20/02/2017

Contents

Acronyms

6LoWPAN IPv6 over low power wireless personal area networks. 7

AH Authentication header. 2, 11

ASK Amplitude shift keying. 7

BPSK Binary phase-shift keying. 7

CoAP Constrained application protocol. 3

CSMA/CA Carrier sense multiple access collision avoidance. 7

CSS Chirp spread spectrum. 7

DAO DODAG destination advertisement object. 8

DDoS Distributed denial of service. 3, 48, 79

DIO DODAG information object. 8

DIS DODAG information solicitation. 8

DODAG Destination-oriented directed acyclic graphs. 8

DTLS Datagram transport layer security. 3

ESP Encapsulating security payload. 2, 11, 51

FCFS First come first serve. 83

GC/KS Group controller/Key server. 3, 13, 15, 20, 25, 61

GFSK Gaussian frequency-shift keying. 7

GPAD Group peer authorization database. 21

GSA Group security association. 4, 12, 16, 20, 62

GSAD Group security association database. 21, 51

GSAK GSAs for encrypting Keys. 24, 64

GSAT GSAs for encrypting Traffic. 24, 64

GSPD Group security policy database. 21, 51

IDS Intrusion detection system. 47, 80, 82

IKEv2 Internet Key Exchange version 2. 2, 12

IoT Internet of things. 1, 2, 6, 15

IP Internet protocol. 2

IPsec IP security. 2, 11

ISP Internet service provider. 6

KEK Keys encrypting Key. 13, 17, 25, 73

LKH Logical key hierarchy. 45, 48, 84

MAC Medium access control. 7, 50

MPSK M-ary phase-shift keying. 7

MTU Maximum transmission unit. 7

OFT One-way function tree. 17, 45, 84

OS Operating system. 50

PAD Peer authorization database. 12, 21

PSK Pre-shared Key. 71

QPSK Quadrature phase-shift keying. 7

SA Security association. 12

SAD Security association database. 12, 21

SID Sender ID. 31

SPD Security policy database. 12, 21

SPI Security parameter index. 12

TEK Traffic encrypting Key. 13, 25, 73

TLS Transport layer security. 3

UDP User datagram protocol. 8

UWB Burst position modulation combined with BPSK. 7

1

Introduction

The vision of IoT has pictured a fully connected world, in which any object or any type of device like a smart phone, a tablet, a personal computer, a sensor, a smart meter, an auto-mobile part and even a toothbrush, can be connected to the Internet, constituting a hybrid network of devices.

IoT technologies are applicable in a vast variety of fields, such as in health and wellness, in agriculture, in the industry, at home, and in infrastructure. Physical activity monitoring in aged people can now be enabled by interconnecting body sensors that measure motion and vital signs by a smart phone. The generated data are collected and analysed by the smart phone in order to evaluate the activity level. Dental health activity can also be monitored with the interconnection of our toothbrush with our smart phone using Bluetooth, so that records of our everyday dental habits can be evaluated later on by our dentist. Applications in the smart home domain, such as safety monitoring can be enabled with the interconnection of cameras and movement sensors with a remote control system. Gas, water and energy monitoring can also be enabled using smart meters connected with a control entity which the user can monitor, either locally or remotely using a smart phone. IoT applications in the smart agriculture can optimize electricity, watering and the amount of fertilizer, using sensors that communicate with a remote control system [1]. All the applications ultimately aim to make everyday life simpler and easier, prioritizing sustainability and environment protection.

Group communication has found significant resonance among IoT applications that require simultaneous one to many or many to many communication. Compared with pairwise communication, group communication can utilize more efficiently the device and network resources, which is highly desirable in a hybrid network of devices with limited and non-limited capabilities. Several applications of IoT require a network entity to remotely prompt a group of devices, such as sensors, smart meters, and actuators, to take an action or acquire data. For instance in a vehicle control management system, a remote management entity simultaneously requests from a group of on-board car sensors

to collect measurements and data that can verify the status of crucial auto-mobile parts, and determine if maintenance or replacement is needed. Multicast communication can also be used in cases that simultaneous remote control of devices is needed. In an indoor air quality application, sensors located in a building measure air quality, and send the acquired measurements to a fan controller, which simultaneously controls particular groups of the ventilation fans to improve the air quality. In a smart home IoT application, a user might desire to remotely switch on the water heater and the thermostat at the same time, or control simultaneously all the lights within the house, which can be achieved using group communication.

1.1 Background

IoT applications require different types of networks, consisting of devices with limited and non-limited capabilities, to be successfully interconnected. The network standard that enables interconnection of billion of devices due to its high address availability is IPv6. Low-rate personal area networks are able to use IPv6 with the help of 6LoWPAN standard. It compensates for the heavy headers of IPv6 using compression and fragmentation. 6LoWPAN networks are connected to the Internet through 6LoWPAN border routers, which are responsible for performing header compression/decompression and fragmentation/defragmentation to the outgoing and incoming IPv6 traffic.

ITU-T has identified that one of the challenges in the IoT vision is to make sure that security and privacy requirements of IoT services are met, while at the same time efficient interconnection of services is achieved by using interoperable information and communication technologies, and by exploiting at most data capture, processing and communication capabilities. According to [1], without meeting security and privacy requirements, IoT proliferation rate will inevitably diminish. One of the most important aspects related to security in IoT is the secure interconnection of any type of device with the Internet. In the current research work, a lot of endeavours have been made to enable secure communication in the IoT domain through different layers of TCP/IP.

In the link layer, IEEE802.15.4 can ensure per hop-based security in a network, using a pre-shared key to encrypt and integrity protect the communication between peers, regardless the network protocol used, IP or non-IP [12]. Nonetheless, link layer security brings forward high exposure risk of the communications in the entire network even if only a single node is compromised by an adversary. In the network layer, IP security (IPsec) is the in-built security solution of IPv6, achieving seamless end-to-end secure communication between resource-constrained and non-resource-constrained devices with the minimum implementation cost. It is constituted by two protocols, Authentication header (AH) and Encapsulating security payload (ESP) [6]. IPsec protocols offer confidentiality, data integrity, authentication, and replay attack protection services to any application running on top of the network devices, regardless of the transport layer protocols being used. Both protocols use security associations to determine which security services and parameters are used to secure the communication. The security associations can either be set manually or negotiated between the peers using Internet Key Exchange

version 2 (IKEv2) protocol[5]. End-to-end secure communication between two peers, can also be achieved in the transport layer with Datagram transport layer security (DTLS) [10]. DTLS is an alternate solution to the Transport layer security (TLS), destined to operate with UDP datagrams, since the connection-oriented characteristics of TCP are not suitable for 6LoWPAN networks [14], due to the lossy nature of resource constrained devices. Reliable transmission of data is assured by protocols that are working on top of DTLS, like Constrained application protocol (CoAP) [11]. Among data integrity, authentication and confidentiality services, DTLS also enables Distributed denial of service (DDoS) attack detection using Cookies.

1.2 Motivation & Objectives

The aforementioned solutions are used to enable secure unicast communication between two peers. However, end-to-end security in multicast communication in the IoT domain, is more complex to achieve, since more security complications arise in terms of source authentication and trustworthiness of a member in a multicast group. Recently, an adaptation of the DTLS protocol that supports multicast has been proposed [3], enabling secure group communication over UDP datagrams. Nevertheless, there is no solution that enables end-to-end secure group communication in the network layer in IoT. Currently, IPsec protocols can establish secure group communication within a multicast group, only if the security associations and key material are manually configured to the network devices that are members of the multicast group. However, this method is not optimum to be adopted in the IoT domain, since many devices, such as smart meters and sensors, can be located in remote places, which makes it more difficult to manually reconfigure membership changes in the multicast group.

This thesis work proposes an extension to IKEv2 protocol, Group-IKEv2, and an extension to IP security architecture for IoT, multicast IPsec. Group-IKEv2 enables dynamic establishment of the security associations and key material in the multicast group which are then used by multicast IPsec utilizing efficiently the device and network resources. We have adopted a centralized approach of group key management, in which a central entity, called Group controller/Key server (GC/KS), is responsible for providing in a dynamic manner the Group Security Associations and key material to the authorized and authenticated network entities that belong to the multicast group. We have assigned the role of GC/KS to a resource-constrained device which is already part of the multicast group in the 6LoWPAN network. We have extended the IKEv2 communication scheme, creating a new protocol, Group-IKEv2, which uses messages and payloads proposed in the Internet Draft Group Key Management using IKEv2 [22]. In that way dynamic establishment of group security associations is achieved. Lastly, to enable the multicast capability in IP security architecture for IoT, we adopt the standardised extensions of the multicast IP security architecture proposed in [29]. Our proposed solution is implemented in C, for Openmote hardware platform using Contiki OS.

1.3 Scope

This thesis work provides a solution that enables end-to-end secure group communication in IoT in the network layer by extending the capabilities of IKEv2 to Group-IKEv2 and IP security architecture to Multicast IPsec. The proposed extensions overcome the shortcomings of a static configuration of the security parameters in the multicast group, by enabling dynamic establishment of group security associations in a 6LoWPAN network. We evaluate our proposed solution in terms of resources utilization in the resource-constrained devices and delineate its limitations. Additionally, we present the security requirements of IoT applications that are met using our proposed framework, and the vulnerabilities of our scheme.

1.4 Research Methodology

Our methodology is based on both analytical and experimental research. The problem that initialized this research is that group communication in IoT is not dynamically enabled and secured in the network layer. Therefore in order to enforce security, a new communication scheme should be designed for IoT. The research questions that motivated our work are:

1. How can the IP security architecture be extended to secure end-to-end group communication in the network layer in IoT?

2. Is it possible to achieve dynamic establishment of Group security association (GSA) and key material in a multicast group of resource-constrained devices with the use of a centralized Group Key Management architecture?

3. Is it possible for a resource-constrained device to play the role of GC/KS in a Group Key Management scheme?

The analytical research conducted within the purposes of this thesis and presented in Chapter 2 expanded our knowledge and enhanced our understanding of protocols that enable IoT in all the layers of the TCP/IP stack. Moreover, we reviewed the existing work on group key management [18],[7],[8],[19] and examined how it could be adapted to the resource-constrained nature of the IoT devices to sufficiently work within a 6LoWPAN network. Next we designed a communication protocol and a system architecture, by combining the vital characteristics of existing group key management protocols integrated with IKEv2, and by extending the IPsec architecture such that the desired requirements are met. The design process consists of the following steps:

- Identification of functionalities that need to be integrated with IKEv2 to enable distribution of group security associations and key material to the multicast group.

- Identification of functionalities that need to be included in IPsec for enabling incoming and outgoing multicast IPsec traffic.

- Design the required extensions in IPsec to support multicast traffic.
- Design of the Group Key Management protocol, Group-IKEv2, based on the message types and payloads of [22].

The design is implemented for the hardware platform Openmote-CC2538 using Contiki OS in C. The basic steps of the implementation phase were the following:

- Implementation of payloads for Group-IKEv2.
- Implementation of message types for Group-IKEv2.
- Integration of Group-IKEv2 with the existing communication scheme of IKEv2.
- Implementation of the extensions in IPsec to support multicast traffic.
- Adaptation of UDP-example of Contiki to use Group-IKEv2 and Multicast IPsec.
- Debugging, testing and evaluation of the implementation using Openmote-CC2538 platform.

After the implementation, experiments were conducted using the platform CC2538, measuring the memory and energy utilization, the message sizes and execution time. Finally, we examined the sufficiency of our solution by comparing the collected results with the requirements of our system in terms of performance and security.

1.5 Thesis Outline

Chapter 2 provides an overview of the protocols used in the IoT domain for enabling interconnection of any device with the Internet through different layers of the TCP/IP stack. The basic characteristics of the IP security architecture, IKEv2 protocol and Group Key Management with IKEv2 are also presented. The related work conducted for enabling group communication is provided in Chapter 3 motivating the contribution of this work. In Chapter 4 the architectures of Multicast IPsec and Group-IKEv2 are presented, delineating the limitations of the existing schemes to support multicast and introducing the necessary extensions to support multicast. Chapter 5 provides a thorough description of the implementation of the aforementioned architectures, explaining the overall logic and the features that were implemented, and the limitations that arose during this phase. Chapter 6 presents the results acquired for the performance evaluation along with the security analysis of our proposed solution. Lastly, Chapter 7 concludes the thesis, and Chapter 8 presents the future work.

2

Background

This chapter aims to familiarize the reader with the necessary technologies, protocols, and security requirements in the IoT domain that were used throughout this work in order to build our solution. Initially in Section 2.1 the concept of IoT is presented. We explain how it has evolved and how it is deployed and evolved nowadays. We introduce the technologies, which enable global connectivity of resource-constrained devices with the Internet, providing an insight of how communication is established in different layers of TCP/IP. Later on, applications of multicast communication in IoT are presented, emphasizing the necessity of group communication in the IoT domain in terms of performance. Section 2.2 presents the security requirements that need to be fulfilled and are necessary in group communication. Finally, in Section 2.3 we introduce the security protocols on which we based our solution, discussing their major functionalities and characteristics.

2.1 Technologies

2.1.1 Internet of Things

Nowadays, any object can have a virtual identity and can interconnect through the Internet with any host or device. An object can be an embedded device of limited memory and/or processing and communication capabilities. The aforementioned object operates within a low-power lossy network that is connected to the Internet, enabling applications such as remote control of home appliances, vehicle automation, environment, energy, and health monitoring. For instance, a sensor that belongs to a wireless sensors network can communicate with a remote host through the backbone of an Internet service provider (ISP) to monitor the services. Network adaptation and interoperability are empowered by using several protocols introduced in the following subsections.

2.1.2 IEEE802.15.4 : Low-Rate Wireless Personal Area Networks

IEEE802.15.4 is a standard used in wireless personal area networks. It defines the physical layer and the Medium access control (MAC) sublayer, so that connectivity among portable moving devices with power limitations and limited coverage is enabled. It is destined for devices operating at the 868–868.6, 902–928 and 2400–2483.5 MHz bands, with maximum achievable data rate at 250 kb/s, and minimum rate at 20 kb/s, which depends on the device needs and the system requirements. It can support various modulation types such as Quadrature phase-shift keying (QPSK), Binary phase-shift keying (BPSK), Amplitude shift keying (ASK), M-ary phase-shift keying (MPSK),Chirp spread spectrum (CSS), Burst position modulation combined with BPSK (UWB), and Gaussian frequency-shift keying (GFSK). The physical layer offers two types of services, the physical data service and the physical management service. The first is responsible for transmission and reception of data using the radio, channel selection, and activation and deactivation of the radio [12]. Physical management service provides access to the layer management functions and is responsible for maintenance of a database called physical personal area networks information base.

The MAC sublayer of IEEE802.15.4 is responsible for channel access with the radio, using Carrier sense multiple access collision avoidance (CSMA/CA) mechanism. It enhances link reliability between two devices by successful verification of the frame check sequence, and offers per hop base security services like confidentiality, integrity, and replay attack protection. If security is enabled, auxiliary security header is added at the end of the MAC frame header. It offers eight security levels, each with different set of security services enabled. The Maximum transmission unit (MTU) size of the frame in the physical layer is specified to be 127 bytes, out of which 25 bytes are MAC layer overhead [13],[14] with 102 bytes remaining for the higher layers. If security is enabled then the available MTU size for higher layers is even more reduced, ranging between 93 to 81 bytes, depending on the selected cryptographic suite. The available MTU size for higher layers makes evident the need for 6LoWPAN, which is able to compress IPv6 packets into smaller MTU sizes as it is described in detail in the following section.

2.1.3 6LoWPAN : IPv6 over Low power Wireless Personal Area Networks

The IPv6 over low power wireless personal area networks (6LoWPAN) standard allows IPv6 to be supported by low-rate WPAN, constituting an adaptation layer between the MAC sublayer of IEEE802.15.4 and IPv6 in the network layer. As it has been already mentioned in the previous section, the available MTU size after decapsulation of the IEEE802.15.4 frame is 81 bytes when security is enabled and 102 when it is disabled. Assuming that layer 2 security is disabled, only 54 bytes remain for transport and application layers since 48 bytes out of 102 bytes are IPv6 packet overhead [14]. 6LoWPAN tackles the MTU size limitation posed by the IEEE802.15.4 standard, by applying context aware header compression to IPv6 packets and to higher layer protocols. IPv6 header compression is achieved by omitting IPv6 header fields which are known by

all the nodes within the 6LoWPAN network, and adding an IPv6 header compression encoding of 2 bytes size. Next protocol headers like User datagram protocol (UDP) and IP extension header can be also compressed using next header compression encoding. The encoding added for this compression, is usually of 1 octet size indicating the next layer protocol within its first variable length bits. One of the supported compressions using the next header compression mechanism is the IPsec header compression proposed in [15][16], aiming to reduce the heavy payloads of the IPsec protocol. The next header compression encoding size is 8 bits, out of which 3 bits are used for IPv6 extension header ID. Six out of eight possible values of IPv6 extension header ID, are already used, making possible to specify if AH or ESP header compression is performed using the remaining 2 values (101, 110). As a result AH overhead is possible to be reduced from 24 to 16 bytes and ESP overhead from 30 to 24 bytes. 6LoWPAN also supports a fragmentation mechanism which enables IPsec packet encapsulation even if Link-Layer security is enabled in IEEE802.15.4. Each fragment includes an offset and a reassembly tag so that they can be easily reassembled in every communication end.

The functionalities of compression and decompression along with fragmentation are performed in particular entities of a 6LoWPAN network called, border routers, which are responsible for interconnecting a traditional IP Network with a Low power Wireless Area Network, so that seamless communication can be achieved.

2.1.4 RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks

RPL is a distance vector routing protocol for IPv6 low-power lossy networks, where the nodes have limited processing capabilities, memory and energy resources. It is compliant with any link layer technology requiring bidirectional links between the nodes. The protocol forms Destination-oriented directed acyclic graphs (DODAG) across the network towards a selected node, which is usually the 6LoWPAN border router, called DODAG root. An objective function defines the routing metrics and optimization objectives that are used to form a DODAG, and they can be node or link-based. For example a node-based metric can be the node's energy level, whereas a link based metric can be the link latency or the link reliability. The DODAG is formed by ranking the nodes with respect to the DODAG root, using the defined Objective Functions.

There are four new types of ICMPv6 control messages defined for this protocol, DODAG information solicitation (DIS), DODAG information object (DIO), DODAG destination advertisement object (DAO), and DAO acknowledgement [17]. Initially the DODAG root sends a DIO message to advertise the information of the graph. If the nodes receiving the DIO messages are configured as routers, they join the graph, select their rank and their parent depending on the defined objective function, and advertise their rank with a DIO message to their neighbours. If the receiving node is a leaf, then it simply joins the graph without sending any DIO message. This process is repeated until the complete DODAG to the root is formed. In that way all the nodes have a routing entry towards their parent. In order to enable downwards traffic directionality, all the nodes joining the graph send a DODAG Advertisement Object to their parent. As soon as a node receives this type of message, either the child's prefix is added to its

routing table if storing mode is supported, or sends its own DAO message including the prefixes it received from the child node. This process continues until the DAO message reaches the DODAG root. DODAG Information Solicitation message is sent by a node requesting DODAG Information Object from a RPL node. In that way the traffic is routed within a 6LoWPAN network.

2.1.5 Multicast in IoT

Multicast communication in the IoT domain is essential in applications that require simultaneous communication to a group of devices and efficient utilization of device and network resources. The energy consumption is minimized since with multicast communication, the device is transmitting the data simultaneously to all the members of the group, in contrast with unicast communication in which n transmissions are required for n number of members in the group. Some of the usage cases of multicast communication in IoT domain are the following:

1. A typical example of an IoT application that utilizes multicast communication is a light control system, where users are able to remotely control the lights, using an application in their smart phone. The lights and the switches of a house can be organized in groups depending on the location in the house. The application is able to send a multicast control request to the corresponding switches through the Internet to the 6LoWPAN network. The 6LoWPAN border router compresses incoming request and forwards it to the corresponding switches which in turn send a turn off/on request to the particular group of lights in order to turn them off/on.

2. A vehicle control management application can also use multicast communication. A 6LoWPAN network in the vehicle consists of smart meters, that are configured to determine the status of crucial auto-mobile parts. A remote host prompts a particular group of smart meters, by sending a multicast status request message through the Internet towards the 6LoWPAN network. The 6LoWPAN border router compresses and forwards the request to the smart-meters that are members of the group, in order to instruct them to acquire measurements so that the status of the auto-mobile parts can be determined. Then, the smart-meters forward the data to the host through the 6LoWPAN border router.

3. In emergency situations, such as in case of a fire in a building, a simultaneous transmission of emergency information to multiple ends is needed. In that case a 6LoWPAN network consisting of smoke detection sensors and automated sprinklers interconnects the building with remote monitoring systems of multiple ends of fire brigade departments. In case that a sensor detects smoke, it transmits using multicast information about the smoke levels and the location to the closest fire departments. Simultaneously, it stimulates particular group of sprinklers, depending on the location of the smoke in the building, to release water such that immediate action is taken to extinguish the fire [1].

4. In configuration or update of multiple devices with same functionalities, multicast communication is more efficient in terms of resource utilization. For instance, if the 6LoWPAN border router needs to provide to all nodes within the network an update regarding the latency to a particular host in the Internet, it is preferable to send this information with multicast, instead of transmitting them to all the nodes one by one [3].

A potential security breach in IoT applications, as the ones described earlier has inevitable consequences not only in the network interconnection, but also to the availability of the offered service. The criticality of data disclosure using group communication mandates the enhancement of security in using multicast.

2.2 Security Concepts

The fact that many types of networks, with various security risks are interconnected, necessitates the enhancement of security in the IoT domain. A potential security breach over applications running in a 6LoWPAN network has different level of impact compared to a security breach in a database which contains personal information. Regardless of the impact level of a potential security breach in a system, the following security requirements should be met in a system.

2.2.1 Confidentiality

One of the most crucial requirements of IoT is confidentiality. Data belonging to a particular party should not be revealed to an unauthorized individual. Only the owner of the data should be able to control to whom the data can be disclosed and by whom they can be changed. Loss of confidentiality in IoT can lead to violation of privacy and exposure of personal data [9]. One way to ensure confidentiality in IoT is using security protocols like IPsec and DTLS in which the transmitted data are encrypted using various cryptographic suites.

2.2.2 Data Integrity

In IoT applications such as health, energy and logistics monitoring, it is essential to ensure that the data are not altered by any means. Data integrity assures that the data are not modified deliberately by any adversary or inadvertently during transmission. Data integrity in the IoT domain can be ensured using hash functions in Message Integrity Codes and checksums.

2.2.3 Authentication

Authentication verifies if the data originator is the one that it is claiming to be, preventing data insertion. One of the ways that authentication can be achieved is by using digital signatures. The digital signature of the sender is attached to the message so that the recipient can verify the identity of the source of the data.

2.2.4 Availability

Availability assures that a system is always available providing in a timely and reliable manner services to the authorized individuals at any time. Due to the nature of resource-constrained devices, availability constitutes a challenge for the research community of IoT, since those devices are more vulnerable to attacks that can exhaust their computational and energy resources.

2.2.5 Freshness

In many communication schemes Nonces in combination with sequence numbers ensure the freshness of data [5]. This attribute guarantees that valid data are not resent or sent delayed by any party within the communication, or by any adversary in a given period of time, making always sure that they are fresh. In IoT domain, due to the lossy nature of resource-constrained devices, many data can be lost; therefore, freshness is a very important attribute to be preserved.

2.2.6 Forward - Backward Secrecy

In group communication, forward and backward secrecy are vital concepts that a network administrator should bear in mind before defining any group policies. Backward secrecy safeguards the communication of a group upon new member registration, such that no data from past communication within the group, can be retrieved by the new member. Forward secrecy on the other hand ensures that no data can be accessed by a member that is evicted from a group.

2.3 Security Protocols

2.3.1 Internet Protocol Security (IPsec)

The internet protocol security (IPsec) provides end-to-end security services, between two hosts, two gateways, or a host and a gateway. It comprises the protocols, the Authentication Header (AH), and the Encapsulating Security Payload (ESP) that can be applied separately or in combination with each other to enable security [6]. Both AH and ESP, provide integrity, data origin authentication, optional replay attack protection and access control. ESP additionally can offer confidentiality. They support two different modes of operation, the transport mode and the tunnel mode. Tunnel mode is used typically for communication within a corporate network between gateways. The original IP packet is entirely protected by IPsec, and a new IPsec header precedes the original IP header along with the upper layer protocol headers. The main difference between AH and ESP in tunnel mode is that, ESP does not sign the new IP header while AH partly does. On the other hand, transport mode is typically used in end-to-end communication between two hosts, where the IPsec header is appended immediately after the IP header, protecting the next layer protocol payloads.

The security services that a particular IP traffic affords, are called Security associations (SAs). The network administrator is responsible to define the security requirements that have to be fulfilled by the SAs within a network, depending on the end user traffic utilization. AH and ESP use SAs to determine the security services that they carry out in an IP traffic. The processing model of IP traffic utilizes three types of databases, the Security policy database (SPD), the Security association database (SAD), and the Peer authorization database (PAD). SPD maintains the policies for all types of IP traffic, inbound and outbound. SAD contains all the parameters that are associated with security associations. Finally, PAD is a database that links SPD with the security association management protocol, by indicating among other things which peers are authorized to use a specific IPsec entity. The policies defined in SPD, indicate how a specific traffic should be handled, either to be protected by IPsec, to be discarded, or to be bypassed by IPsec.

In unicast communication, the SAs are identified by their Security parameter index (SPI) and can be either generated manually or dynamically by using a security association management protocol, such as the IKEv2 protocol, which is described in the next section. In case IPsec processing is required for outgoing traffic, then a SAD lookup is performed using the SPIs. If there are no SAs found for the specific SPI, internet key exchange version 2 (IKEv2) is used to negotiate the SAs between two peers and install the newly negotiated SA in the SAD. If an SA exists, then IPsec protocols are able to process the traffic. In case of inbound IPsec protected traffic, for which no SAs exist in SAD, the traffic is dropped and the event is added in the audit log.

The security services deployed in multicast communication are called GSAs. They are usually set by a central entity, the Group Controller/Key Server, which is also responsible for the assignment of Group SPIs without any intervention of a group key management protocol like IKE. SAD stores all types of security associations, for both multicast and unicast traffic. Therefore, it is of high importance that during the IPsec lookup, SAs and GSAs are distinguishable. Chapter 4 provides a detailed description of how an IPsec entity is amended in order to support incoming and outgoing multicast traffic.

2.3.2 Internet Key Exchange version 2 (IKEv2)

The purpose of IKEv2, [5], is to dynamically set SAs between two peers, the first having the role of the original initiator and the second the role of responder. It supports two types of Security Associations, IKE SAs and Child SAs. The first type determines how keys are generated, specifying parameters for Diffie-Hellman key exchange, [2], cryptographic, integrity transforms, and pseudo-random functions that generate random numbers for the communication sessions. Child SAs define the sets of encryption and integrity transforms that are used to protect traffic in IPsec protocols.

IKE SAs are set in the initialization of the protocol by exchanging IKE SA INIT messages between the peers. Through this type of message the two parties negotiate the IKE SAs that they are going to use to encrypt and ensure the integrity of all types of IKE messages after the initial exchange. In particular, they exchange their nonces and

Diffie-Hellman numbers. Then IKE_AUTH messages are exchanged, to authenticate the aforementioned messages, to exchange identities and (optionally) certificates between the parties, and to establish the first Child SA. In many cases, more than one child SAs are needed to be set. Therefore CREATE CHILD SA messages are exchanged, setting the required parameters. For error reports, maintenance and deletion of SAs, INFORMATIONAL messages are sent using the appropriate payloads. It should be mentioned that all IKE communications are performed in pairs, sending request and response types of messages.

Despite the fact that IKE can be used to dynamically set SAs for IPsec protocols, AH, and ESP for unicast communication, there is no such possibility for multicast communication. Group security associations are either predefined and established for a group, or usually built as an aggregation of SAs of different purposes, like registration, rekeying, and data security. The senders and receivers of traffic in the latter case are interacting with a GC/KS using a group security association management protocol in order to request registration to the multicast group. GC/KS is the management entity of the group, responsible for authenticating the identity of the requesting entity and for checking if it is authorized to join the group. Upon successful authentication and authorization, GC/KS provides to the requesting entity the GSAs, and key material, as described in the following section.

2.3.2.1 Group Key Management using IKEv2

The Internet draft proposed by Rowles, [22], provides a solution to enable end-to-end secure group communication in any IPsec protocol, using Group Key Management in IKEv2. It enables member registration in an insecure channel and secure distribution of a group key within the group. There are three types of messages introduced in addition to the original IKEv2 message types, GSA AUTH, GSA REGISTRATION, and GSA REKEY messages and three extra next payload types, Group identification (IDg), Group security association(GSA), and Key download (KD). Furthermore, the terms of Keys encrypting Key (KEK) and Traffic encrypting Key (TEK) are included, in order to describe the keys that encrypt the Group Key Management messages, and the keys that encrypt specific protocol traffic respectively.

In the Internet draft [22], two peers with different roles, the candidate member and the Group Controller/Key Server, exchange IKE SA INIT messages to negotiate cryptographic suites, exchange Diffie-Hellman values, and nonces. Following this exchange, and assuming that the candidate members initiate the group registration, a GSA AUTH request is sent to the GC/KS in order to authenticate the previous negotiation, and at the same time to request for registration to the desired group. If the candidate member is authorized to be part of the group, a GSA AUTH response is sent back, including the GSAs and key material. In an already established secure channel, the candidate member requests for registration to a group sending a GSA REGISTRATION request to GC/KS, which in turn responds with the corresponding GSAs and KD. Upon updating group memberships, GSA REKEY is required to be sent by GC/KS to the group members, either to update their GSAs and key material or to delete them. It is

important to be mentioned that this draft proposes two approaches for distributing the key material to the group members, either by rekeying the group members with a single GSA REKEY message destined to all members or by using a group key distribution algorithm for distributing and organizing in an efficient manner the key material within a message.

3

Related Work

This chapter presents existing work that has been done towards enhancing security in group communication for IoT. Many efforts have been made during the past decades to enable secure group communication in traditional networks, by determining the required means to handle the key material of a group.

The principles and requirements of a group key management architecture are stated in [8]. As an informational RFC, it provides an insight into the features that should be included in a group key management protocol, so that it is compatible with any application, transport, or network protocol, which is intended to be achieved in this master thesis. Moreover, it indicates the mechanisms that should be considered for registration, de-registration, and re-keying of group members. It also suggests the use of group key distribution algorithms to maintain scalability with the minimum messages exchanged among users.

According to surveys, there are three different approaches to select an appropriate group key management algorithm, centralized, distributed, and decentralized approach [18][19][20]. In the centralized approach, a central entity, which is considered to have higher computational power, is responsible for the key distribution and generation of the group key within the multicast group. In the distributed approach, all the members of a group are equally responsible for the distribution of keys and the derivation of a group key. In this approach, all the nodes are equally secure, trusted, and have enough computational power to generate keys. Whereas, in the decentralized approach, a group is divided to subgroups, in each of which, one entity is the subject of key distribution and derivation. In both distributed and decentralized approach it is assumed that the nodes have enough computational power to generate keys. This assumption is unlikely most of the times in IoT since most of the devices have limited resources. A centralized approach is adopted in our solution with a 6LoWPAN border router as the GC/KS, responsible for storing the keys and distributing them to the group, but not generating them.

A solution for enabling secure group communication in the IoT domain was provided in [3], in which the authors proposed establishment of secure group communication using DTLS for multicast traffic. It empowers security in the transport layer, creating separate DTLS sessions for each application running on the device. This is in contrast with Multicast IPsec in which secure communication is established in the network layer for all the applications running on top of the device. The selection of the layer that security should be enhanced clearly depends on the security requirements of the particular applications running on top of a device, and on the capabilities of the device. Moreover, Multicast DTLS mandates the adaptation of a particular application to the needs of DTLS, whereas multicast IPsec is independently configured without requiring any explicit action in any application.

Multicast DTLS assumed a centralized approach to key management similar to this thesis, where a group controller is responsible for providing the GSAs and key material to the group with a standardized key management scheme. In particular, there are two roles defined in this protocol. First, a device within the 6LoWPAN network that is configured with the multicast IPv6 address and is able to send multicast traffic, having the role of the sender. Second, a device with the role of a listener that is configured with the multicast IPv6 address and maintains the GSAs and key material of the group. The sender is able to encrypt and authenticate a multicast request message using the group security associations and key material, and send it to the corresponding multicast group. A potential listener is able to decrypt and authenticate the message using the multicast IPv6 address and destination port number in order to find the corresponding GSAs and key material. The listener in turn is able to reply to the sender by sending a group message response which is encrypted with individual key material derived from the group key material. The original sender upon reception of such message is able to decrypt it and verify its authenticity by using the group key material and deriving the individual key of the particular listener.

Two solutions for group key establishment in IoT that enable secure group communication are presented in [21]. The authors propose two approaches to secure derivation of a group key using Elliptic curve cryptographic operations among the initiator of the multicast communication and the active group members. The first protocol is more suitable for a distributed system whereas the second for a centralized approach. In both protocols group key derivation is done by exchanging broadcast and unicast messages within the network and deriving the group key using public and private keys of the group members. Authenticity of the exchanged messages is ascertained by using public key cryptography. The proposed solutions use well-known methods for key derivation. However, there is no discussion regarding the integration of this communication scheme with any standardised solution for secure multicast communication such as IPsec, unlike with our solution.

The ultimate goal of a group key management protocol in IoT is to exchange the least messages as possible with the minimum size, while preserving security. This challenge motivated researchers to develop group key distribution algorithms that could be integrated to existing security association management protocols like in Group Key

Management using IKEv2 [22]. It is vital to choose a suitable algorithm based on the requirements of a specific application or protocol.

The authors in [7] were among the first that pointed out the need for a key distribution algorithm in a group key management protocol. Using such an algorithm, the group can be resistant to compromise by an adversary, ensuring perfect forward and backward secrecy, use minimal storage and generate minimal communication overhead. They present four different approaches to distribute keys in a group, among which hierarchical tree approach is claimed to balance the diverse requirements of applications.

Eliminating security vulnerabilities in multicast communication, while keeping communication and storage overhead to the minimum in the members of the group, are the requirements of IoT. Logical Key Hierarchy (LKH), is claimed to fulfil these requirements, by reducing the number of rekeying messages to O(log m) with size 2 $\cdot \log_d m$ and stored keys at most to $(1 + \log_d m)$, where m is the number of group members and d the tree degree, along with providing security features [18]. Its basic principle is that each user maintains an individual key that is considered to be a leaf of the tree, the intermediate nodes are KEK, and the root of the tree is the group key. Each user keeps record of the keys that are part of the path to the root. There are various modified versions of LKH such as LKH+, Diffie-Hellman LKH (DH-LKH), Distributed LKH (DLKH) [19], Search tree LKH (S-LKH), B+ Tree LKH (B+LKH) [23], LKHTreeManager (LTM) [24], and a heuristic search algorithm for LKH [25]. These modified versions aim to reduce the exchanged messages and the message size.

In LKH+, members can calculate new keys from the old ones and send them to other members, which arises security issues, since knowledge of prior keys is required. This means that a potential key compromise can lead to a security loophole, which is undesirable. In Diffie-Hellman LKH and DLKH, members generate KEK within the path to the root and there is no entity that has knowledge of all the keys. Diffie-Hellman LKH uses Diffie-Hellman algorithm for the key generation of KEKs in the upper levels of the tree. On the other hand, S-LKH and B+LKH simply follow different tree construction methods for initialization and key updates. As it is described in [23], S-LKH uses a simple search tree structure, whereas B+LKH constructs B+ search trees to retain balance of the tree. LTM [24], uses AVL management techniques to determine the suitable join and leave node location in the tree to preserve the balance. For both B+LKH and LTM, designers have to consider the trade-off between keeping the tree balanced (so that fewer rekey messages are exchanged) with the cost of increasing computational complexity (which is not desired for resource-constrained devices). Compared with the aforementioned algorithms, heuristic algorithm for LKH proposes a dynamic formation of the LKH tree with different degree value for each level in the tree. The degree value of the nodes in each level is computed as a function of eviction and join probabilities of the nodes such that the encryption and transmission overhead are optimized, which is a practical approach for IoT.

A group key distribution algorithm that competes with LKH is introduced in [18],[19],[26] and [27] is One-way function tree (OFT). Similarly to LKH, the tree's root is the group key and the leaf nodes are the member keys. The difference between OFT

and LKH is that KEKs are generated by the nodes of the tree instead of assigned by the central entity. More specifically, only the leaf-member is assigned a secret-key, from which it computes the so-called blinded key, using an one-way function. Intermediate nodes within the path from a leaf to the root i.e, KEKs, and the root (group key) are generated by mixing the blinded keys of their children. All the intermediate nodes from a leaf to the root make the ancestor set and the siblings of the ancestor set construct the sibling set. A group member possesses its own secret-key assigned to its leaf, the calculated keys (unblinded keys) of the ancestor set, and the blinded keys of sibling set, which is the same amount of keys as in LKH, if in both algorithms the tree is balanced.

The adoption and enforcement of a group key distribution algorithm presumes the existence of a secure communication scheme to enable the actual distribution of the key material, like IKEv2. This thesis work proposes a new protocol, Group-IKEv2 for IoT, which is based on the Internet Draft Group Key Management using IKEv2 [22], for dynamic establishment of group security associations and keys. The use of a group key distribution algorithm in our solution is optional and depends on the policies defined for a particular system and application. The following chapter presents the architecture proposed in this thesis work with a detailed description of the overall system, the features added in the existing protocols and the group and system policies that need to be taken into consideration in such a system.

4

System Architecture

This chapter introduces our system architecture that is proposed in this thesis work and provides a detailed description of how an existing system, operating with the IPsec protocols and IKEv2, can be extended such that secure group communication is enabled. Initially, some usage scenarios are outlined, highlighting the need for such architecture extension. Later on, multicast IPsec and Group-IKEv2 for IoT are presented, with a detailed description of their data structures, messages format, and payloads. Section 4.3.6 provides an insight on the key distribution algorithms that can be integrated with a group key management protocol such as Group-IKEv2. Finally, Section 4.4 presents the system and group security policies that should be taken into consideration by the network administrator in such a system.

4.1 Usage scenario

The necessity of secure group communications has already been determined since mid-nineties. From that time, it was evident that applications would require a secure, simultaneous communication of multiple parties, that share the same interests, needs, functionalities, and requirements. Now, with Internet of Things, it is even more essential to utilize efficiently the provided bandwidth and resources, while enabling secure group communication.

One potential usage of secure group communication is in autonomous cars, where all the sensors with the same functionality communicate with each other. It is of high significance that the communication scheme used among the sensors, is secure from external risks, and at the same time efficient. One way to achieve the first goal, is to enable security in the Network layer of OSI, by using the IPsec protocols. In this case an IPsec secure channel is established between each sensor pair, enabling pairwise data exchange. However, in that way efficiency cannot be achieved, since pairwise exchange of data in a network of arbitrary number of sensors, k, requires k^2 separate IPsec tunnels,

which is very costly for each sensor's memory and energy resources, as well as for the network's resources.

Another potential usage, is the remote control system of home appliances. Home appliances can be controlled remotely using IPsec protocols and establishing, in a pairwise manner, secure channels between the user and each home device, as in the previous scenario. This way is sufficient in a case where the user wishes to remotely control each device separately. However, if the user wishes to remotely control a group of home appliances, this way is inefficient, magnifying the communication overhead in the network.

The architecture proposed in this thesis, accomplishes the establishment of a secure channel among a group of devices. Practically, in the first scenario the sensors, utilizing the attributes of multicast IPsec with Group-IKEv2, are able to forward the data to a particular group of sensors. In the second scenario, instead of insufficiently controlling each device one by one, the users are able to control securely and simultaneously a group of devices. The following sections delineate the complete system architecture that is proposed in this thesis.

4.2 Multicast IPsec for IoT

The previous section has made evident the need for enabling secure group communication. Yet, as it has been already mentioned in Chapter 2, the IPsec standard offers only end-to-end secure communication among two peers and not among a group of network entities. This section presents in detail the extensions that should be applied in IPsec data structures in order for IPsec to support secure multicast traffic in IoT.

In an analogous manner as in unicast traffic, the security parameters that are required in a particular multicast traffic, are called Group Security Associations. They are set by a central entity, the Group Controller/Key Server, as it is defined in the Security Architecture for the Internet Protocol, [6]. However in the aforementioned standard, GC/KS is responsible for the assignment of Group SPIs, without any intervention of a group key management protocol, as the GSAs are statically configured in each network entity. This thesis work, in contrast with the aforementioned standard, has adopted the optional Multicast Extensions to the Security Architecture for the Internet Protocol, [29], in which the use of any group key management protocol, is essential to dynamically distribute Group SPIs, GSA, and key material. The motivation for this choice is that a group of resource-constrained devices, within the IoT domain, can be exposed to different levels of security risks depending on the current environment, and the group can often suffer of membership changes. For these reasons it is more secure and efficient to change dynamically GSAs depending on the current security needs of the multicast group instead of manually configuring them in the network nodes. This thesis describes an extension to IKEv2, Group-IKEv2, as a group key management protocol that accomplishes dynamic GSA allocation to a multicast group. In the following section, extensions of the IPsec architecture are thoroughly presented.

4.2.1 IPsec Database Extensions for multicast

This section discusses the extensions that need to be made in IPsec in order to support multicast traffic. We adopted extensions proposed in Multicast extension of the IPsec protocol [29], which introduces an expanded version of the databases included in the IPsec architecture, SPD, SAD, PAD.

Group security policy database (GSPD) is the expansion of SPD, and is able to support both unicast and multicast traffic. The purpose of this database remains the same, as per to identify the policy applied for a particular inbound or outbound traffic, to be "protected", "bypassed" or "discarded". The notions of SPD cache remain the same as well, with GSPD-I and GSPD-O to represent inbound and outbound GSPD cache respectively.

We have adopted the extensions of [29], in which two new attributes are added in the GSPD entries. The first is related to address preservation in tunnel mode of IPsec, with two options, local tunnel address (for source address), and remote tunnel address (for destination address), to indicate that the inner IP header address is used to the outer IP header address, during IP header construction in tunnel mode. The second attribute added in GSPD entries is the directionality. Depending on the specific application (TCP, UDP port), the directionality of GSA should be indicated. There are three different types of directionality, the "symmetric", the "sender-only", and the "receiver-only". The "symmetric" type is used when a particular GSA is applied to both inbound and outbound GSPD. The "sender-only" is used, in case the particular multicast traffic is allowed to be sent, but not to be received, indicating that only GSPD-O cache uses this policy and the traffic is discarded in GSPD-I. Last but not least, in the "receiver-only" directionality, the particular GSA is applied only in GSPD-I, since the multicast traffic is accepted, whereas outbound traffic is discarded by GSPD-O. The Discard events can be registered locally in the device in the audit log [29],[6].

The Group security association database (GSAD) is the expansion of SAD with the capability to store all types of security associations, for both multicast and unicast traffic. Depending on the directionality defined in the corresponding GSPD entry for the particular multicast traffic, the analogous GSAD entries are created. However, it is possible that GSAD entries for particular traffic already exist, even though there are no GSPD entries for protecting the traffic, which can be resulted by GSPD changes. In that case the traffic is handled in accordance with the GSPD policy.

The expansion of PAD in multicast IPsec is called Group peer authorization database (GPAD). This database is the link between the group key management protocol and the GSPD. GPAD identifies which GC/KS is authorized to provide GSAs and key material to potential group members for a particular multicast traffic. It also determines the method used to authenticate GC/KS, providing the corresponding authentication data to the candidate member. Moreover GC/KS using this database is able to determine which candidate members are authorized to send and receive multicast traffic. All these functionalities are used in conjunction with a Group Key Management protocol, in order to dynamically assign GSAs and provide key material to requesting members in a group.

Within the purpose of this thesis, it is assumed that all network entities are

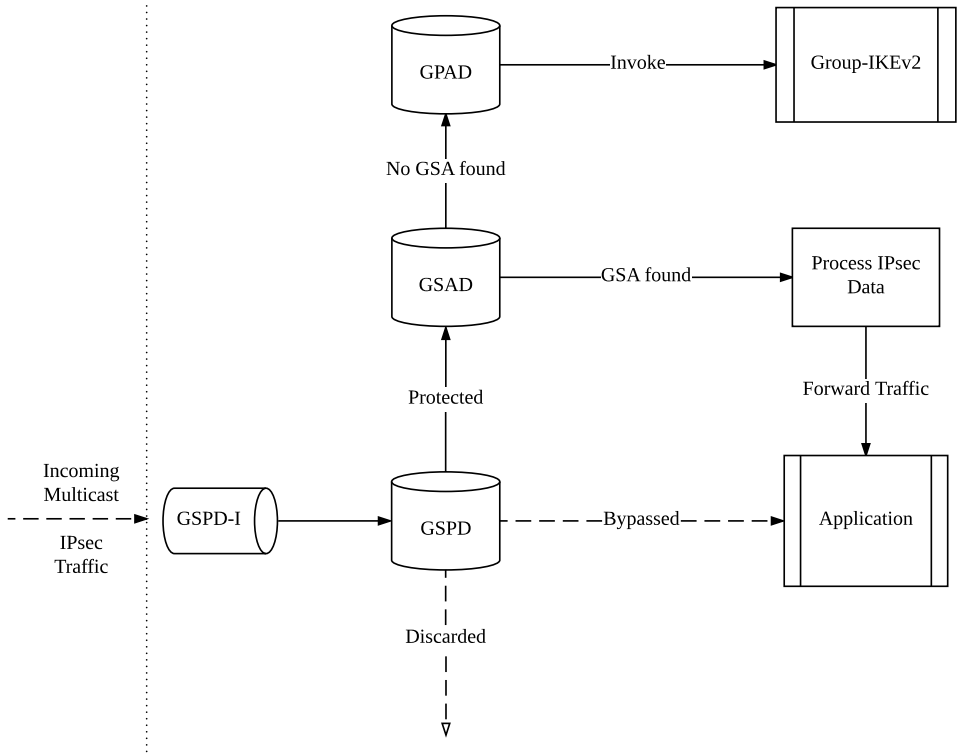

Figure 4.1: IPsec entity representation. Inbound IPsec Data processing.

configured with multicast IPsec extensions, with predefined group policies in the GSPD and predefined GC/KS in the GPAD, and that the network administrator has already configured each network entity with the multicast IPv6 address. Figure 4.1 displays how inbound multicast traffic is processed within an IPsec entity of a node. Upon reception of the inbound multicast traffic, the IPsec entity processes the packet headers to determine the source, destination address, and port, in order to lookup for the corresponding GSPD entry in GSPD-I cache. All the entries in this cache are obtained by GSPD. If there is no policy for the traffic, or if the found policy is for "sender-only" directionality, the traffic is discarded. If the found policy has, either "symmetric" or "receiver-only" directionality, IPsec entity proceeds with looking up through the GSAD, to find the "longest" match between the source address and SPI of the incoming packet, and the source address and SPI of a GSAD entry. If no GSAD entry is found, then GPAD provides to the IPsec entity, the ID of the authorized GC/KS for this traffic. Then it requests from GK/KS for GSA and key material by invoking Group-IKEv2 Protocol. In a successful GSAD lookup, the IPsec entity is able to process the incoming IPsec data, using the GSA and key material found in the GSAD entry. It should be noted that the IPsec entity is looking for the "longest" match in GSAD, so that unicast and multicast GSAs are clearly

distinguishable.

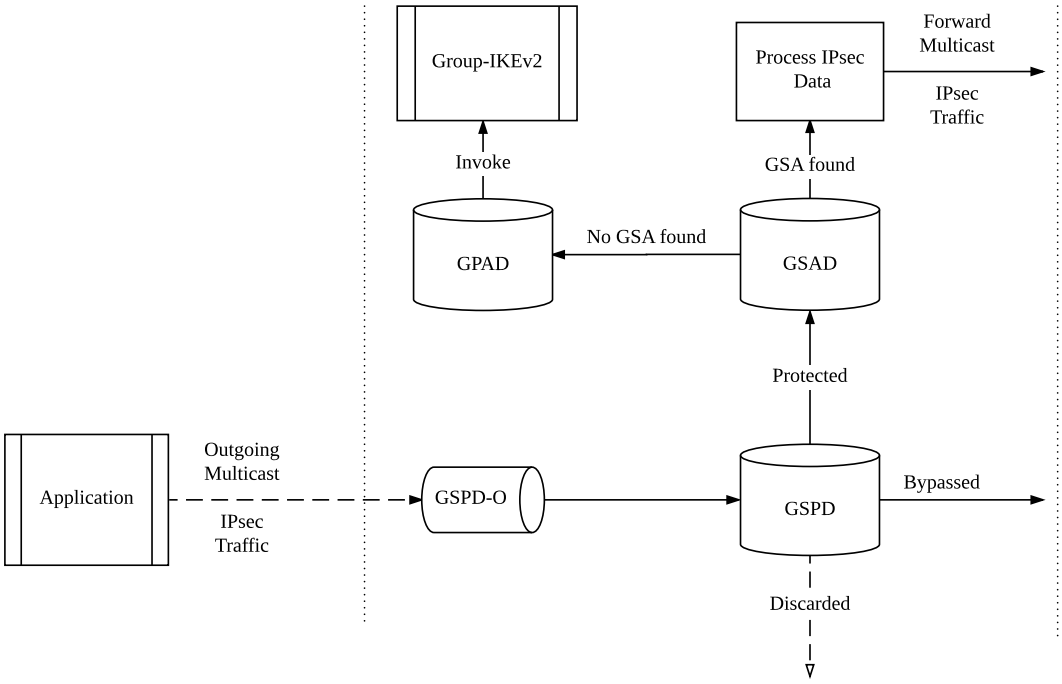

Figure 4.2: IPsec entity representation. Outbound IPsec Data processing.

When a network entity wishes to send outbound multicast traffic, the application layer of the device forwards data to the network layer, in which the IPsec entity is lying. Then, the IPsec entity initiates a lookup in GSPD-O to determine if there is a policy for the given multicast destination and port. If no policy was found, or if the found policy had "receiver-only" directionality, the traffic is discarded and the event is added in the audit log. If a policy is found, either with "symmetric" or "sender-only" directionality, then the IPsec entity goes ahead to initialize GSAD search, so that the corresponding GSA and key material can be found as it is shown in Figure 4.2. Similar to the inbound traffic, the "longest" match has to be found, searching for both SPI and destination multicast address to match with the corresponding attributes of a GSAD entry. If no entry is found, then GPAD provides to the IPsec entity the ID of the authorized GC/KS for this traffic, such that the device can request the GSAs and key material for this outbound traffic, by invoking Group-IKEv2 Protocol.

The following section outlines how Group-IKEv2 is used by a candidate member in order to obtain GSAs and key material for multicast IPsec traffic.

4.3 Group-IKEv2 for IoT

The group key management protocol proposed in this thesis, is based on the IKEv2 standard [5] and the Internet draft, Group Key Management using IKEv2 [22]. Our proposed protocol is able to dynamically allocate group SPIs, GSA, and key material that are to be used by multicast IPsec. Moreover, our protocol is destined to be deployed in a network of resource-constrained devices, where communication overhead, along with power and CPU utilization is of high significance. Before explaining how IKEv2 is extended, it is essential to understand how IKEv2 works.

IKEv2 standard enables the network entities to pairwise negotiate with other peers the security associations and key material, that are to be used in unicast IPsec. The first party, intending to communicate with another peer, initiates the negotiation only if there is no established security association for the IPsec protocols in its SAD. In that case, IKEv2 is invoked and the entity takes over the role of the "original initiator", sending to the other peer an IKE_SA_INIT request. The peer receiving the request is taking the role of the "responder" and replies back with an IKE_SA_INIT response. Up to this point, the peers have negotiated the IKE SAs that are to be used in later IKE messages and exchanged Diffie-Hellman group numbers and Nonces. Upon reception of IKE_SA_INIT response, the original initiator continues with the IKE_AUTH exchange, with which both initiator and responder authenticate each other and establish Child SAs. Child SAs are used by the IPsec protocols to secure the traffic between those two peers. In case that a peer needs to communicate with multiple peers at the same time using IPsec, separate IKE sessions have to be established, to negotiate separately the SAs for each pair. This process becomes very costly with respect to network and device resources, if an entity within the network needs to communicate with large number of nodes.

Therefore, we proposed an extension of IKEv2, Group-IKEv2, which is based on the Internet Draft Group Key Management using IKEv2, with which a network entity can obtain in a dynamic way the GSAs and key material that are to be used in multicast IPsec. In short, a network entity configured with multicast IPsec, requests the GSAs and group key material by a centralized management entity, which is under the network administrator's authority. If it is authorized to be member of the group, it receives the GSAs and group key material and then it is able to send and receive securely data using multicast IPsec.

4.3.1 Extensions to IKEv2 for Group-IKEv2 in IoT

Since we are dealing with group communication, there is the need to categorize the GSA depending on the protocol carrying the traffic. The group security associations used in IPsec protocols are called GSAs for encrypting Traffic (GSAT), and the ones used in Group-IKEv2 are called GSAs for encrypting Keys (GSAK) [22]. This classification is derived from the fact that GSAs for Group-IKEv2 are used to encrypt and integrity protect the GSAs and key material that are used by IPsec protocols. It is highly recommended that GSAT and GSAK to be different. Group-IKEv2 messages should be

encrypted and integrity protected using different transforms and key material than IPsec protocols to achieve more secure membership management. The key material provided to the authorized members in Group-IKEv2 is called the KEK Key Download, KD_{KEK}, whereas the key material used by IPsec protocols is called TEK Key Download, KD_{TEK}.

Two types of entities are defined in this architecture, similarly to [22], the candidate member and the GC/KS. The candidate member is the network entity that needs to obtain the GSAs and the key material in order to be able to use multicast IPsec and be part of the security group. The Group Controller/Key Server (GC/KS), is an entity within the network that maintains all the GSAs and the group key material to be used by both IPsec protocols and Group-IKEv2 for a given multicast traffic. This entity is under the absolute control and supervision of the network administrator, using a centralized approach of group management.

Group-IKEv2 follows a similar invocation principle as IKEv2. As it is described in Section 4.2.1, a candidate member, aiming to establish secure communication with multicast IPsec, initially performs a GSPD lookup to determine the policy for the given multicast traffic. Then, if a Protect policy is applied, the IPsec entity proceeds with GSAD lookup to check if a GSAT exists. If no GSAT entry exists, GPAD ascertains which is the authorized GC/KS for this traffic and provides its ID to Group-IKEv2. Having this information, the candidate member will be able to request from the GC/KS the GSAs and key material for this multicast group. The candidate member initiates the protocol by sending an IKE_SA_INIT request to the GC/KS, for exchanging nonces, Diffie-Helman values, encryption, and integrity transforms that are supported. At the same time, it indicates with a Notify payload that Group-IKEv2 is initiated. In turn, the GC/KS sends back to the candidate member an IKE_SA_INIT response, providing the selected transforms that are going to be used in the GSA_AUTH message, its Nonce and Diffie-Helman values, similarly to IKEv2. However, upon reception of the Notify payload, GC/KS also invokes the Group-IKEv2 protocol. In that way a secure channel is established between the candidate member and GC/KS before the candidate member proceeds with requesting registration to the multicast group.

As it can be seen in Figure 4.3, when IKE_SA_INIT is received, the candidate member sends a GSA_AUTH request message. The GSA_AUTH message exchange has the same functionality as IKE_AUTH, the two peers to be mutually authenticated, since the candidate member and the GC/KS exchange their identities $\mathrm{ID_m}, \mathrm{ID_s}$. In addition to this functionality, a candidate member sending this message also requests for becoming a valid member of a particular multicast group. This is done by including the Group Identification Payload, $\mathrm{ID_g}$, in the message, which indicates the multicast group that the candidate wishes to join. GC/KS parsing such a payload, initializes first a lookup in GSPD and then in GPAD. With the first lookup it is able to determine, if the traffic to or from the specified multicast group is protected with the IPsec protocols and to locate the corresponding GSAs. By looking up the GPAD, GC/KS is able to verify if the requesting candidate member is authorized to become member of the multicast group. If there is a policy for this traffic and the candidate is authorized, then GC/KS searches through GSAD for the corresponding GSAs. If GSAs exist for traffic to and from this multicast

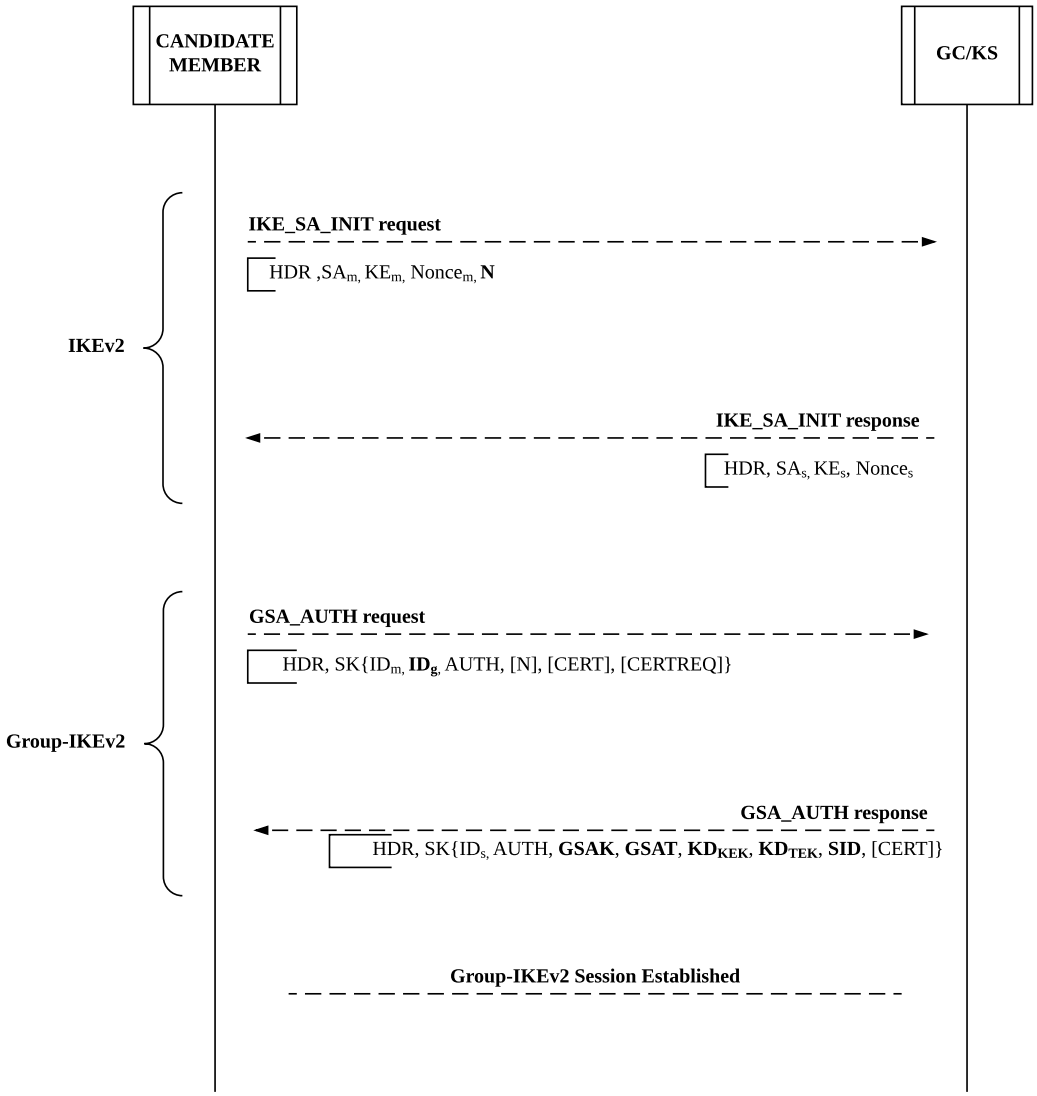

Figure 4.3: Group-IKEv2 message exchange. A candidate member requests from GC/KS to register to the multicast group.

group (for policy with symmetric directionality), GC/KS sends back to the candidate member a GSA_AUTH response message including the GSAs and key material that are to be used by IPsec protocols (GSAT), and also GSAs and key material to be used by future Group-IKEv2 messages (GSAK). Then the validated member will update its GSAD, enabling the capability to receive and send multicast IPsec traffic. Additionally, the member will update the Group-IKE session information to accept Group-IKEv2 messages. In the unfortunate occasion at which the candidate member is not authorized to be a part of the security group, GC/KS initiates an INFORMATIONAL exchange,

with a Notify payload indicating that the candidate was not authorized to be part of the group. This results in the termination of the Group-IKEv2 session.

After successful member registration to the group, GSA_REKEY messages are sent by GC/KS under three circumstances. The first is when the lifetime of GSAs is close to expire. In this case a periodic GSA_REKEY message is sent to renew the GSAs and key material of the group. The second is when a new member wishes to join the multicast group. In this case a GSA_REKEY message is sent to the existing members of the group before registration of the new candidate member is done, such that backward secrecy is ensured. The third occasion is when an existing member of the group wishes to leave the group or it is compromised by an adversary and has to be evicted from the group. In such case, forward secrecy is ensured by updating the GSAs and key material to the rest of the members in a way that the leaving member cannot participate to the rekeying process itself. The network administrator is responsible to define the group policies that should be applied by the GC/KS to all members and set the lifetime of GSAs. The most vital and difficult to define policies are the ones performing member eviction from the group in case the member wishes to leave the group or it is compromised by an adversary. More information about the policies that should be considered in our proposed solution are provided in Section 4.4.

As it has been already mentioned, our proposal is based on the Internet Draft Group Key Management using IKEv2 [22]. However, for the purposes of this thesis, we adjusted some features of [22], such that our proposal fulfils the requirements of resource-constrained devices. One of the changes that have been made is that we did not adopt the use of GSA_REGISTRATION messages, as it is proposed in the Internet Draft. According to the draft, the member registration process is performed with the exchange of IKE_SA_INIT, GSA_AUTH and GSA_REGISTRATION. Due to the fact that GC/KS can optionally provide the GSAs and key material in GSA_AUTH response, the candidate member has to send a GSA_ REGISTRATION request, using the security associations and key material that were negotiated with the GC/KS, in order to request registration to any group it wishes to join.

From a performance perspective, by not using this message, we perform the registration one step earlier, minimizing the time required for a member to register for the very first time to the group. Thus, if a candidate member wishes to join only a single group, the registration time and the energy consumption would be significantly reduced. On the other hand, registration to multiple groups using a single IKEv2 session would require less total number of messages exchanged between GC/KS and a candidate member. This leads to less energy consumption, since only two messages (GSA_REGISTRATION messages) would be exchanged instead of four (two IKE_SA_INIT and two GSA_AUTH messages). However, that would require the members to store through their whole lifetime IKEv2 session information. This would burden their memory, compared with temporarily storing IKEv2 session information, especially if registration to a single group is needed. From a security perspective, using the same IKEv2 session for registering to multiple multicast groups, which serve different purposes and probably are used by different applications of the device, poses a higher security risk than using separate IKEv2

sessions for each group. Although the applications running on top of the devices are not aware of the network layer security, an adversary might be able to take advantage of this feature to compromise the communication among GC/KS and the candidate member.

In addition to the aforementioned amendment, we introduced a new payload of Sender ID, which enhances the authorization process of incoming multicast IPsec traffic as it is described later on in Section 4.3.5.6. Last but not least, we have neglected particular payload fields proposed in the draft, as their functionality was not needed in Group-IKEv2 for IoT. The following sections describe in detail the original IKEv2 message types and payloads, that are used by Group-IKEv2 along with the new message types and payloads that are defined in Group-IKEv2.

4.3.2 IKEv2 Messages used in Group-IKEv2

This section presents IKEv2 messages that are used in Group-IKEv2 for IoT. As it has already been mentioned, Group-IKEv2 is initialized in both the candidate member and the GC/KS by exchanging IKE_SA_INIT messages, such that they can proceed with negotiating the IKE SA and keys that are going to be used in GSA_AUTH messages.

4.3.2.1 IKE SA INIT

In particular, IKE_SA_INIT request message is sent by the candidate member to the Group Controller/Key Server (GC/KS) to initialize negotiation of IKE SA and key material that are going to be used in the next IKE message. By sending an IKE_SA_INIT request message, the candidate member shares the supported cryptographic suites, which are included in the SA payload SA_m, its selected Diffie-Hellman group, which is included in Key Exchange Payload KE_m and its nonce, N_m. *This thesis proposes to also include the Notify payload* ***N****,* to differentiate the initialization of Group-IKEv2 session from IKEv2 session. This is an amendment to IKEv2 standard and Group Key Management using IKEv2 Internet Draft [22]. It notifies the GC/KS to initialize Group-IKEv2 process, so that the GC/KS is aware of the expected messages and payloads that are going to be sent later on by a candidate member. It is worth mentioning that this type of message is not encrypted or integrity protected. The format of IKE_SA_INIT request is shown in Table 4.1.

HDR,	SA_m,	KE_m,	N_m,	**N**,	[Cookie]

Table 4.1: IKE_SA_INIT request message format with the Notify payload included to indicate that Group-IKEv2 is initialized in the candidate member.

Upon reception of IKE_SA_INIT request, the GC/KS ascertains if there is a Notify payload for Group-IKEv2 and acts accordingly. After Group-IKEv2 initialization, GC/KS sends an IKE_SA_INIT response message to the candidate member. The payloads included in this response are the Security Associations payload SA_s, containing the

selected cryptographic transforms, the Key exchange payload KE_s, containing selected Diffie-Hellman group, and GC/KS's nonce N_s.

It should be mentioned that Cookie payload is optionally included and follows the format of IKEv2 standard. It is used to prevent Denial of Service attacks, in which an adversary sends a bulk of fake messages to the victim to overwhelm it with the computation of many modular exponentials for Diffie-Hellman public key derivation [28]. The format of IKE_SA_INIT response is shown in Table 4.2.

HDR,	SA_s,	KE_s,	N_s,	[Cookie]

Table 4.2: IKE_SA_INIT response message format.

4.3.3 New Messages for Group-IKEv2 in IoT

This section presents the new message types defined in our protocol, Group-IKEv2, which are based on Group Key Management using IKEv2 internet draft,[22], with necessary alternations. The particular payload format modifications are also discussed in this section.

4.3.3.1 GSA AUTH

The main purpose of GSA_AUTH request in Group-IKEv2 is for the candidate member to request registration in a particular group. This is accomplished by including the Group Identification Payload ID_g in GSA_AUTH request message such that it is indicated to the GC/KS which multicast group the particular request refers to. GSA_AUTH is also used to authenticate the parties of IKE_SA_INIT message exchange, by using the candidate member's Identification payload, ID_m, and the GC/KS's Identification payload ID_s in combination with the AUTH payload in correspondence with IKEv2 standard. The Certificate request payload, *CERTREQ*, and Certificate payload, *CERT*, are optionally included depending on the administrator's decision on authentication method. If no certificates are included, any of the authentication methods mentioned in [5] Sections 2.15 and 3.8, can be used.

In short, after the candidate member receives the IKE_SA_INIT response from the GC/KS, it knows which transforms and key material to use to protect GSA_AUTH message. In order to indicate the multicast group that it wishes to join, in accordance with [22], it includes the Group Identification Payload ID_g, using the IPv6 address as ID type. The candidate member's ID payload ID_m, is also included along with AUTH payload, so that the GC/KS can authenticate the identity of the candidate member. It is worth mentioning that all members by default are allowed to receive multicast traffic only from GC/KS. However, in IoT domain, a device running with a particular application would be preconfigured in GSPD using the directionality attribute to receive multicast traffic from any member authorized to send multicast traffic. This functionality will

not alter unless the particular member is either reconfigured or it is evicted from the group. It is vital for all the members to know the authorized senders in the multicast group in the case they have "symmetric" or "receiver-only" directionality in their GSPD. Therefore, *this thesis proposes that the candidate can request for all the Sender IDs of all the members that are authorized to send multicast traffic, by including the corresponding Notify payload in the GSA_AUTH message.* In this case, GC/KS includes, within the Sender ID payload, all the IDs of the authorized senders in the group regardless if they are active or not. The Sender ID proposed in this thesis is described in detail in Section 4.3.5.6.

The authors in [22] have approached this matter from a different perspective. The candidate members do not obtain in any way the IDs of authorized senders of multicast traffic. They use a notify payload such that a candidate can request to be a sender itself. However, in our communication scheme, it is necessary to know the valid sender IDs, which is actually the IPv6 address of the node, so that each candidate member can create source-specific GSAD entries for the particular multicast IPsec traffic. It is necessary to maintain source-specific GSAD entries so that each member can perform a local authorization check. In that way, if the source of the multicast IPsec traffic does not originate from an authorized sender, the traffic will be discarded. The format of GSA_AUTH request is shown in Table 4.3, where the payloads inside SK are considered to be encrypted and integrity protected.

HDR, SK{ ID_m, AUTH, $\mathbf{ID_g}$, [CERT], [CERTREQ], [**N**] }

Table 4.3: GSA AUTH request message format with optional Notify payload added in case the candidate requests for the Sender IDs of the group members. The payloads in brackets are optionally included in this message.

Upon reception of the GSA_AUTH request, GC/KS initiates the process to authenticate the candidate member, in order to determine if it is the party that is claiming to be, using the ID_m and $AUTH$ payloads. After successful authentication of the peer, GC/KS includes its own Identity payload, ID_s, in the GSA_AUTH response with the corresponding AUTH payload, so that the candidate member can perform the same check. Then GC/KS proceeds with GSPD lookup, in order to find out if the the policy for the requested group, indicated in ID_g, is protected by the IPsec protocols. If it is protected, GC/KS goes ahead with looking up the GPAD to find out if the particular candidate member is authorized to be part of the group. If authorization fails, GC/KS initializes an INFORMATIONAL message exchange with the candidate member to notify it that authorization failed by using the corresponding Notify payload. In successful authorization, GC/KS proceeds with searching through GSAD for the corresponding GSAT of the particular multicast group. If this cross-check is successful, GC/KS includes the GSAK with KEK key material, KD_{KEK}, and the GSAT with TEK key material, KD_{TEK}. If GSA_AUTH request contains the Notify payload requesting for the Sender IDs of the group members, GC/KS checks if there are any authorized members that can send mul-

ticast IPsec packets. If there are, the Sender ID (SID) payload is also added. *It should be noted that in order to make Group-IKEv2 more suitable for IoT devices, we altered the format of GSAK, GSAT, KD_{KEK}, KD_{TEK} and Sender ID compared with [22], as we discuss in section 4.3.5.* We have neglected fields that do not serve any purpose in Group-IKEv2 for IoT, such as the *Source Traffic Selector*, *Destination Traffic Selector*, the *Lifetime* in *GSAT*. We also have changed the significance of other payload fields such as the *Lifetime* and *Message ID* fields in *GSAK* payload. The format of GSA_AUTH response is depicted in Table 4.4.

HDR, SK{ **ID_s**, AUTH, [CERT], **GSAK**, **GSAT**, **KD_{KEK}**,**KD_{TEK}**, **[SID]** }

Table 4.4: GSA AUTH response message format.

4.3.3.2 GSA REKEY

A GSA_REKEY message, is necessary to be sent by a GC/KS under two different circumstances. First when there are membership changes within the group, meaning that either new members join the group, existing members leave the group, or if they are considered compromised and are evicted from the group. The second case is when the GSAs and key material are close to be expired and GSA_REKEY messages are sent to renew them. All types of GSAs are valid for a given period of time, defined by the network administrator. The validity duration depends on the security risks that arise within the network, along with the application requirements in the network entities.

In case of membership changes, the GC/KS has to update particular group members with either multicast, or unicast GSA_REKEY messages. In case a new member is about to join the group, GC/KS has to update the existing group members sending a multicast GSA_REKEY message, in which new *GSAK* and *GSAT* along with new KD_{KEK} and KD_{TEK} payloads are included. The format of GSA_REKEY message can be seen in Table 4.5.

HDR, SK{ **AUTH**, **GSAK**, **GSAT**, **KD_{KEK}**, **KD_{TEK}** }

Table 4.5: GSA REKEY request message format in case of membership change.

If an existing member is evicted or wishes to leave the group, GC/KS has to update the rest of the members with new *GSAK* and *GSAT*. However in that case, GC/KS will first update *GSAK* and KD_{KEK} by sending unicast or multicast GSA_REKEY messages. Then it will proceed with sending unicast or multicast GSA_REKEY messages with the new *GSAT* and KD_{TEK} using the new *GSAK* and KD_{KEK}.

Moreover, when GSAs and key material are soon to be expired, GC/KS sends multicast GSA_REKEY messages without including both GSA types and KD types in the same message. In this case, it is under the network's administrator authority to

decide how often GSA_REKEY messages should be sent to update each type of GSA and KD separately. The format of this message is shown in Table 4.6.

HDR, SK{ **AUTH**, **GSAT**, $\mathbf{KD_{TEK}}$ }

Table 4.6: GSA REKEY request message format in case of GSA expiration including only $GSAT$ and KD_{TEK}.

A security concern regarding GSA_REKEY messages is the authentication of the GC/KS by the group members. The authentication process described in IKEv2 standard in [5] Sections 2.15, refers to a pair of peers that have exchanged IDs, SPIs and Nonces during IKE_SA_INIT exchange. Then in order for the peers to authenticate each other, they create a particular text, using the aforementioned information of the other peer, and then they sign it. However, in case of GSA_REKEY messages, all the members have to authenticate GC/KS by signing the GSA_REKEY message itself instead of signing a particular text. Therefore, *this thesis in accordance with Internet Draft Group Key Management using IKEv2 [22], proposes GC/KS to use a digital signature authentication method to sign GSA_REKEY messages. In that way source authenticity is assured to all the members using either RSA, DSS Digital Signature or ECDSA as they are described in [5] Section 3.8.* The format of the new Authentication payload is described in detail in Section 4.3.5.2.

4.3.4 IKEv2 Payloads used in Group-IKEv2 for IoT

This section presents IKEv2 payloads that are involved in Group-IKEv2 messages. For compatibility reasons their format is kept the same as in IKEv2 standard specifications in [5].

4.3.4.1 IKE Header (HDR)

All messages exchanged in Group-IKEv2 begin with the IKE header payload. It's format is shown in Table 4.7.

IKE SA Member's SPI				
IKE SA GC/KS's SPI				
Next Payload	Mj Version	Mn Version	Exchange Type	Flags
Message ID				
Length				

Table 4.7: IKEv2 Header format.

- *IKE SA Member's SPI* and *IKE SA GC/KS's SPI* fields are set with unique 8 octet values, chosen by each member and GC/KS respectively, in order to identify a unique IKE Security Association during IKE_SA_INIT and GSA_AUTH message exchange. In case of a GSA_REKEY message, *our solution specifies that the field IKE SA Member's SPI is set to zero*, because all group members have different IKE SA SPIs between them and the GSA_REKEY message has to be matched and accepted by all group members. Since the same GSAK SPI has been set for all the group members and the members need to distinguish which GSAK this message refers to, *IKE SA GC/KS's SPI field is set with GSAK's SPI value.*
- *Next Payload* indicates the type of the following payload after IKE Header. The next payload types are in accordance with IKEv2 standard specifications in [5] with the addition of five new payload types shown below and described in detail later in this chapter.
  - IDg, Identification - Group
  - GSAK, Group Security Association for encrypting Keys
  - GSAT, Group Security Association for encrypting Traffic
  - KD, Key Download
  - SID, Sender ID
  - N, Notify
- *Major Version* Payload indicates the major version of IKE protocol used.
- *Minor Version* Payload indicates the minor version of IKE protocol used.
- *Exchange Type* Payload indicates the type of messages exchanged. The types that are being used, are in accordance with IKEv2 standard specifications with the addition of two new types of message exchanges shown below. The types of messages in Group-IKEv2 are:
  - GSA AUTH, 38
  - GSA REKEY, 39
- The field *Flag* specifies if the message is a response or request, and if the transmitter of the message is compatible with higher protocol major version than the one indicated in the respective field.
- *Message ID* is a 4 octet identifier used to match requests with responses, as well as controlling retransmissions.
- *Length* is the total length of the message.

4.3.4.2 Generic Payload Header

The generic payload header is used before any payload, to indicate the payload that immediately follows. The same format as in IKEv2 [5], which is included here for completeness, can be seen in Table 4.8.

Next Payload	C	Reserved	Payload Length

Table 4.8: Generic Payload Header format.

The purpose of this payload is to assist parsing Group-IKEv2 messages, by indicating which kind of data are expected to be parsed, regardless of the order that they are placed within a message.

4.3.4.3 Security Association (SA)

The security association (SA) payload is included in IKE_SA_INIT message exchange between a member and GC/KS to negotiate the security parameters, encryption, integrity, pseudo-random function, and Diffie-Hellman group, so that they can be used in GSA_AUTH message. The candidate member includes many different proposals of security parameters in the SA payload, and GC/KS upon reception of this payload selects the cryptographic suit that has the best match among the candidate member's proposals and his own. The SA payload, which is built in accordance with [5], is shown in Table 4.9 for completeness.

Generic Payload Header	Proposals

Table 4.9: SA payload format.

The SA Payload can possibly contain many different proposals from the corresponding peer. Each *Proposal* payload among other things, includes the *Proposal number*, the *Protocol ID*, the *SPI size*, the *Number of transforms* included in this proposal, and the *Transforms* that are proposed following IKEv2 standard. The exact format can be seen in Table 4.10 and it is included here for completeness of this work.

Last Substruct	Reserved	Proposal Length	
Proposal Number	Protocol ID	SPI size	Number of Transforms
SPI			
Transforms			

Table 4.10: Proposal Payload format.

- *Last Substruct* takes the value 2 if there are more proposals following or the value 0 if this is the last proposal.
- *Proposal Length* specifies the length of this proposal.
- *Proposal Number* is used to identify the proposals. The accepted proposal number must match the number of the proposal that was accepted by the peer.
- *Protocol ID* is a 1 octet identifier to specify the protocol that this Security Association is used for. In this case since SA payload is used to negotiate the IKEv2 SA that is going to be used in GSA_AUTH message, the *Protocol ID* will always be 1, indicating that it refers to IKEv2.
- *SPI Size* is 0 for the initial IKE SA INIT.
- *SPI* is obtained from IKE Header.

The Transform payload format is shown in the Table 4.11, and follows IKEv2 specifications [5].

Last Substruct	Reserved	Transform Length
Transform Type	Reserved	Transform ID
Transform Attributes		

Table 4.11: Transform payload format.

- *Last Substruct* has the same functionality as in Proposal payload.
- *Transform Length* indicates the length of the transform.
- *Transform Type* indicates one of the four types mentioned earlier. IKE proposals include the following types of transforms:
  1. Encryption algorithm, transform type 1
  2. Pseudo-random function (PRF), transform type 2
  3. Integrity algorithm, transform type 3
  4. Diffie-Hellman group, transform type 4
- *Transform ID* specifies the specific instance of transform type proposed. For example, for encryption algorithm transform the instance indicates which type of encryption algorithm is used.

Last but not least, the transform attribute is used to modify the specifications of the transform. Currently the attribute type defined by IKEv2 specification and included in this architecture is the Key Length, which is used by encryption algorithms with variable length keys.

4.3.4.4 Key Exchange Payload (KE)

The Key Exchange Payload is used in IKE_SA_INIT message exchange in order to indicate the selected Diffie-Hellman group number and provide the selected public value of each peer. The format of this payload is shown in Table 4.12. It should be mentioned that it follows IKEv2 standard and is included here for completeness.

Generic Header Payload	
Diffie-Hellman Group Number	Reserved
Key Exchange Data	

Table 4.12: Key Exchange Payload format.

- *Diffie-Hellman Group number* field identifies the Diffie-Hellman group in which the Key Exchange Data was computed. It must be the same as the one specified in a proposal included in the SA payload, in the same IKE_SA_INIT message.
- *Key Exchange Data* field contains the Diffie-Hellman public value of the participant.

4.3.4.5 Nonces

Nonces contain random data of variable length 16 to 256 octets, generated by the transmitter. The candidate member generates the Member Nonces, N_m, and the GC/KS generates the Server Nonces, N_s. It should be mentioned that Nonces must not be reused.

4.3.4.6 Identification Payload (ID)

The format of identification payload, which is in accordance with IKEv2 standard [5] Section 3.5, is shown in Table 4.13. The *ID Type* field indicates which type of ID is used for a particular identification payload. The member's identification payload, ID_m, and GC/KS identification payload, ID_s, follow this format and they are used in combination with $AUTH$ payload by each peer in GSA_AUTH messages to determine if the other peer is the one that is claiming to be.

Generic Header Payload	
ID Type	Reserved
Identification Data	

Table 4.13: Identification payload.

4.3.4.7 Certificates - Certificate Requests

Certificates are optionally used for authentication purposes. It might be efficient in terms of communication overhead, if they were omitted but security requirements should always be taken into consideration. The format of *Certificate (CERT)* payload and the *Certificate Request payload (CERTREQ)* have the format specified in IKEv2 standard [5], Sections 3.6 and 3.7 respectively.

4.3.4.8 Authentication Payload (AUTH) for GSA_AUTH messages

The authentication payload is used both in GSA_AUTH and GSA_REKEY messages. The authentication methods supported are the same as in IKEv2 standard specifications,[5] Section 3.8. However, the computation of Authentication Data field differs among the GSA_AUTH and GSA_REKEY messages in Group-IKEv2.

In GSA_AUTH messages, *Authentication Data* in either the candidate member or in GC/KS are computed as in IKEv2 standard specification, by appending the first message received by the other peer, with the Nonce of the other peer and the signed ID payload of the peer itself. It is evident that the authentication data is unique between a pair of peers, which means that it cannot be used for authentication purposes between a group of members and the GC/KS, since all the members must be able to authenticate the GC/KS using the same data. *Therefore, we modified the format of the Authentication Data of AUTH payload in GSA_REKEY messages* as it is explained in Section 4.3.5.2.

4.3.5 New Payloads for Group-IKEv2 in IoT

This section presents Group-IKEv2 new payloads and their format, proposed by this master thesis. The design of these payloads is based on the Group Key Management using IKEv2 Internet Draft, [22], with appropriate adaptations for IoT.

4.3.5.1 Group Identification Payload

The group identification payload, ID_g, has the same format as the identification payload, described in Section 4.3.4.6. It is included in GSA_AUTH request messages by the candidate members in order to indicate the multicast group they wish to register. The *ID Type* field indicates which type of ID is used for this ID. The types supported by ID_g follow Group Key Management using IKEv2 Internet Draft [22].

4.3.5.2 Authentication Payload (AUTH) for GSA_REKEY messages

The authentication payload format for GSA_REKEY messages remains the same as in IKEv2. However, the *Authentication Data* for GSA_REKEY message cannot make use of the signed octets proposed in IKEv2 standard, since these information are unique per GC/KS - member pair. On the other hand GSA_REKEY messages are sent from GC/KS to all the members and source authenticity has to be ensured. *This thesis proposes GC/KS to sign GSA_REKEY message itself using its private key and include*

it in the Authentication Data field of the Authentication payload. Then upon reception of GSA_REKEY messages, the members will be able to verify the source authenticity of the message by using the public key of GC/KS, which is distributed during registration process. The authentication methods that can be used are RSA, DSS Digital Signature, and ECDSA. The format of Authentication payload is displayed in Table 4.14. It should be noted that extensible authentication is not enabled, since it requires multiple message exchanges, a fact that would be very costly for resource constrained devices.

Generic Header Payload	
Authentication Method	Reserved
Authentication Data	

Table 4.14: Authentication payload format for GSA_REKEY messages, in which a different format of signed octets is used than in GSA_AUTH messages.

Thus, the way the Authentication payload is now calculated, enables all the existing members of a group to determine if GC/KS is the originator of GSA_REKEY message.

4.3.5.3 Group Security Association for encrypting Keys (GSAK)

This thesis proposes a group security association payload for encrypting keys (*GSAK*) which is a modified version of the corresponding payload in [22] Section 4.4. This payload can be included in both GSA_AUTH and GSA_REKEY messages, depending on the policies defined by the network administrator. The GSAK payload format is presented in Table 4.15.

GSAK SPI			
Lifetime		**Length**	
SPI Size	Reserved	**Message ID**	**Number of KEK Attrib**
KEK Attributes			

Table 4.15: The format of GSAK payload in Group-IKE for IoT. The fields in bold are the new fields defined in this work.

The fields of GSAK Payload are the following:

- The *GSAK SPI* field is common for all the group members. It is used to distinguish GSAKs of different multicast groups.
- The *Lifetime* is the validity time of the corresponding GSAK. It is set by the administrator depending on the network requirements and how often a renewal of the GSAs is needed.

- The *SPI size* is the size of the corresponding SPI. Since it refers to GSAK SPI, it is set to value 4.
- The *Length* is the total length of the payload without Generic Header Payload.
- The *Message ID* is used to determine the freshness of the received message.
- The *Number of KEK Attributes* is the number of GSAK attributes included in this payload.
- The *KEK Attributes* are group security transforms that are used by GC/KS in GSA_REKEY messages. The transforms specify:
  - KEK distribution algorithm: if a group key distribution algorithm is used, like LKH, the value of this attribute is 1. It should be noted that 0 is the reserved value.
  - KEK encryption algorithm
  - KEK integrity algorithm
  - KEK authentication method

This thesis proposes the following modifications to Group Key Management using IKEv2 Internet Draft, [22] for Group-IKEv2 for IoT.

1. Two separate payloads are considered for the Group Security Associations instead of one that was defined in [22]. The new payloads are the Keys Encrypting Keys GSA payload (*GSAK*) and the Traffic Encrypting Keys GSA payload (*GSAT*). The Internet Draft [22] distinguishes the GSA types within a single payload by using the *type* field. We have differentiated the definitions of each GSA type because the set of information required for each GSA is different and therefore they should be clearly distinguishable.

2. The *Source Traffic Selector* and *Destination Traffic Selector* fields of GSA payload are discarded. The purpose of these fields is for the network entities to share information from their SPD and GSPD to their peers. The *Source Traffic Selector* field indicates the source address range from which Group-IKEv2 traffic can be sent, whereas the *Destination Traffic Selector* field indicates the destination address range Group-IKEv2 traffic is destined to. In our design, the source address of the multicast Group-IKEv2 traffic is the IPv6 address of GC/KS. GC/KS is considered by default the only entity from which multicast Group-IKEv2 traffic can originate. It is automatically set as the source IPv6 address of the Group-IKEv2 session information upon reception of GSAK payload. The *Destination Traffic Selector* for multicast traffic is always the multicast IPv6 address of the group. In our solution, both *Traffic Selectors* types are neglected since the senders and receivers of multicast Group-IKEv2 traffic are not negotiable between GC/KS and a candidate member. Upon registration, the candidate member sets the GC/KS's

IPv6 address as the default originator and the multicast IPv6 address as the default destination of the multicast traffic. By neglecting the Traffic Selector fields, we reduced significantly the total size of *GSAK* payload.

3. The *Lifetime* field is considered as an independent field from the KEK attributes, whereas in [22] it was considered as a KEK attribute. This field indicates how long the particular GSAK is valid. It must be always included in *GSAK* payload in order for the members to be aware when the GSAs are going to be expired and act accordingly. Once the lifetime expires, the member has to reinitiate registration to the group. If the *Lifetime* field was considered a KEK attribute as in [22], it could possibly not be included in the payload. Thus, in case the members did not receive any GSA_REKEY message to update the GSAs and keys, upon lifetime expiration they would not know that they have to reinitiate registration to the group and eventually they would be unwillingly evicted.

4. Unlike [22], the *Message ID* field is considered as an independent field from the KEK attributes. The reason for including this feature as an independent field is that it must be always included to determine the freshness of the particular GSA. If this attribute is not set, freshness of the GSAK cannot be determined, providing no replay attack protection against adversaries.

5. The *Length* field is added in order to check the length of the received GSAK. In that way the recipients of this payload can determine if it is properly received.

6. The *SPI Size* field is added in order to check if the SPI is correctly set. This is the same method as in IKEv2 standard, in which the SPI Size is also checked in SA proposals.

7. The *Number of KEK attributes* field is added, indicating the number of attributes that are included in *GSAK* payload. In that way the payload is parsed in a similar way as the transform payload is parsed in an SA proposal in IKEv2 standard.

4.3.5.4 Group Security Associations for encrypting Traffic (GSAT)

The *GSAT* payload contains the groups security associations destined to be used by the IPsec protocols, Authentication Header (AH) and Encapsulating Security Payload (ESP). It is included in Group-IKEv2 messages, GSA_AUTH, and GSA_REKEY. The payload format can be seen in Table 4.16.

- The *Protocol ID* field indicates the IPsec protocol in which the GSA is going to be used: Authentication Header or Encapsulating Security Payload.
- The *GSAT SPI* is used to index the GSAD entry of the particular multicast group.
- The *Length* is the total length of the GSAT payload.

GSAT SPI		
Length		
SPI Size	Protocol ID	**Number of TEK attrib**
TEK Attributes		

Table 4.16: GSAT payload format.

- The *SPI Size* is the size of the corresponding SPI. Since it refers to GSAT SPI, it is set to value 8.
- The *Number of TEK attributes* is the number of GSAT attributes that are included in GSAT payload.
- The *TEK attributes* are the group security transforms that are set, in order to be used by the members in IPsec protocols. The transforms specify:
  - The TEK encryption algorithm.
  - The TEK integrity algorithm (if needed).
  - The Key length for encryption algorithms that use variable key length.

In order to reduce as much as possible the size of the messages, we have modified the payload proposed in [22], Section 4.5. The modifications we have done for Group-IKEv2 for IoT are the following:

1. *GSAT* is independently defined from *GSAK* payload for the reasons explained in the previous section.
2. In contrast with [22], the *Source Traffic Selector* and *Destination Traffic Selectors* are not included in GSAT payload. The *Source Traffic Selector* indicates the source of the multicast IPsec traffic. The GC/KS and any valid sender within the group can be the source of multicast IPsec traffic. In our design, the IPv6 address of the GC/KS is considered the default source IPv6 address that is set in new GSAD entries. In order for the GC/KS to share the IPv6 addresses of all the authorized senders with all the members of the group, it would have to include multiple *Source Traffic Selector* payloads. However, this would significantly increase the size of *GSAT* payload, which is not desired. Therefore, we decided to create a payload, called *Sender ID (SID)*, in order to include all the IPv6 addresses of all the authorized senders of the group, as it is discussed in Section 4.3.5.6. The *Destination Traffic Selector* indicates the destination of the traffic, which is always the multicast IPv6 address. A member upon reception of *GSAT* payload creates a GSAD entry for the corresponding group, setting the destination IPv6 address to the multicast IPv6 address it requested to register. The purpose of Traffic Selectors

in IKEv2 standard is to negotiate the address range of the source and destination of the traffic. In this design, there is no negotiation between the candidate members and the GC/KS since the GC/KS as a central authority, provides all the GSAs and key material for the requested group and the senders and receivers of multicast traffic are automatically set upon reception of *SID* and *GSAT* payloads. Any risk for GSAD misconfiguration due to disregarding the traffic selector fields is eliminated because the GSATs are indexed with unique *GSAT SPIs*. Thus, even if there are groups within the network that have the same authorized senders, the GSATs are not mixed up among the groups, because of the unique SPIs.

3. The *TEK attributes* field includes all the TEK transforms that are going to be used in multicast IPsec traffic. The attribute Life Duration is not considered, since the lifetime of a particular GSAT is determined by the lifetime of the corresponding GSAK of the group. In particular, once the lifetime of a specific GSAK is expired, a GSA_REKEY message is sent to renew the GSAK. Once this is done, it is necessary to send another GSA_REKEY message in order to renew the GSAT of the particular group. In that way it is assured that the multicast group is secured in case of membership changes within the group.

4. *Length* field is added to cross-check the length of the received GSAT.

5. *SPI Size* field is added to check if the *GSAT SPI* has been set correctly such that GSAD misconfigurations are avoided.

6. The *Number of TEK attributes* is added to indicate how many TEK attributes are expected while parsing this payload. In that way we use a compliant method to IKEv2 standard for parsing the transforms in an SA proposal.

4.3.5.5 Key Download (KD)

The key download payload includes key material either for keys encrypting keys (KEKs), or keys encrypting data traffic (TEKs). These two types are distinguishable by using specific key attributes that indicate the type of traffic that each key refers to. If either *GSAT*, or *GSAK*, or both payloads are included, then the corresponding key material KD_{TEK} or KD_{KEK} must be included as separate payloads using the corresponding key attributes. In that way the key material is clearly distinguishable for each type of GSA avoiding any misconfiguration in GSAD and in Group-IKEv2 session information. Key Download payload format is shown in Table 4.17.

- The *SPI* field indicates the SPI of the corresponding GSA, that this KD payload is referring to.
- The *Length* is the total length of KD payload without the Generic Payload Header.
- The *Number of Keys* is the total number of keys included in this payload.

SPI	
Length	Number of Keys
Key Attributes	

Table 4.17: KD payload format.

- The *Key Attributes*, contain the actual key in the *Key Attribute Value* field and indicate the type of the key in *Key Attribute Type* field. The Key Attribute Types are the following:
  - KEK_ENRC_KEY type with value 1.
  - KEK_INTEG_KEY type with value 2.
  - KEK_AUTH_KEY type with value 3.
  - TEK_ENCR_KEY type with value 4.
  - TEK_INTEG_KEY type with value 5.

The format of this payload is a modified version of KD payload of [22], in which the following amendments have been made:

1. Either KEK or TEK type of key material can be included in a single KD payload, whereas in [22] all types of key material are appended in the same payload and they are distinguished using the KD type field. We have discarded the KD type field, and we have differentiated KEK from TEK key material using the *Key Attribute Type* field. In that way we discarded one field (KD type) and reduced the size of the payload, while at the same time made distinguishable the different KD payloads.

2. The SPI field indicates the corresponding GSA that the particular key material refers to. We specify if the key material is destined to be used as KEK or TEK by checking the SPI included, and also by checking the Key Attribute types since they specify if the keys are used by GSAK or GSAT. We have chosen this way for including the keys, so that we distinguish the group keys as much as possible from each other and make them independent. In that way a possible error in the creation of one of these payloads will not affect the other.

3. Unlike [22], we haven't included any payload for LKH array, since it is out of the scope of this thesis to implement an group key distribution algorithm.

4.3.5.6 Sender ID payload (SID)

The sender ID payload is included in GSA_AUTH response messages upon request of the candidate members, in order to provide them with all the sender IDs of the existing group members that are authorized to send multicast IPsec traffic to the group. As

soon as the candidate member receives this payload, updates its GSAD entries, so that incoming IPsec traffic from these senders is accepted.

The format of Sender ID payload is depicted in Table 4.18.

SPI	
Length	Number of Sender IDs
SID Attributes	

Table 4.18: SID payload format.

- The *SPI* field is the same as *GSAT SPI* included in the same message.
- The *Length* field is the total length of this payload without the Generic Header Payload.
- The *Number of Sender IDs* field is the number of Sender IDs that are included in this payload.
- The *SID Attributes* field contain all the valid Sender IDs in IPv6 Address format.

The sender ID payload is included only in GSA AUTH and not in multicast GSA_REKEY messages, since the purpose of GSA_REKEY messages is only to update the existing Group Security Associations and not to create completely new ones.

The purpose of this payload in this thesis is different from Group Key Management using IKEv2 Internet Draft [22], in which the sender ID was a unique number that was provided to the authorized senders and it was used for key derivation purposes when counter-based mode transform was used. Since key derivation is out of the scope of this work, we assumed that the senders have unique IV parts when counter-based encryption is selected.

The security implication that arises in Group-IKEv2 when sharing the authorized sender IDs is that if a member is compromised by an adversary, all the IPv6 addresses of all the authorized senders within the group will be known, fact that would expose group memberships.

4.3.6 Key Distribution and Update in Group-IKEv2 for IoT

The group security associations for encrypting Keys (GSAKs) include an optional transform to indicate if a group key distribution algorithm is used. Such an algorithm is responsible for organizing and distributing the key material to the group members upon membership changes by sending the least possible number of messages.

In the proposed architecture this option is set by default to be none. In particular, upon member registration a multicast GSA_REKEY message is sent to the existing members to ensure backward secrecy before the registration of a new member is completed,

and upon member eviction, unicast GSA_REKEY messages are sent in order to ensure forward secrecy. Unicast GSA_REKEY messages are encrypted and integrity protected using pairwise keys, shared between each member and GC/KS, which are distributed upon registration of each member.

Figure 4.4 shows an example of a candidate member initializing the registration process. Initially the candidate member and the GC/KS exchange IKE_SA_INIT messages to exchange nonces and Diffie-Hellman group and negotiate the transforms that are going to be used in GSA_AUTH message. Upon reception of IKE_SA_INIT response, the candidate member sends a GSA_AUTH request message to authenticate the identity of the GC/KS and most importantly to request registration to a particular multicast group. Upon reception of such request, GC/KS checks if the candidate member's identity is valid and if it is authorized to be part of the group. Then, if both checks are successful, it checks if there are any active members within the group. If there are, it sends a multicast GSA_REKEY message, encrypted with the current GSAK, to update with new GSAK and GSAT along with their corresponding key material the existing group members . Once the message is sent, GC/KS sends the GSA_AUTH response to the candidate member providing the new GSAK, the new KEK group key, the individual KEK key, GSAT with the corresponding TEK group key material, and all the authorized sender IDs of the group.

However, when there are many membership changes in the group, in which members are often evicted or leave the group, the use of a proper and more efficient group key distribution algorithm is preferred compared with separately sending unicast GSA_REKEY messages to each group member. In that way less GSA_REKEY messages are sent, creating the minimum communication overhead possible. As we have already mentioned in Chapter 3, there are many different algorithms proposed in literature, among which Logical key hierarchy (LKH) and OFT are mostly preferred.

This thesis suggests LKH as the group key distribution algorithm to be combined with Group-IKEv2 for IoT. It is based on the hierarchical tree approach provided in [7], Section 5.4. The keys are organized in such a way that they form a hierarchical tree. Starting from the bottom, the leaves represent pairwise keys shared between each member and GC/KS. The root of the tree is considered as the KEK group key, which is the same for all the devices within the group. The intermediate nodes of the tree constitute auxiliary keys that are used for encrypting rekeying messages addressed only to specific group members that are under this node in the tree. Each member stores its individual key, the group key, and the auxiliary keys that are within the path from themselves to the root of the tree. All the set of keys are assigned and distributed in a pairwise manner to all members by the GC/KS. In the event of a group membership change, GC/KS updates the keys in such a way that forward and backward secrecy are ensured (see Section 5.4.1 in [7]). The update of the keys can combine unicast and multicast rekeying messages, keeping Rekey messages as least as possible.

All the membership changes are detected and mostly triggered by GC/KS, which is responsible for deciding if a network entity can join the security group, or if it needs to leave the security group. These decisions are based on eviction and join group poli-

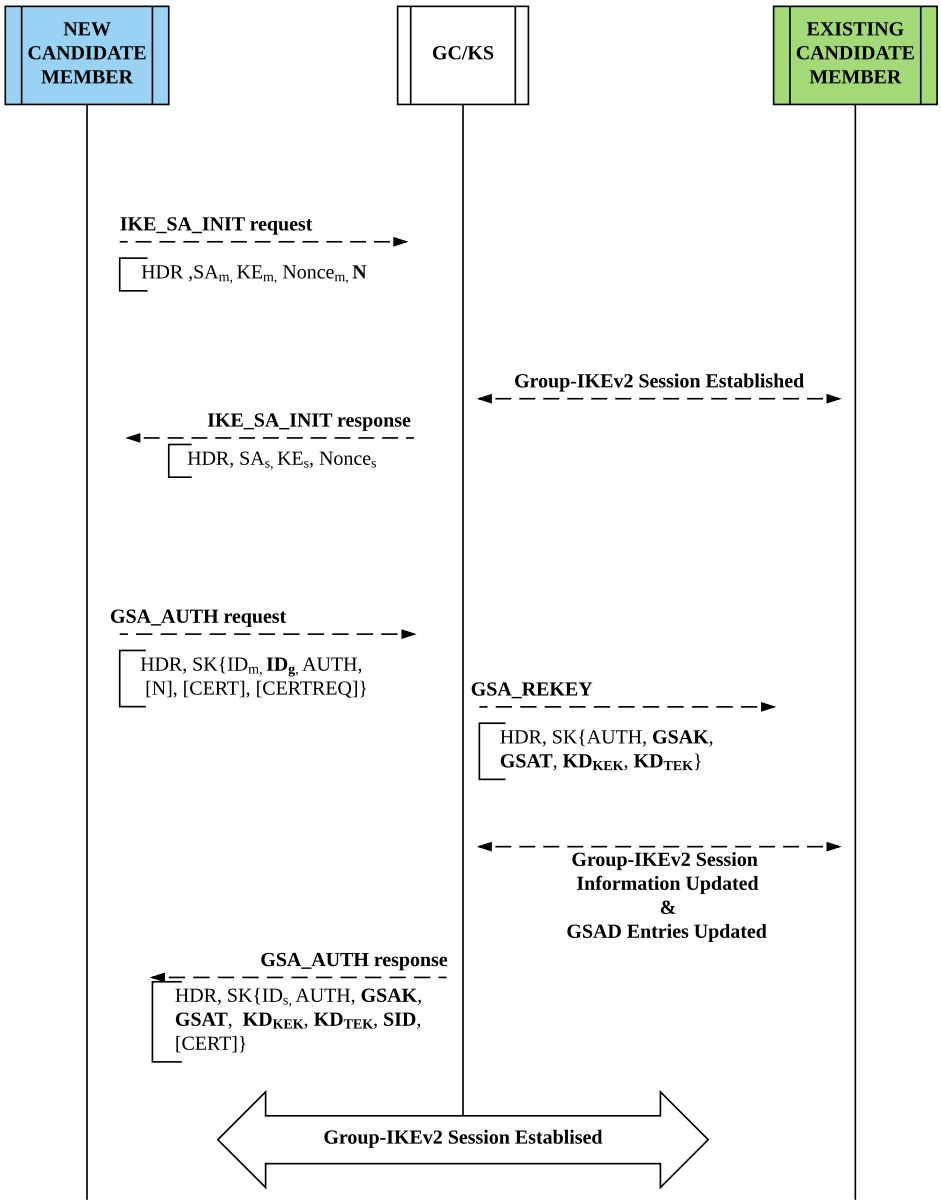

Figure 4.4: Group-IKEv2 process. A new candidate member requests for GSAT and TEK KD for multicast IPsec. GC/KS has to first update the existing group members with new GSAs and key material, before providing them to the new member in order to ensure backward secrecy.

cies that are predefined by the network administrator in GC/KS. The following section

presents the Group-IKEv2 policies that we propose in this thesis work.

4.4 Group-IKEv2 Policies

Depending on the type of applications running on top of the network, the system administrator has to define specific policies that should be applied in GC/KS and all the group members. These policies can be classified to system policies and group policies. The first class of policies specifies system settings that should be globally applied to all devices in the network during configuration process, regardless of their type, whereas the second class specifies policies applied by GC/KS to the group members.

The system policies among other things, define the time that the operating system of all the devices has to wait upon reception of new GSAs, either GSAT or GSAK, in order to update its Group-IKE session parameters and GSAD entries, called update time delay [29]. The update time delay has to be set such that the longest GSA coherence is achieved among the network devices. For instance, GC/KS has to delay the update of its Group-IKE session with the new GSAK, until all the members within the Group have received the GSA_REKEY message.

The group policies are configured in GSPD of GC/KS by the network administrator so that GC/KS has absolute control of the group memberships, GSAT and GSAK updates, and membership changes. Depending on the applications running on top of the network, and the security risks that arise because of them, the administrator decides upon the validity duration of GSAKs. Upon expiration of their validity time, all GSAD entries and Group-IKE session information are erased and the members have to initiate group registration process as described earlier in this chapter. Due to the validity expiration, the time to next GSA update is also defined by the administrator in order to refresh the GSAs and prolong the lifetime of the security group. Moreover, group polices define the payloads that must be included in GSA_REKEY messages under different occasions. For instance if a new node wishes to join the group, GC/KS provides to the existing group members both the GSAK and GSAT in the same GSA_REKEY message. However, this must not be done if a member wishes to leave or is evicted from the group. In that case GC/KS has to send a GSA_REKEY message to the rest of the group members with the new GSAK and key material, and then provide the new GSAT and key material in a separate GSA_REKEY message, using the new GSAK and KEK key material to protect it. In that way the leaving node is excluded from the rekeying process itself and forward secrecy is preserved.

One of the most important tasks of the network administrator is to investigate the potential security breaches in the group, and depending on them to define group policies to prevent security compromises by adversaries in the group. Assuming that external network security risks are eliminated, all the potential security risks lie among the trustworthiness of the group members. Therefore, the network administrator has to use mechanisms that recognise malicious behaviour of a member and define policies to evict that member from the group. One way to achieve that is by using an Intrusion detection system (IDS) to monitor traffic. An IDS for IoT is already proposed by Raza in [32].

Using such a system it is possible to detect if a member tries to flood the network with multicast IPsec traffic, i.e., DDoS attack. In order to mitigate such an attack, GC/KS has to evict the member by updating the rest of the group with unicast GSA_REKEY messages with *GSAT*, *GSAK* payloads and their corresponding keys included in separate GSA_REKEY messages. This can also be achieved by sending multicast GSA_REKEY messages to subgroup members using a group key distribution algorithm such as LKH. In that way forward secrecy is ensured, since the evicted node would not be able to further participate in group communication. However, detection of malicious behaviour is quite complicated and costly, and is out of the scope of this thesis. Another aspect related to membership changes is when a candidate member joins the security group. In that case backward secrecy has to be ensured by updating all the existing group members with new *GSAT*, *GSAK*, KD_{KEK}, and KD_{TEK} payloads in a multicast GSA_REKEY message. Such an example is shown in Figure 4.4.

The following Chapter presents how this architecture is implemented and integrated in Contiki OS for Openmote-CC2538 platform.

5

Implementation

In this chapter the implementation of Multicast IPsec and Group-IKEv2 for IoT is presented, delineating the features of our proposed architecture that was implemented, for the purpose of evaluating its performance in resource-constrained devices.

In the first section we present the hardware platform used in the implementation for the purposes of this work, discussing its main characteristics. In Section 5.2 we provide a comprehensive description of how our implementation makes use of uIP/TCP stack of Contiki, so that there is a better understanding on how Multicast IPsec and Group-IKEv2 work. In Sections 5.3 and 5.4, we provide the general logic behind our implementation along with some fundamental features of Multicast IPsec and Group-IKEv2 for IoT. Lastly in Section 5.5, we discuss the limitations and difficulties that we confronted during the implementation phase.

5.1 Hardware Platform

The target hardware platform selected for this implementation and depicted in Figure 5.1 is Openmote. Openmote is an open hardware platform designed to efficiently support standards in the IoT domain. Openmote belongs to Texas Instruments SoC family, it uses a 32-bit Cortex micro-processor with a clock operating up to 32MHz. It has 32kB RAM and it can store up to 512kB in internal Flash memory. The radio is compatible with IEEE 802.15.4 standards, operating in 2GHz band, and contains a 32MHz crystal clock which is turned off when the radio is in sleep mode. There is an additional crystal clock working at 32kHz used as real time clock of the micro-controller to keep track of the time, even when the mote is in sleep mode. For the purposes of this thesis we used Openmote-CC2538 platform connected to Open-USB [36] to upload, test, and debug our implementation.

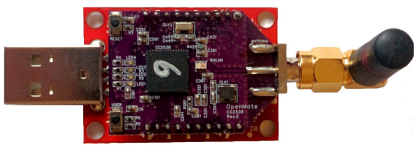

Figure 5.1: Openmote-CC2538 with Openmote-USB.

5.2 Contiki Stack

Our proposed architecture is implemented in C and is based on Contiki 3.0 Operating system (OS). Contiki is an open-source operating system destined to operate in various platforms with resource constrained characteristics. The network entities are configured with Contiki OS using the correspondence of IP stack for resource-constrained devices. Figure 5.2 shows uIP/TCP Contiki Stack, which is running on top of the network entities of our system. As it can be seen in the figure, IEEE802.15.4 is used in the physical layer. It is responsible for transmitting the data and detecting if the transmission is successful, or if a collision occurred, without including any retransmission mechanism. IEEE 802.15.4 MAC is used in MAC layer and is responsible for creating the frames of data that are to be transmitted, or parsing the frames and passing the data to higher layers of the stack. 6LoWPAN works as an adaptation layer between the MAC and the network layer, enabling context-aware header compression and fragmentation. uIP lies with Multicast IPsec on top of 6LoWPAN in the network layer. UDP is used in the transport layer and Group-IKEv2 lies on top of it in the application layer. In our implementation, the application layer is responsible for retransmissions of messages only when it is needed. We integrated Multicast IPsec and Group-IKEv2 to IPsec and IKEv2 implementation of Contiki 3.0 provided by SICS, in order to enable new capabilities.

For the purposes of this thesis work we extended udp-example of Contiki 3.0. In particular, we extended udp-client.c and udp-server.c files such that a udp-client can operate as a candidate member and a udp-server can operate as GC/KS or as a candidate member wishing to receive multicast IPsec messages. For simplicity and portability reasons, IPsec and Multicast IPsec are included as applications in the project's Makefile. In that way it is easier to implement and debug the code, but also to enable IPsec and Multicast IPsec in any other Contiki project without interfering with core files of Contiki OS. Udp-client.c file consists of a main process, called, udp-client process at which the node is configured with a global, a link-local, and a multicast IPv6 address. Moreover, the node is scheduled to transmit every 15 sec "Hello" messages to the multicast group FF1::ABCD through port 3001, while at the same time listens to port 3000 for incoming messages. Udp-server.c file consists of the process called udp-server process, in which it is configured with global, link-local, and multicast IPv6 addresses, and is enabled to listen to port 3000 for incoming messages.

Both udp-client and udp-server processes use a set of processes, which are running

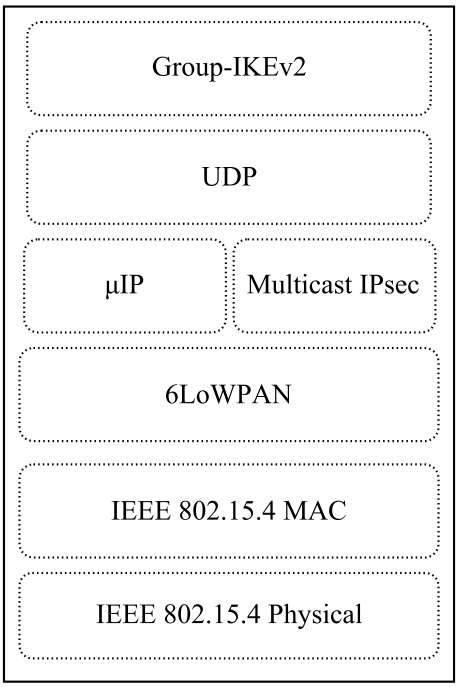

Figure 5.2: Contiki Stack with Multicast IPsec and Group-IKEv2 for IoT.

in the background and are part of the Contiki Stack, in order to enable connectivity with the multicast group. In particular, when udp-client wishes to send a multicast message with IPsec disabled, uIP6 process creates a packet with the data from application layer and then forwards it to the lower layers of the stack. However when IPsec is enabled, uIP6 uses ESP for encapsulating the original message, before passing it to the corresponding layer of the stack. For that reason the IPsec entity takes over to find the corresponding policies and the GSAs of the particular multicast address. If the policy found in GSPD indicates that traffic must be protected and no GSAs exist in GSAD, then Group-IKEv2 process is initiated. In turn, Group-IKEv2 process of udp-client communicates with Group-IKEv2 process of GC/KS using port 500 to obtain the GSAs and keys for the particular multicast group. Then Group-IKEv2 handshake is performed between udp-client (candidate member) and udp-server (GC/KS). After successful Group-IKEv2 handshake, Group-IKEv2 process passes the GSAs and keys to IPsec process of udp-client. ESP then is able to encapsulate the data into a packet which is then forwarded to lower stack layers by uIP6. The packet is then transmitted to the multicast group and the udp-client is able to continue transmitting multicast IPsec messages. Figure 5.3

visualizes the communication among the aforementioned processes.

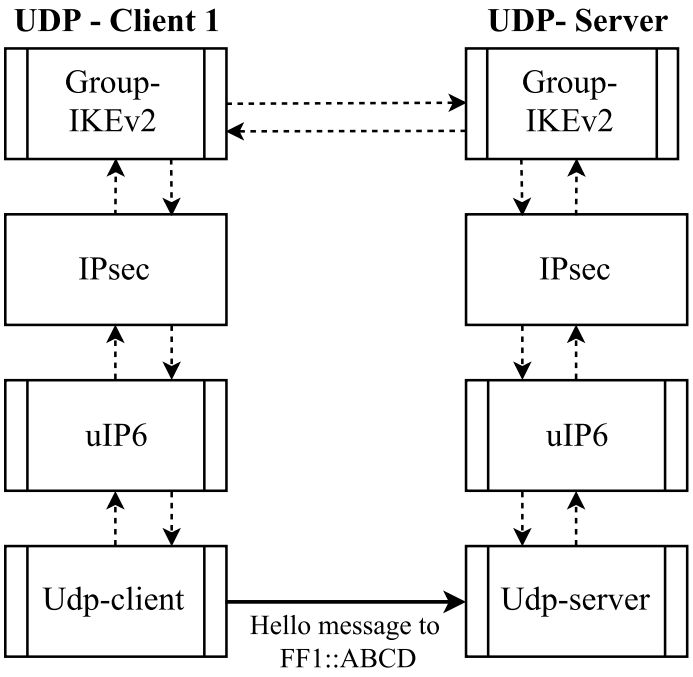

Figure 5.3: The communication between the udp client and udp server processes used in Contiki.

In the following sections we describe in detail how Multicast IPsec and Group-IKEv2 are implemented within this master thesis work.

5.3 Multicast IPsec in Contiki

This section provides a more thorough look into how IPsec was extended to support multicast traffic. It should be mentioned that each node is already configured with multicast IPv6 address, being able to receive multicast packets in port 3000 in uIP6 process and in case IPsec is enabled, the IPsec entity and uIP6 process work together in Contiki Stack. In the provided IPsec implementation, different set of functions are used for policy lookup in GSPD, depending on the directionality of the traffic, incoming or outgoing. In our implementation we preserved this notion as it can be seen in Figure 5.4. When IPsec incoming traffic enters the uIP6 interface of udp-client, i.e., candidate member, the IPsec entity takes over to perform GSPD lookup using the function set in filter.c file. If the traffic is ICMPv6, IKEv2, or Group-IKEv2, then Bypass policy is applied. If there is no defined policy the traffic is discarded, whereas if it is protected, then GSAD lookup is performed. If there is an incoming GSAD entry, then the traffic is decrypted and passed to the application layer of Contiki Stack. If there is no GSAD entry then IPsec checks the destination of the traffic. If the destination is unicast then the traffic is dropped by uIP process in accordance with [6], whereas if it is multicast Group-IKEv2 is invoked in accordance with [29], Section 4.1.1.

When outgoing traffic enters uIP6 interface of udp-client, then the IPsec entity performs GSPD lookup using the function set in spd.c file. In a similar manner as in incoming traffic, either "bypass", "discard" or "protect" policies can be applied. If the traffic is protected then GSAD lookup is performed. If a GSAD entry is found then the traffic is encrypted by ESP and passed to lower layers of Contiki Stack. In case no outgoing GSAD entry is found then IPsec entity examines the destination IP address of the traffic. If the traffic is unicast then IKEv2 is invoked, whereas if it is multicast Group-IKEv2 is invoked.

It should be mentioned that the directionality in GSPD entries of particular multicast IPsec traffic determines when Group-IKEv2 is invoked. If a GSPD entry is found with "symmetric" directionality for a particular multicast IPsec traffic, it means that Group-IKEv2 will be invoked either if the particular network entity receives or wishes to send multicast traffic and wishes to obtain the corresponding GSAs and keys from the GC/KS.

As it can be seen in Figure 5.4 there is no GPAD notion in the implementation of udp-client IPsec entity. The reason for this decision is that we considered only one GC/KS in our experiment setup, and there is no need to store additional information of hypothetical GC/KSs. Thus for simplicity reasons, we configured the udp-clients to automatically select GC/KS without additional processing. However, the notion of GPAD is existent in GC/KS since it is essential for GC/KS to identify the authorized candidate members and authenticate them during Group-IKEv2 handshake, before they become part of the group. GPAD among other things maintains the group IDs, ID_g and their corresponding authorized members' IDs, information about GSAKs and GSATs, and the valid Sender IDs in the group. During GSA_AUTH message exchange GC/KS uses the information provided in GPAD in order to determine if the candidate member

requesting for registration is authorized to be part of the group.

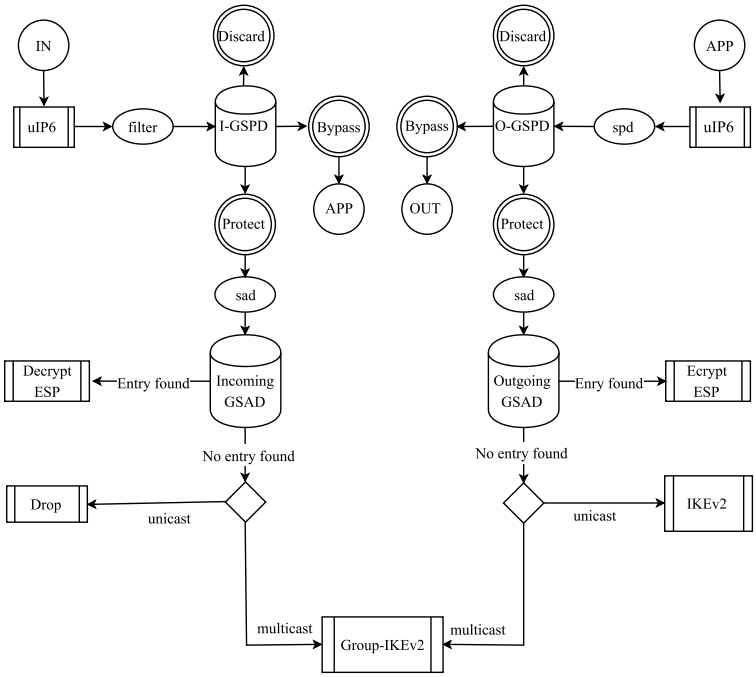

Figure 5.4: IPsec entity of udp-client for both unicast and multicast traffic.

It should be also noted that our implementation supports only specific source GSAD entries. This means that for incoming Multicast IPsec traffic, the IPsec entity is looking through incoming GSAD for the longest match, including the SPI, destination, and source address of the message. In that way a multicast IPsec packet is not accepted if it is originated from an invalid sender, enhancing the security of the system, with the cost of higher memory utilization in the devices.

5.4 Group-IKEv2 in Contiki

The Group-IKEv2 features implemented within the purposes of this thesis are the following:

1. Joining procedure of a candidate member.
2. Establishment of GSAs and key material, which enables secure group communication using multicast IPsec.
3. Periodic distribution of new GSAs and key material by sending multicast GSA_REKEY messages to the group members.
4. Distribution of GSAs and key material to the existing group members with multicast GSA_REKEY message, before a new member is registered to the group, ensuring backward secrecy.

5.4.1 IKEv2 Implementation Components

The IKEv2 implementation provided by SICS, operates in accordance with RFC7296 [5], with two types of messages implemented, IKE_SA_INIT and IKE_AUTH, and without any rekey mechanism for IKE SAs and Child SAs. The message exchanges and message retransmissions in IKEv2 are implemented with mealy state machines. This type of state machine consists of states and transitions. Each state represents the reception of a specific type of message and each transition represents a transmission of a specific type of message. There are three mealy state machines defined in IKEv2 implementation, Initiator-machine, Responder-machine, and Established-machine. The purpose of the initiator mealy state machine is to carry out the actions that the original initiator of the IKEv2 handshake should take. Equivalently, the purpose of the responder mealy state machine is to carry out the actions that the responder should take. Finally, the established machine carries out the actions of both the initiator and the responder once the IKEv2 session is established. Each mealy state machine contains unique states and transitions, which realize the actions that should be taken by each entity. The state and transition phases of each entity are stored in the corresponding IKEv2 session information, since they are specific session related. Upon successful completion of IKEv2 handshake, IKEv2 session information are erased in both the client and the server.

5.4.2 Group-IKEv2 Implementation Components

Group-IKEv2 is integrated to IKEv2 implementation in order to enable the ability in IKEv2 to distribute GSAs and keys to a group of nodes. Figure 5.5 shows the visual representation of Group-IKEv2 and IKEv2 implementation components integrated together.

As it can be seen in Figure 5.5, we created a parallel set of mealy state machines for Group-IKEv2, which uses some common components with IKEv2 mealy state

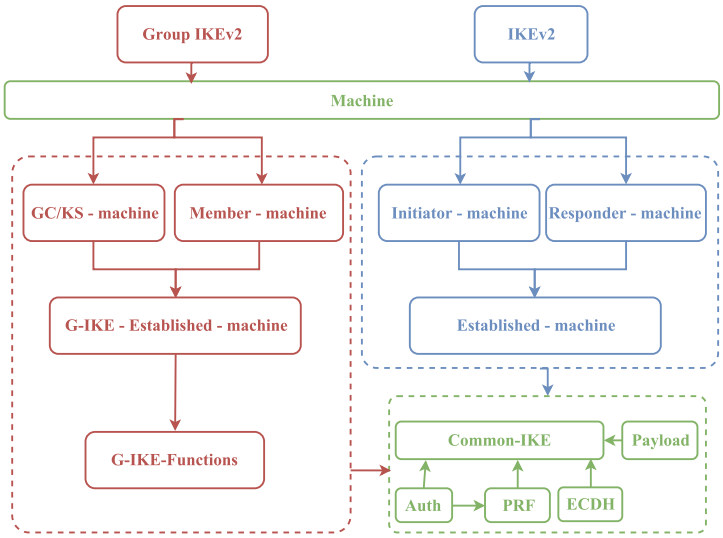

Figure 5.5: Design overview of Group-IKEv2 implementation integrated with IKEv2 implementation. The red components exclusively belong to Group-IKEv2 implementation and the blue belong exclusively to IKEv2, whereas the green components are common for both.

machines. Equivalently to IKEv2, there are three different state machines, Member-machine, GC/KS-machine and G-IKE-Established-machine, each one of them serving the corresponding role in Group-IKE handshake. It should be mentioned that the components coloured in green are used by both IKEv2 and Group-IKEv2, the components coloured in blue are used only by IKEv2, and components coloured in red are used only by Group-IKEv2. The purpose of each component is listed below.

1. The **Machine** component declares functions, which are used by all the mealy state machines of IKEv2 and Group-IKEv2. It enables the execution of states and transitions in a mealy state machine.

2. The **Member-machine** contains the states and transitions that a candidate member carries out during Group-IKEv2 handshake. Section 5.4.4 provides a detailed description of this machine.

3. The **GC/KS-machine** includes the states and transitions that GC/KS carries out during Group-IKEv2 handshake. Section 5.4.3 describes in detail the actions taken by this machine.

4. The **G-IKE-Established-machine** declares the functions that are used by both a candidate member and GC/KS once the Group-IKEv2 session is established.

5. The **Common-IKE** component declares functions that are used by all the mealy

state machines of both IKEv2 and Group-IKEv2. It enables parsing and creation of particular payloads that both IKEv2 and Group-IKEv2 use.

6. The **G-IKE-Functions** component contains additional functions for Group-IKEv2, which are related to creation and parsing of new payloads and messages discussed in Sections 4.3.3 and 4.3.5.

7. The **Payload** component contains all the payloads defined for IKEv2 and Group IKEv2 protocols.

8. The **Auth** component contains functions related to the authentication mechanism used in IKE_AUTH and GSA_AUTH messages.

9. The **PRF** component defines pseudo-random functions which are used by Auth component during authentication.

10. The **ECDH** component includes functions related to signature generation for Certificates.

5.4.3 GC/KS Mealy State Machine

The way Group-IKEv2 is invoked in udp-server is different than in udp-client. Figure 5.6 depicts the logic behind the GC/KS mealy state machine. The udp-server initiates IKEv2 process upon reception of IKE_SA_INIT message. In such event, udp-server enters IKEv2 Responder state machine of IKEv2 and parses the first IKE_SA_INIT request message using the state IKE_SA_INIT request wait. If a Notify payload is included in the message, indicating that this request concerns member registration with Group-IKEv2, the server initiates GC/KS mealy state machine and enters the transition to send IKE_SA_INIT response. A retransmission timer is set to rerun the transition to send IKE_SA_INIT response in case GSA_AUTH request is not received before the timer expires. Then the server enters GSA_AUTH request wait state. Upon reception of GSA_AUTH request, the server parses the payloads included in the message. After parsing ID_g payload, which indicates the multicast group the candidate wishes to join, GC/KS performs GPAD lookup to find out if the candidate is authorized to be part of the group. If it is not authorized, then a single Notify message is sent to the candidate, indicating Authorization Failure, and IKEv2 session information are erased from the server. If authorization is successful, then GC/KS proceeds with the authentication of the member using the Identification, ID_m and AUTH payloads. In case authentication is not successful, a single Notify message is sent to the client indicating Authentication Failure. In case of successful authentication, GC/KS enters transition to send GSA_AUTH response, and creates a GSA_AUTH message including the $GSAK$, $GSAT$, KD_{KEK}, and KD_{TEK}, along with all the valid Sender IDs, SID, for the requested multicast group. Then it stores in Group-IKEv2 session information data structure, the information of the newly registered member of the group, and updates its current active GSAD entries.

When GSA_AUTH response is sent to the first member requested registration to the group, a rekey timer is set. Upon expiration of this timer, a rekey event is triggered

in Group-IKEv2 process of GC/KS. In this event a GSA_REKEY message is created and sent by GC/KS to the whole multicast group, in order to update the $GSAT$ and KD_{TEK}. After successful transmission of this message the rekey timer is reset. If GC/KS receives a new member registration request for the same multicast group, the timer will stop. Upon reception and successful parsing of the new GSA_AUTH request, GC/KS always checks if there are currently active members in the group. In case there are, GC/KS is obligated to send a multicast GSA_REKEY message to the active members in order to update the GSAK, GSAT, and their corresponding key material before registration of the new member is completed. In that way backward secrecy is ensured. Right after GSA_REKEY message is sent, GC/KS proceeds with the creation and transmission of GSA_AUTH message to the new member.

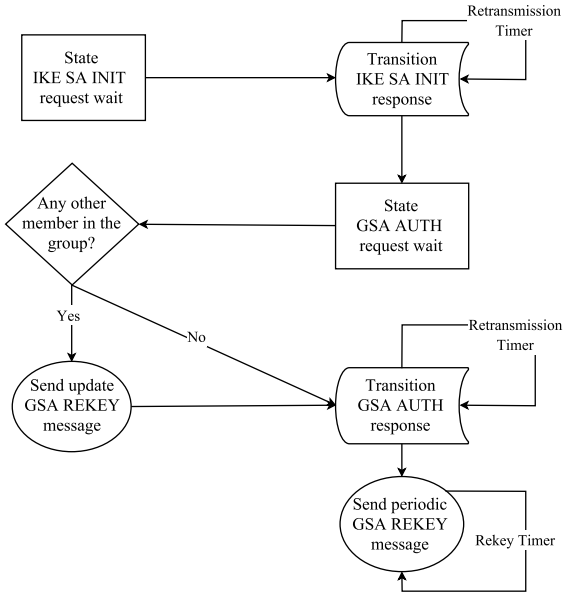

Figure 5.6: GC/KS mealy state machine of Group-IKEv2.

5.4.4 Member Mealy State Machine

In Figure 5.7 we can see the Member mealy state machine. When Group-IKEv2 is invoked in udp-client, Group-IKEv2 process enters transition to send IKE_SA_INIT request to GC/KS, and automatically after the message transmission a retransmission timer for IKE_SA_INIT message is set. Then the client enters in IKE_SA_INIT response wait state. If the retransmission timer is expired before an IKE_SA_INIT response message is received, then the client process enters again the transition to send IKE_SA_INIT request. Upon reception of IKE_SA_INIT message, the mealy state machine enters the transition to send GSA_AUTH request to GC/KS. In a similar manner, a retransmission timer is set upon completion of message transmission. Then the client

enters in GSA_AUTH response wait state. If no GSA_AUTH message is received before the retransmission timer expires, then the state machine repeats the transition to send GSA_AUTH request. If mutual authentication and member authorization are successfully completed after GSA_AUTH exchange, the client is successfully registered in the group and enters G-IKE-established machine for a given period of time, since Group-IKEv2 session is now established. Being in this state machine, the client is able to receive retransmitted GSA_AUTH messages that concern this particular session. The client is now able to store the received $GSAK$ and KD_{KEK} payload data to Group-IKEv2 session information and the received $GSAT$ and KD_{TEK} payload data to incoming and outgoing GSAD. Upon reception of SID payload the candidate member creates additional GSAD entries with specific source address such that reception of multicast IPsec messages from these senders is enabled. Using those information, the client is able to send multicast IPsec messages. From this point forward the client erases IKEv2 session information and stops using the member mealy state machine. Being now a member, the client should be up to date with the group changes and therefore it listens for incoming GSA_REKEY messages. Upon reception of GSA_REKEY message the client updates the corresponding Group-IKEv2 session information and GSAD entries with the new GSAs and key material. It should be mentioned that in our implementation source authenticity of GSA_REKEY message is not taken into consideration, since no digital signature mechanism is implemented, as it is left for future work. For that reason, we have included the expected size of AUTH payload with non-significant bytes in the GSA_REKEY message.

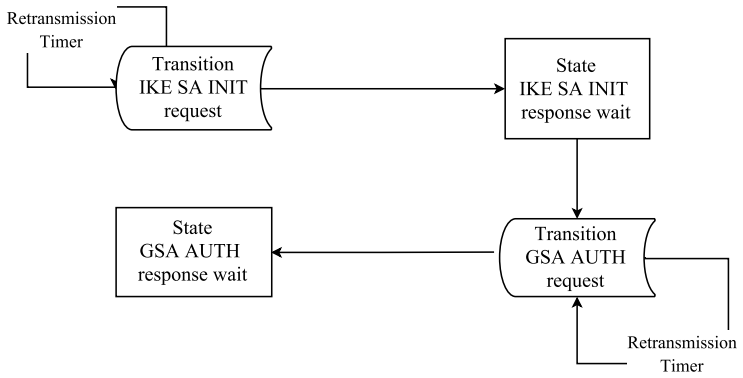

Figure 5.7: Candidate member mealy state machine of Group-IKEv2.

5.5 Implementation Limitations

In this work we have implemented policies for member registration with backward secrecy ensured in a multicast group. However, we did not implement any mechanism for member eviction in case a member wishes to leave the group or if it is compromised by an adversary. Moreover, no group key distribution algorithm was implemented. The only way member eviction is currently performed in the implementation is by setting a maximum limit to the number of GSA_REKEY messages that can be sent to the multicast group. If this limit is reached and the lifetime of GSAs and key material is expired, all members will be automatically evicted from the group.

Furthermore, no source authentication mechanism for GSA_REKEY messages, such as RSA digital signature, has been implemented as it is expected to be done in future work. Currently, the content of AUTH payload in GSA_REKEY messages consists of random bytes, such that the actual size of this payload is taken into consideration in this message. It should also be mentioned that no key generation algorithm was implemented since it was out of the scope of this thesis. We assumed that GC/KS maintains a pool of keys that can be used for different purposes of Group-IKEv2 protocol, such as encryption, integrity, and authentication.

The main challenge faced during the implementation phase was that the provided implementation of IPsec and IKEv2 was designed specifically for 32-bit platforms. Consequently our extensions were also implemented particularly for such platforms such as Openmote. Unfortunately, this platform is not included in the Cooja simulator, making the implementation and testing process more difficult. The debugging techniques that were used with the actual platforms were more difficult to conduct since synchronization and flawless communication had to be assured. Moreover, the fact that Group-IKEv2 is not standardized and there is no public implementation of a group key management protocol working in the Internet limited the range of experiments that we could conduct. Therefore, we only examined the behaviour and performance of our protocol within the 6LoWPAN network and not with a server over the Internet.

The following chapter presents the performance evaluation of our implementation in terms of communication overhead, memory utilization, energy consumption, and time. Moreover, a security analysis of the proposed architecture is done emphasizing the strong and vulnerable points of our communication scheme.

6

Evaluation

This chapter presents the performance evaluation and the security analysis of our system architecture for IoT devices. The performance evaluation is conducted in the experiment setup as our implementation to determine the suitability of our proposed system architecture to resource-constrained devices and determine the limitations that are posed in the devices. The security analysis aims to delineate the strengths and vulnerabilities of our system to potential security attacks in the IoT domain. Initially a detailed description of the experiment setup is provided, including the selected configuration settings and group and system policies. In Section 6.1.2 the results of our evaluation are presented, providing an insight into how the measurements of different metrics were acquired from the system and evaluating the results. An overall performance analysis is provided in Section 6.1.3. Last but not least, Section 6.2 presents the security analysis of our proposed architecture.

6.1 Performance Evaluation

The purpose of performance evaluation is to examine how efficiently our proposed architecture is supported by resource-constrained devices, in terms of memory utilization, energy consumption, member joining time, key distribution time, and communication overhead.

The challenge of Group-IKEv2 in IoT lies in the fact that, candidate members and GC/KS are resource-constrained devices. A candidate member has to preallocate specific memory for GSAD, and dynamically store Group-IKEv2 session information. Under circumstances at which it belongs to many multicast groups, it has to store multiple Group-IKEv2 sessions, each for every multicast group, and at the same time process incoming Group-IKEv2 messages of each session. From GC/KS's perspective, memory usage constitutes a big issue, since RAM is utilized so that GC/KS can simultaneously handle incoming join requests and store Group-IKEv2 sessions of all active members,

while at the same time Flash memory maintains all the required parameters for all groups and all their members. Moreover, communication overhead of Group-IKEv2 in 6LoWPAN is quite important since resource-constrained devices should not be overwhelmed by processing too large messages.

The energy consumption in resource-constrained devices when a new member joins the group is also of high interest. The network administrator should be aware of the amount of energy that is needed for this process, in order to configure suitable configuration settings, group and system policies. Moreover, it is important to know the amount of energy needed in GC/KS to distribute GSAs and key material to a candidate member, so that the system administrator is aware of the power limitations posed by this communication scheme and can choose accordingly a power strategy.

The time required for GC/KS to distribute the GSAs and key material to all members, and the time needed for a new member to join the multicast group are also of high importance for the system administrator. Time efficiency is a prerequisite in order to achieve GSA coherence within the multicast group, especially when operating in an unstable environment with many membership changes, as in IoT domain. The following section presents the experiment setup used in the performance evaluation of our proposed system.

6.1.1 Experiments setup

In the framework of this thesis, we have considered a specific experiment setup to assess how particular attributes of our proposed architecture fit to resource-constrained devices. In particular, we have considered through all our experiments a single group, which is initially empty without any active member, in which nodes request from GC/KS to join the group. Hence, we have mainly focused our evaluation on key provisioning to the joining nodes, rather than the group rekeying process for the existing members of the group.

We have used two different entities in our experiments, a udp-client and a udp-server, from Contiki udp-examples, with Multicast IPsec and Group-IKEv2 enabled. The udp-client plays the role of a candidate member that wishes to join a multicast group, and the udp-server plays the role of the authorized GC/KS of the multicast group. The udp-client sends periodically, every 15 sec, Hello messages to the multicast group, using the IPsec protocol, ESP. Group-IKEv2 is invoked in order for the candidate member to acquire the corresponding GSAs and key material from the GC/KS, as it has been described in Chapter 4. The platform used to evaluate our implementation of the system is Openmote. For simplicity reasons, we have assumed that there is only one multicast group that all candidate members wish to join, and that this group is initially empty. We have also considered that the maximum number of candidate members requesting for joining the group is five. Figure 6.1 depicts the experiment setup of five members and one GC/KS. It is assumed that all five members are valid senders in the multicast group.

Both candidate members and GC/KS have been configured with the following Security Association preferences:

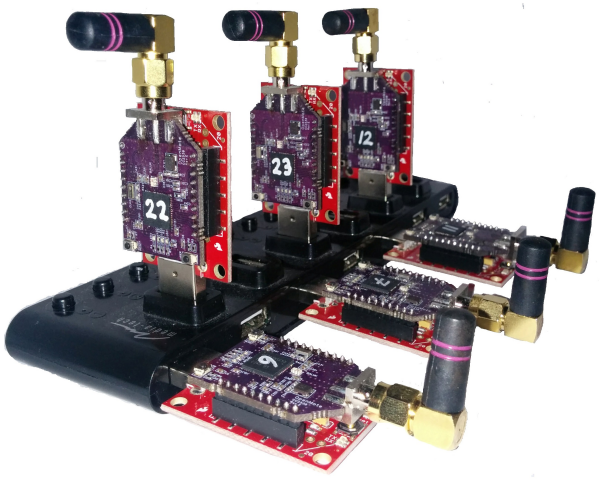

Figure 6.1: Experiment setup of five udp-clients and one udp-server.

- **IKE Security Association attributes used in GSA_AUTH:**
  1. Encryption and Integrity transform: AES-CCM8 [33],[34].
  2. Pseudo-random transform: PRF-HMAC-SHA2-256 [33].
  3. Diffie-Hellman group: DH256-RND-ECP-GROUP [33].

As mentioned in Chapter 4, the network administrator has to define the policies that should be applied to each group, and to the system. For our experiments the following policies have been applied:

- **Group Security Association encrypting Keys attributes (GSAK):**
  1. Encryption and Integrity transform: AES-CCM8 [33],[34].
  2. Encryption Key Length: 16
  3. Group Key Distribution protocol: NONE
  4. Authentication method: RSA Digital Signature
- **GSA encrypting traffic attributes (GSAT):**
  1. Protocol: ESP
  2. Encryption and Integrity transform: AES-CCM8 [33],[34].
  3. Encryption Key Length: 16
- **General Group Policies:**
  1. GSPD applies “bypass” policy to ICMPv6 and Group-IKEv2 traffic.

2. GSPD applies "protect" policy to incoming and outgoing traffic from and to the multicast group.
3. GSPD applies "discard" policy to all other traffic.
4. "Symmetric" directionality is assumed for the particular multicast traffic in GSPD of the candidate members and GC/KS.
5. The *lifetime* of GSAK has been set to 70 sec.
6. The *time to next update* has been set to 50 sec. It should be mentioned that this parameter has to be less than the lifetime of GSAK.
7. Group join policy has been defined, imposing to GC/KS to always send multicast GSA_REKEY message, before a new member joins the multicast group. In that way backward secrecy is enabled.
8. GSA_REKEY messages include only $GSAT$ and KD_{TEK} payloads, when time to next update expires and rekey event is triggered.
9. GSA_REKEY messages include both $GSAK$ and KD_{KEK} and $GSAT$ and KD_{TEK} when a new member is about to join the group.
10. No policy has been set for a member requesting to leave the group since such mechanism is not addressed in this thesis.
11. No eviction policy has been set, since no compromise detection mechanism has been implemented within this work.

- **System policies set to all the devices:**

1. Time delay to update is set to zero so that GSAD entries are immediately updated after parsing $GSAT$ and KD_{TEK} payload.
2. Time delay to update is set to zero so that Group-IKEv2 session information are immediately updated after parsing $GSAK$ and KD_{KEK} payload.
3. GSAD entries are source-specific meaning that in each entry the source address of the multicast traffic is specified.

6.1.2 Results

This section presents the performance evaluation results acquired from our experiments. In Section 6.1.2.1 we study how Group-IKEv2 messages size scales for different configuration settings and group policies. The memory utilization and energy consumption are discussed in Sections 6.1.2.2 and 6.1.2.3 respectively. Finally, Section 6.1.2.4 provides the results of measuring the key distribution time in GC/KS, and the member joining time in the candidate members.

6.1.2.1 Message Size

In this section, Group-IKEv2 message size is evaluated depending on the message type, the configuration settings applied in the network entities, and the group policies. We are

interested to examine how the message size varies for different types of messages and various configuration settings, so that we have a complete picture of how the communication overhead can range. The measured message size is the length of Group-IKEv2 message, without the IP header and Link layer header, as it is stored in the device buffers before it is sent to lower layers of Contiki Stack. It should be noted that in order to examine how the message size scales, we used the experiment setup described in Section 6.1.1, for various IKE SA, GSAK and GSAT parameters. The cipher suit combinations used in those experiments are shown in Table 6.1. When only AES-CCM8 algorithm is used, both encryption and integrity services are enabled without the need of any additional integrity algorithm. On the other hand, AES-CTR encryption algorithm is used along with AES-XCBC-MAC96 integrity algorithm so that both encryption and integrity are enabled.

Cipher suit	Encryption	Integrity
AES-CCM8	✓	✓
AES-CTR	✓	✗
AES-XCBC-MAC96	✗	✓

Table 6.1: Cipher suits used in IKE SA, GSAK and GSAT payloads for message size evaluation.

Tables 6.2 and 6.3 show the sizes of GSA_AUTH request and response messages for different IKE SAs, GSAK, GSAT, and Authentication Methods. As it can be seen in Table 6.2, the minimum message size of GSA_AUTH request is achieved when using Pre-Shared Key authentication and AES-CCM8, whereas the maximum message size is generated when Certificates are used and IKE SA uses both AES-CTR and AES-XCBC-MAC96. The shortest GSA_AUTH response is generated when Pre-shared Key authentication is used and AES-CCM8 is used in GSAK, GSAT, and IKE SA. On the other hand, GC/KS generates the longest message when Certificate authentication is enabled, and both AES-CTR and AES-XCBC-MAC96 are used in GSAK, GSAT and IKE SA. In the latter case, different keys have to be included in KD payload for AES-CTR and AES-XCBC-MAC96, which increases the total message size. Certificates increase significantly the size of both GSA_AUTH request and response messages, since both parties sign a specific text to authenticate each other, which in turn increases the overall communication overhead.

We can see that there is a difference of 4 bytes when using AES-CCM8 in IKE SA compared with using AES-CTR and AES-XCBC-MAC96 with any authentication method. This is because the encrypted message when using the cipher suit AES-CTR and AES-XCBC-MAC96, appends an Integrity Check Value (ICV) of 4 bytes length to the end of ESP packet in order to perform integrity check. On the other hand, combined mode algorithms, such as AES-CCM8, is responsible for ensuring integrity without necessarily appending the ICV to the end of ESP packet [35], hence the shorter

message size.

Authentication Method	AES-CCM8 (IKE SA)	AES-CTR, AES-XCBC-MAC96 (IKE SA)
PSK	148	152
Certificates	540	544

Table 6.2: GSA_AUTH Request message sizes in bytes for different configuration settings.

In overall, regardless of the configuration settings and group policies selected, it can be seen that GSA_AUTH response messages are longer than GSA_AUTH request, which is expected since GSA_AUTH response messages contain all the security associations for Group-IKEv2 (GSAK) along with their key material, all the security associations for IPsec (GSAT) and their key material and possibly the Sender IDs of the valid senders in the group. The more valid senders the more Sender IDs included in GSA_AUTH response, thus the longer the size of the message.

Authentication Method	IKE SA	AES-CCM8 (GSAK & GSAT)	AES-CTR, AES-XCBC-MAC96 (GSAK) - AES-CCM8 (GSAT)	AES-CTR, AES-XCBC-MAC96 (GSAK & GSAK)
PSK	AES-CCM8 (IKE SA)	427	463	499
	AES-CTR, AES-XCBC-MAC96 (IKE SA)	431	467	503
Certificates	AES-CCM8 (IKE SA)	795	831	867
	AES-CTR, AES-XCBC-MAC96 (IKE SA)	799	835	871

Table 6.3: GSA_AUTH Response message size in bytes for different configuration settings
.

Table 6.4 provides the sizes of GSA_REKEY messages. It is vital to clarify, that we have not taken into consideration source authenticity in GSA_REKEY messages, hence we did not implement any digital signature mechanism as it is left for future

work. Therefore, we have filled with non significant bytes the expected size of AUTH payload in GSA_REKEY messages in order to evaluate the size of this message. We have separated the results depending on the situation under which GSA_REKEY message is sent. Periodic GSA_REKEY messages are sent when time to next update is expired, so a GSA_REKEY message is sent periodically to renew the GSAs and key material before their lifetime expires. The update GSA_REKEY messages are sent when a new candidate member is requesting registration to the group and backward secrecy is enabled in the group policies. The payloads that should be included in each type of message under different circumstances are defined in the group policies. Specifically in the group policies defined in our implementation, both GSAK and GSAT payloads are included in the update GSA_REKEY message along with their corresponding KD payloads sent upon a new node joining the group, whereas in periodic GSA_REKEY messages only GSAT payload is included, along with its corresponding KD, so that communication overhead is minimized as much as possible. It is therefore expected and also obvious from our results that update GSA_REKEY messages have longer size than periodic GSA_REKEY message.

Message Type	AES-CCM8 (GSAK & GSAT)	AES-CCM8 (GSAK), AES-CTR, AES-XCBC-MAC96 (GSAT)	AES-CTR, AES-XCBC-MAC96 (GSAK), AES-CCM8 (GSAT)	AES-CTR, AES-XCBC-MAC96 (GSAK & GSAT)
Periodic GSA_REKEY	148	184	152	188
Update GSA_REKEY	315	351	355	391

Table 6.4: Periodic and Update GSA_REKEY message sizes in bytes for different cryptographic suits used in GSAK and GSAT.

Comparing GSA_AUTH with GSA_REKEY message sizes, we can see that GSA_REKEY messages are shorter than the total size of GSA_AUTH messages. Hence we can conclude that from a communication overhead perspective, the joining process is more costly than the rekeying process.

6.1.2.2 Memory Utilization

Since IoT devices have limited memory, it is important to quantify the memory requirements of our solution to determine if it is suitable for IoT or not. Therefore, here we measure the memory utilization of both RAM and Flash of multicast IPsec with Group-IKEv2 in both entities, Candidate Member and GC/KS.

Memory utilization is determined by processing the executable files of our implementation for Openmote platform. All the implementation data are stored in RAM and are obtained by the "data" and "bss" segments of the .elf files, whereas the program code is stored in flash memory, and is obtained by the "text" segment of the .elf file. Table 6.5 shows the results obtained from the executable files with Multicast IPsec and Group-IKEv2 enabled and disabled.

Entities	RAM (Bytes)	Flash (Bytes)
udp-server as GC/KS with Group-IKEv2	24236	79840
udp-client as CM with Group-IKEv2	24300	80072
udp-server without Group-IKEv2	13916	43548
udp-client without Group-IKEv2	13984	43612

Table 6.5: RAM and Flash memory utilization in bytes, in udp-server as GC/KS and in udp-client as candidate member (CM) with IPsec and Group-IKEv2 enabled, and in udp-server and udp-client with IPsec and Group-IKEv2 disabled.

As it is observed in Table 6.5, RAM utilization in udp-server operating as the GC/KS and in udp-client as a candidate member, is much higher than the RAM utilization in simple udp-server and udp-client. This is due to the fact that five additional data structures are included in order to enable multicast IPsec and Group-IKEv2. Table 6.6 shows the memory utilization of different components in Multicast IPsec and Group-IKEv2 implementation.

The maintained data structures are connected lists in Contiki OS, with specified maximum number of elements. The total memory needed for each list is preallocated in advance during compilation time. Both candidate member and GC/KS store information regarding their active IKEv2 connections in ike_statem_session_t structure, along with temporary information used to establish IKEv2 sessions in ike_statem_ephemeral_info_t. Additionally, Group-IKEv2 session information are stored in a structure called gsak_entry_t, maintaining the GSAKs for all active multicast groups. The group security associations for IPsec protocols are stored in two separate connected lists of the data structure type sad_entry_t. One list is used for incoming and one for outgoing traffic.

It should be mentioned that the required memory in a candidate member for joining a single multicast group is the sum of the memory needed for IKEv2 session, IKEv2 session ephemeral information, and Group-IKEv2 Session data structures, which is 1357 bytes. The required memory for enabling outgoing multicast traffic to f groups is $116 \cdot f$ bytes, whereas the required memory for enabling incoming multicast traffic in a single group is $116 \cdot h$, where h is the number of valid senders in the multicast group, since it is necessary to maintain source specific entries in GSAD for incoming traffic.

Figures 6.2 and 6.3 show RAM and Flash memory utilization of Group-IKEv2 implementation together with Contiki stack in GC/KS, for varying number of total members in the group. It is evident that as the size of the group grows, the memory

Data Structure	Memory (Bytes)
IKEv2 Session Information	264
IKEv2 Session Ephemeral Information	960
Group-IKEv2 Session Information	191
Encryption Key in Group-IKEv2 Session	20
Integrity Key in Group-IKEv2 Session	16
Authentication Key in Group-IKEv2 Session	91
Incoming GSAD	116
Outgoing GSAD	116
Encryption Key in GSAD	20
Integrity Key in GSAD	32
Certificate	444

Table 6.6: Memory size in bytes of Multicast IPsec and Group-IKEv2 components.

required to store data for the IKEv2 session, Group-IKEv2 session and GSAD, is also increased both in RAM and Flash Memory, fact that is expected since more information are stored for more members.

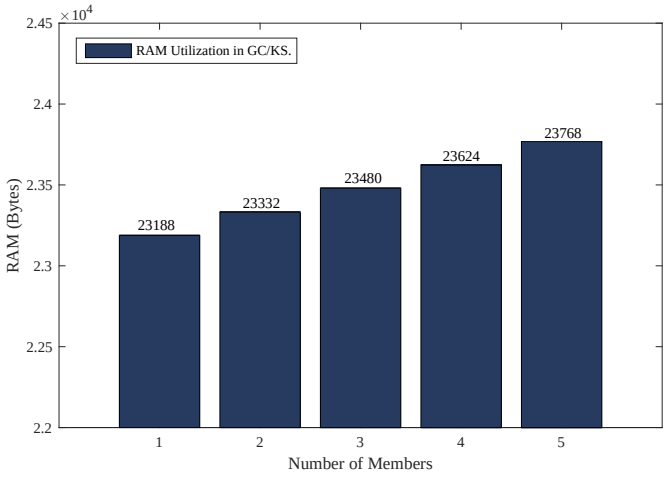

Figure 6.2: RAM utilization in Bytes in GC/KS versus the number of active members in the group.

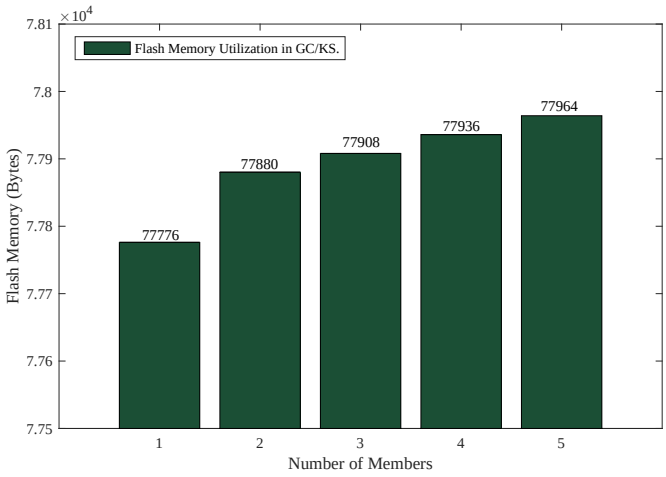

Figure 6.3: Flash memory utilization in Bytes in GC/KS versus the number of active members in the group.

6.1.2.3 Energy Consumption

One of the most important metrics that has to be taken into consideration in order to evaluate the efficiency of Group-IKEv2 in IoT, is the energy consumption. We estimate the total energy consumed by a candidate member during joining process in the group, considering the candidate sending IKE_SA_INIT request, parsing IKE_SA_INIT response, sending GSA_AUTH request and parsing GSA_AUTH response message. Moreover, we have calculated the total energy consumed by GC/KS during distribution of GSAs and key material to a member, including the reception of IKE_SA_INIT request, sending IKE_SA_INIT response, parsing GSA_AUTH request, and sending GSA_AUTH response. Additionally we evaluate the energy consumption during joining process in a candidate member and the energy consumption during key distribution in GC/KS when using different authentication methods for mutual authentication in GSA_AUTH messages, Pre-shared Key and Certificates. The network administrator should be aware of the energy consumption under these different situations, in order to take them into account before selecting the group policies.

The total energy consumed is calculated with the help of Energest and Powertrace tools of Contiki. Energest uses timers to estimate the time duration that the micro-controller is in CPU and LPM state, and how long the radio is in Transmit or Listen state. Making use of Energest tool, Powertrace prints out the estimated number of cycles (ticks), that the micro-controller and radio have been in the corresponding states. Then the actual time duration spent in each state, δt, is calculated by dividing the number of cycles spent in the corresponding state, with the frequency of rtimer for Openmote, which is 32768 cycles/sec, according to Openmote datasheet [30]. Given that the supply Voltage is 3 Volt, we calculated the energy consumption in each state with (6.1). It should be mentioned that current consumption, I_{state}, differs in each state.

$$E_{state} = V \cdot I_{state} \cdot \delta t \tag{6.1}$$

Finally, the total energy consumption is calculated by the sum of the energy consumption in each state (6.2).

$$E_{total} = E_{CPU} + E_{LPM} + E_{Tx} + E_L \tag{6.2}$$

It should be mentioned that in order to provide more accurate results, we have repeated the experiments 10 times. Figure 6.4 shows the estimated energy consumption of the joining process for a candidate member, for all our experiment attempts, using Pre-shared Key (PSK) authentication in GSA_AUTH messages. It can be seen that most of our estimations are close to the average energy consumption, with the exception of three experiments, which appear to have higher variance.

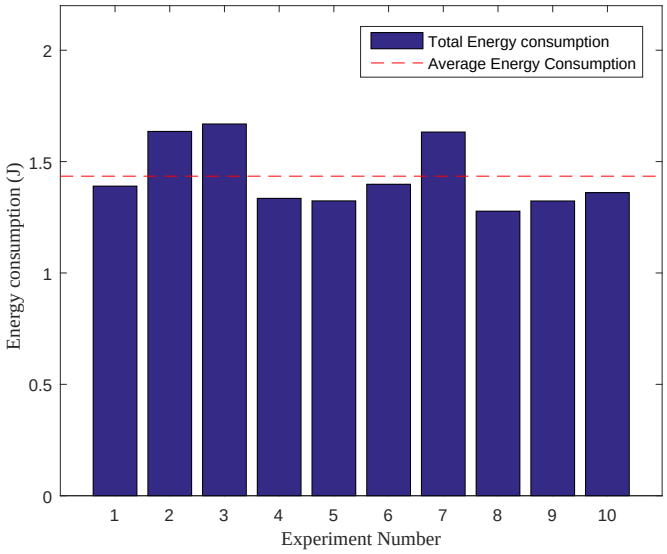

Figure 6.4: Estimated total energy consumption in Joules in a candidate member during member joining process in 10 experiments.

The results of the estimated energy consumption in a candidate member during the joining process and during the distribution of GSAs and key material in GC/KS, using two different authentication methods in GSA_AUTH messages, are shown in Figure 6.5. The estimated total energy consumption in a candidate member during the joining process is measured to be less than the energy consumption during key distribution in GC/KS, when both entities use Pre-shared Key authentication method. In particular, GC/KS consumed 0.033 Joules more, than a candidate member. However, during our experiments we noticed that the standard deviation of the estimated energy consumption in a candidate member, is higher than in GC/KS, Table 6.7. This means

that the energy consumed in a member did not remain the same throughout our experiments. We observed that the estimated time duration that the radio was in listen state, and the estimated time that the micro-controller was in CPU state experienced variations, fact that increased the standard deviation of our result.

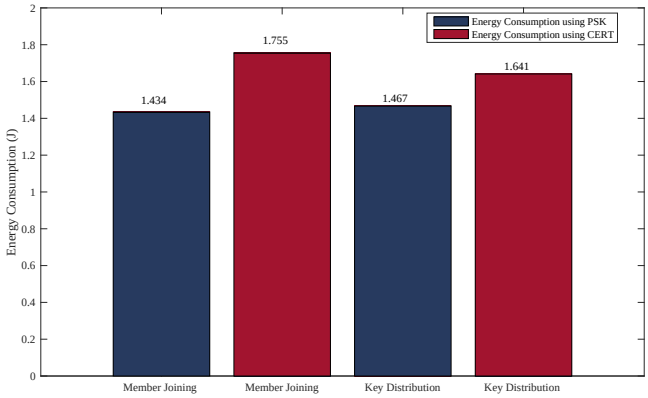

Figure 6.5: Total Energy consumption in a candidate member during joining process and in GC/KS during key distribution processes for different authentication methods.

Event	Total Energy Consumption (J)	Standard Deviation
Joining process with PSK	1.4345	0.1423
Joining process with CERT	1.7555	0.1812
Key Distribution with PSK	1.4675	0.0437
Key Distribution with CERT	1.641	0.044

Table 6.7: Energy Consumption in Joules in Group-IKEv2 processes.

Figure 6.5 shows how the energy consumption scales depending on the authentication method used in GSA_AUTH in a candidate member during the joining process and in GC/KS during the key distribution process. It is obvious that when certificates are used, the estimated total energy consumption in both processes is much higher than when PSK is used. Moreover, the energy consumption in the candidate member during joining process is higher than the estimated total energy consumption in GC/KS during key distribution when both are using Certificates. The standard deviation of the energy consumption during joining process is 0.181, fact that implies that there are variations in the estimation of the energy consumption, which mostly originate from the variations in the estimated time that the radio of the device is in Listen state. On the other hand the standard deviation of the estimated energy consumption in GC/KS during key distribution is 0.044, which implies that the energy consumption in GC/KS was almost the

same for all our experiment attempts, as it can be seen in Table 6.7. Therefore, it is possible that the energy consumption in a member is not always higher than the one in the GC/KS.

6.1.2.4 Key Distribution Time and Member Joining Time

The key distribution time is the time needed by GC/KS, to provide all the GSAT and their TEK, along with all the GSAK and their KEK, to all the candidate members of a group. We measure this time in order to examine how it varies with the number of candidate members.

The calculation of the key distribution time is done with the assistance of the function clock_time() of Contiki, which outputs the current system time. The key distribution time is obtained by subtracting the system time when IKE_SA_INIT request is received and Group-IKEv2 is initialized in GC/KS, from the system time when key distribution to all members is completed, that is when the GSA_AUTH response is sent to the last candidate member, see (6.3). It should be noted that this time also includes the time needed to send multicast GSA_REKEY messages to the group members before a new member joins the group. For the purposes of this experiments, we have considered that all the candidate members have requested to join the group at around the same time, with the GC/KS processing the incoming join requests, following a LIFO policy.

$$Key\_Distribution\_Time = T_{current} - T_{Init} \tag{6.3}$$

Additionally, we have measured the joining time of a candidate member, in order to evaluate how it varies depending on the total number of candidate members requesting for registration at around the same time. This time is calculated by subtracting the system time when the candidate member sent an IKE_SA_INIT request, from the system time when successful parsing of the GSA_AUTH response is completed, see (6.4). It should be mentioned that in this implementation, if the retransmission limit is reached in a candidate member, all Group-IKEv2 session information are erased, and registration process is re-initiated. In that case, the calculation of the total registration time is re-initiated as well. The experiments were repeated 25 times to enhance the accuracy of our results, by considering the average.

$$Member\_Joining\_Time = T_{current} - T_{Init} \tag{6.4}$$

The key distribution time is evaluated with varying number of candidate members requesting at the same time for joining the group, and for different authentication methods for GSA_AUTH messages, PSK, and certificates. Figure 6.6 shows the visual representation of our results.

As it can be seen, the more candidate members request for registration, the longer time is needed in GC/KS to distribute the GSAs and key material to all the members, regardless of the authentication method used in GSA_AUTH messages. Furthermore, it is evident that the rate the key distribution time increases is not linear with the number of members requesting registration. This is due to the variations of our results,

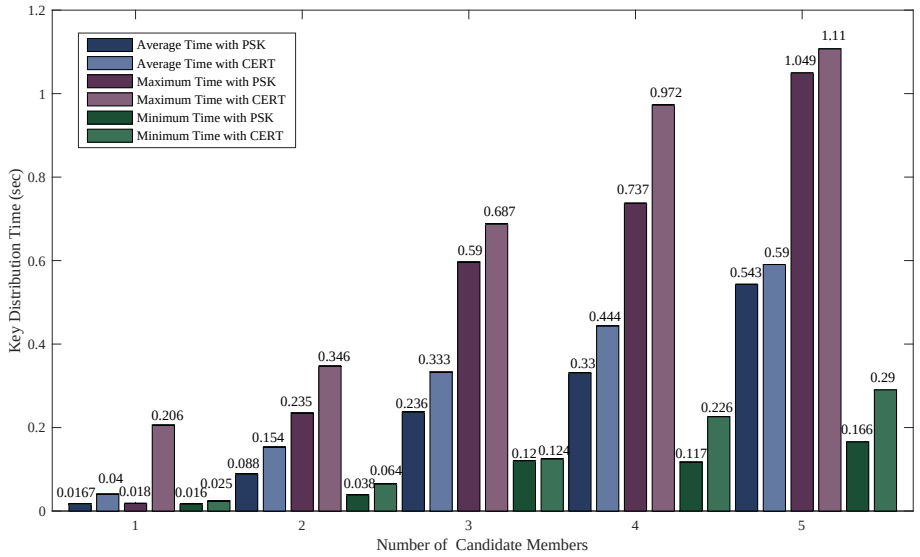

Figure 6.6: Average, Maximum and Minimum Key Distribution Time in GC/KS using either Pre-shared Key or Certificates in GSA_AUTH messages versus the total number of candidate members in the group.

which are also escalating with the number of candidate members. In fact, the average key distribution time for only 1 member is approximately 32 times shorter than for 5 members. Due to the fact that all the members initiate joining process at around the same time, which means they transmit their requests at around the same time, some messages may collide and get lost, therefore they need to be retransmitted. Moreover, it is obvious that the average key distribution time is significantly higher for the whole range of members, when Certificates are used, instead of Pre-shared Key authentication.

The member joining time is evaluated for different authentication methods, PSK and certificates, and with varying number of candidate members requesting to join the group. The average, minimum, maximum of our results are presented in Figure 6.7.

It is evident, by comparing the minimum maximum and average time, that as the number of candidate members increases, the member joining time is prolonged, and the variation among our results becomes higher. This fact implies that the joining time in the experiments of one candidate member is similar for most of the experiments. On the other hand, this does not apply for the experiments of five candidate members requesting to join the group since the joining time varies significantly. This is because multiple candidate members request to join the group at the same time, which results to collisions and retransmissions of messages. Moreover, we can see that the member joining time using PSK authentication is much shorter through the whole range of candidate members, compared with when certificates are used. It is remarkable that the minimum member joining time lies between 0.022-0.024 sec when using Pre-shared Key authen-

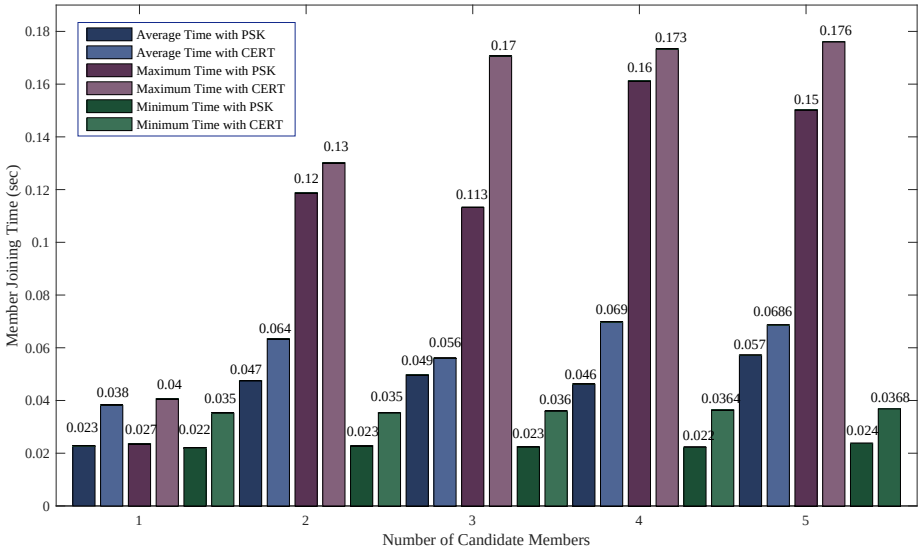

Figure 6.7: Average, Minimum and Maximum Member Joining Time in a candidate member vs the total number of candidate members in the group using either Pre-shared Key or Certificates as authentication method in GSA_AUTH messages.

tication and between 0.035-0.037 sec when using Certificates regardless the number of candidate members, which means that the time needed for the first candidate member to join the group is almost the same. However, the rest of the candidates suffer from delay and retransmissions of their request.

Figure 6.8 shows the member joining time durations of all the experiment attempts, for one to five number of candidate members in the group. It should be noted that those results are specific for our implementation and can be improved in the future by improving the way incoming requests are handled in GC/KS. Taking a closer look in the figure, it is indicated that at least one candidate member almost immediately joined the group, but the rest of the candidates experienced delays. This is due to the fact that GC/KS is implemented to handle incoming join requests following a LIFO policy. The usual outcome of receiving multiple join requests at the same time is that the first candidate is served first, however the next candidate to be served is the last candidate in the list, leaving the rest candidates to wait. Some of the remaining candidate members may register right after the first join attempt or eventually have to retransmit their requests, and if the maximum retransmission limit is reached without any response, they have to re-initiate the joining process, which inevitably results to a delayed join.

The system administrator has to consider how long it takes for a GC/KS to distribute all GSAs and key material to all the candidate members in order to define the time delay to update Group-IKEv2 session information and GSAD information when the number of candidates is known, so that IPsec packet loss is prevented. If the registries of

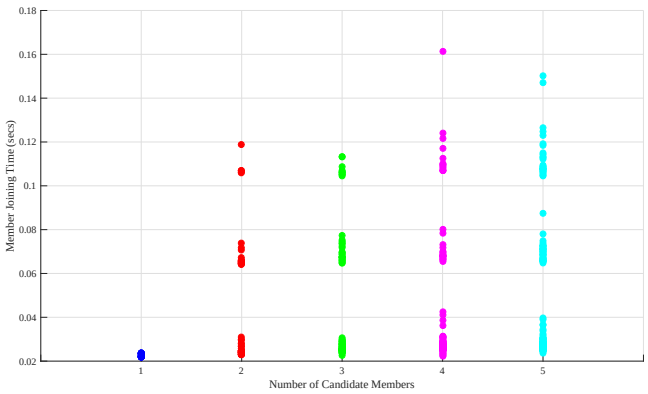

Figure 6.8: Total Member Joining Time for each candidate member versus the total number of group members for 25 experiments using our default experiment setup.

the members are updated too soon, before key distribution to all members is completed, then a potential IPsec packet of the previous GSAs and keys will be discarded.

6.1.3 Performance Analysis

In this section we discuss the suitability of our proposed architecture to resource-constrained devices, depending on the performance results presented in the previous section.

The communication overhead of our implementation, depends strictly on the group security associations and the group policies that the network administrator will select. As it is seen in Section 6.1.2.1, GSA_AUTH messages are longer than GSA_REKEY messages. Moreover, by examining how the size varies depending on selected cipher suits, we found out that by using Certificates and both AES-CTR and AES-XCBC-MAC96 in GSAK, GSAT and IKE SA, the size of GSA_AUTH messages increases significantly. Before selecting a cryptographic suit and authentication method in the group security associations, the network administrator should always take into consideration the capabilities of the network devices, while at the same time consider the security risks that arise. There should be a compromise between network performance and security in the system. Overall, the communication overhead generated by Group-IKEv2 depends on the total number of candidate members requesting to join the group, and the *lifetime* of GSAK. The less the lifetime of GSAs the less the *time to next update* the more frequent the periodic GSA_REKEY messages, thus the higher the communication overhead. Nevertheless, it should be noted that periodic GSA_REKEY messages create less communication overhead than re-initiating member registration, by exchanging IKE_SA_INIT and GSA_AUTH messages.

Memory utilization results indicated that the proposed system architecture can be applied to class 2 resource-constrained devices, described in [31], since RAM and Flash memory required for this scheme with maximum 5 members in the group, reach approximately 24kB and 78kB respectively. Moreover, it should be mentioned that the amount of data stored by the devices directly depends on the number of active members within the group since the more members, the more RAM and Flash memory is needed to store all GSAs and keys. Considering that the available RAM in Openmote platform is 32KB, for every additional member in the group 144 additional bytes are needed, and assuming that there is only one group, the maximum supported number of members in GC/KS is approximately 65 members. However, in practice more groups are present in a network and such many members will impact significantly the energy consumption and key distribution time in the GC/KS and also the energy consumption and the joining time for the candidate members. GC/KS's micro-processor will be burdened by at least 65 registration requests, which are impossible to handle effectively in short time, and the members will suffer from countless retransmission and re-initiations of joining process, which inevitably result to system failure. Thus, an adequate number of members that can be supported by GC/KS with this specific implementation would be at maximum, 10 members.

The proposed architecture is suitable for devices of energy limitation classes E1 and E9 [31], in which either the devices use batteries that are periodically recharged or replaced, or they have access to power supply. The selected group policies of the system determine the energy limitation class that can be used. This is due to the fact that they regulate the maximum number of members, thus the number of times the key

distribution process is initiated. They define the frequency of periodic GSA_REKEY messages, thus the number of times periodic rekey process is initiated. Additionally, if backward secrecy is enabled in the group policies, update GSA_REKEY messages are sent, which increases the overall energy consumption. The communication scheme used requires both the members and the GC/KS to follow always-on strategy for using power for communication, where the device is always connected to the network in order to receive GSA_REKEY messages. As we have already mentioned in Section 6.1.2.3 PSK authentication method is less costly than certificates. Additionally, the estimated consumed energy during joining process in a candidate member has quite high standard deviation given all the number of the experiments. This occurs due to the variation of the estimated energy consumption during CPU and Listen state, since the time in those states in every joining attempt was quite different. On the other hand the standard deviation of the estimated consumed energy in the GC/KS during distribution of GSAs and key material is insignificant, which means that almost the same amount of energy is consumed in GC/KS every time this process is initiated. The overall energy consumption in GC/KS depends on the number of members requesting to join the group, and on the number of GSA_REKEY messages sent.

The member joining time and key distribution time was evaluated depending on the number of candidate members in the multicast group and the authentication method used in GSA_AUTH messages. The dependence of the calculated member joining time and key distribution time on the number of candidates is ascertained. The more the number of candidates grows, the more time is needed to distribute all the GSAs and keys to the whole group. Additionally, the use of certificates prolonged even more the joining time and the key distribution time. It should be noted that the total time needed to distribute the GSAs and keys is inevitably influenced by the overall member joining time. The GC/KS is able to handle simultaneous registration of the members by processing the incoming requests using a LIFO policy. In particular, the incoming request that is handled at any time is the one that has more recently been received, even if there are already requests waiting to be processed. For that reason, most of the times the "delayed" candidate members have to retransmit their request, until the maximum retransmission limit is reached. After this limit, the candidates erase their Group-IKEv2 session and wait till the joining process is initiated again. This results in higher variations among the candidate members' total joining time, but also in the dramatic increase of the total key distribution time in GC/KS.

6.2 Security Analysis

Global connectivity in IoT introduces many security risks in 6LoWPAN. In this section we provide a thorough analysis of Multicast IPsec with Group-IKEv2 for IoT from a security perspective, in order to determine if it is sufficient to protect multicast traffic from attacks in the IoT domain. It should be mentioned that within the purposes of this analysis we have assumed that the vulnerabilities of IKEv2 are known and therefore we discuss only the vulnerabilities introduced by Multicast IPsec with Group-IKEv2.

Multicast IPsec for IoT enables secure transmission of multicast traffic, providing message confidentiality, integrity protection, group authentication, and replay attack protection. Confidentiality is achieved by encrypting the traffic and integrity protection by determining if the packets of the traffic are modified or not. With group authentication, it is examined if the source of the multicast traffic belongs to the multicast group. The members can prevent replay attacks using the sequence numbers, which are maintained in source-specific GSAD entries. Moreover, the use of GSPD in all IPsec entities offers a similar functionality as an access list. Particular "bypass", "protect", and "discard" rules are applied to packets from specific origin IP address to specific destination IP address and port before they go through the outgoing or incoming interface of the IPsec entity.

Group-IKEv2 for IoT ensures confidentiality, message integrity, and mutual authentication between each candidate member and the GC/KS, and source authenticity of GSA_REKEY messages. The selected cipher suits for encrypting multicast IPsec traffic along with their corresponding key material are distributed securely with Group-IKEv2 messages by GC/KS to all the authorized group members. An adversary not originating from the group is able to receive those messages, but is not able to decrypt them and compromise the group communication. Moreover, backward secrecy is enabled by updating the GSAK along with their keys and the GSAT along with keys, in the existing members of the group before registration of a new member is done.

One of the most common attacks in IoT domain is DDoS. The attacker aims to overwhelm a network entity by flooding it, such that the entity's computational and memory resources are exhausted. Such an attack can be done by an adversary flooding GC/KS with IKE_SA_INIT requests, using fake IP addresses. Group-IKEv2 is possible to detect such an attack by using Cookies. The GC/KS includes a Cookie payload to the response and requests from the candidate member to repeat the request with the Cookie payload included. In that way, the GC/KS increases the complexity of the attack, and if the adversaries are not able to intercept the cookie and reply with a spoofed address, the GC/KS verifies the validity of the candidate member and proceeds with the request.

The highest security risk of Group-IKEv2 lies in GC/KS's and member's trustworthiness. In GSA_AUTH exchange, mutual authentication is used in order for each GC/KS - candidate member pair to validate their identities. After successful authentication, the member registration is complete and GC/KS initiates rekey events. In GSA_REKEY messages, source authentication is used in order for each member to authenticate if the originator of the message is GC/KS using a digital signature authentication method, such as RSA, DSS Digital Signature, or ECDSA.

Our solution addresses the event where a member wishes to leave by proposing to update the rest of the members in the group individually with unicast GSA_REKEY messages protected with pairwise key material that was provided upon registration. Alternatively, a group key distribution algorithm could be used such as LKH to efficiently rekey the group members using unicast and multicast GSA_REKEY messages.

If a group member is compromised by an adversary, our proposed architecture is not able to detect and deal with it immediately. Detection of malicious behaviour within

the multicast group can be done using an IDS, proposed in [32]. However, it is quite complicated to integrate with our system and it is out of the scope of this thesis. It is under the network administrator's decision to include a separate mechanism to detect malicious behaviour and update accordingly the GC/KS. Then in turn, the GC/KS without considering the actual reason for eviction, it can proceed with updating the rest of the member in the group individually with unicast GSA_REKEY messages, in the same manner as when a member wishes to leave the group.

Last but not least, in the unfortunate event that GC/KS is compromised by an adversary, Group-IKEv2 will fail since GC/KS is a single point of failure. All the GSAs and key material will be compromised and the adversary will be able to impersonate the GC/KS.

7

Conclusion

Throughout this thesis work we have designed and implemented a system that enables Multicast IPsec in IoT, with the use of a group key management protocol, Group-IKEv2. Our proposal has been evaluated in terms of performance in order to determine the suitability of this protocol in IoT domain. It also has been analysed in terms of security in order to identify the strong and weak points of this protocol and how they can be eliminated in the future.

Using our results, we are able to answer our initial question, which motivated this thesis, regarding the possibility of extending the IP security architecture for IoT such that end-to-end secure group communication in the network layer is achieved. We have adopted the standardized optional Multicast Extensions to the Security Architecture for the internet protocol [29], in which very few changes in IPsec architecture are required, and the use of a group key management protocol is mandatory. Moreover, we have achieved dynamic establishment of group security associations and key material in a multicast group with the use of a centralized approach of group key management architecture. We conclude that this is achievable with the use of a network entity as a Group Controller/Key Server (GK/KS), responsible for managing and distributing the group key material to the authorized entities in the group.

Throughout our experimental research, we have examined the possibility of GC/KS to be a device with limited capabilities. We found out that even though GC/KS can successfully distribute the GSAs and key material to the group, the limited memory and computational power of the device posed limitations to the number of entities that can be served simultaneously, to be less than 10. This issue could be overcome with improvements in the way a GC/KS is serving the registration requests, which is a more implementation-specific solution or with the use of multiple GC/KSs for a single multicast group, adopting a decentralized approach of group key management architecture.

In terms of energy utilization, we have concluded that in case Certificates are used for authentication purposes instead of Pre-shared Key, more energy is required to

process the data and higher communication overhead is created. Therefore, the network administrator should always take into consideration the security requirements of the system along with the desired performance in the devices and in the network, before defining the group policies. In addition, depending on the scale of the group, the network administrator should take into consideration the time required by GC/KS to distribute the keys, in order to define the system policies.

In conclusion, our proposal is sufficient to establish end-to-end secure group communication within a group of resource-constrained devices with backward and forward secrecy. Our proposal ensures confidentiality, integrity, authentication, replay attack protection, and DDoS attack detection. Some aspects have not been addressed during this work, such as detection of malicious behaviour in the group with an IDS, group key generation, and integration of group key distribution algorithms in Group-IKEv2 protocol. The final chapter delineates the possible future work related to this thesis.

8

Future Work

This chapter presents the future work that can be conducted as continuation of this thesis in order to improve the performance of our proposed scheme and to enhance security. The features that can be added to this work in order to improve Group-IKEv2 performance in resource-constrained devices are the following:

1. Improve the way GC/KS handles incoming join requests in Group-IKEv2 implementation, by changing the LIFO policy to First come first serve (FCFS) or priority-based depending on the type of the traffic.

2. Implement acknowledgement mechanism for GSA_REKEY messages. In that way, the GC/KS will be able to know when retransmission of GSA_REKEY messages is required, and also will be able to estimate in combination with the key distribution time, when GSAD entries need to be updated. Thus, the member joining time will be improved.

3. Implement a decentralized approach for group key management, by configuring the GC/KS to a powerful device rather than in a resource-constrained device. This GC/KS will be responsible for distributing GSAs and keys to subgroup of GC/KSs, which will be resource-constrained devices. In turn, each GC/KS will be responsible for distributing the group keys to the corresponding subgroups. In that way, GC/KS will not be a single point of failure and memory resources needed to store all the GSAs and keys will be significantly reduced.

The features that can be added to the implementation of this thesis to enhance security are the following:

1. Implement mechanism in Group-IKEv2 for a member eviction or requesting to leave the group.

2. Implement an authentication method for GSA_REKEY messages such that source authenticity is ensured.

3. Integrate an Intrusion Detection System [32] to detect malicious behaviour within the group, which in turn will be able to trigger member eviction.

4. Integrate a group key distribution algorithm, like LKH and OFT, with GroupIKEv2 in order to preserve forward secrecy in a more efficient manner.

5. Include a key generation mechanism for creating the group keys depending on the needs of the group key distribution algorithm.

Bibliography

[1] Ovidiu Vermesan, Peter Friess, "Internet of Things - From Research and Innovation to Market Deployment", *River Publishers*, 2014.

[2] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", *RFC 4253*, *IETF*, January 2006, .

[3] M. Tiloca, S. Raza, K. Nikitin, S. Kumar. "Secure Two Way DTLS-Based Group Communication in the IoT", draft-tiloca-dice-secure-groupcomm-00 (work in progress), *Internet Engineering Task Force*, 2015.

[4] M. Tiloca, K. Nikitin and S. Raza. "Axiom - DTLS based secure IoT group communication", accepted for publication in the ACM Transactions on Embedded Computing Systems, *Special issue on Embedded Computing for the Internet-of-Things (IoT), ACM*, 2016 (To appear)

[5] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, *RFC 7296*, *IETF*, October 2014, .

[6] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", *RFC 4301*, *IETF*, December 2005, .

[7] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", *RFC 2627*, *IETF*, June 1999, .

[8] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", *RFC 4046*, *IETF*, April 2005, .

[9] Stallings W. "Cryptography and Network Security Principles and Practice Sixth Edition", *Pearson Education*, 2014, 2011, 2006.

[10] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", *RFC 6347*, *IETF*, January 2012, .

[11] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", *RFC 7252*, *IETF*, June 2014, .

[12] IEEE P802.15 Working GroupIEEE, "Standard for Local and metropolitan area networks— Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)", *Approved 14 August 2012 American National Standards Institute.*

[13] Montenegro G., Kushalnagar N., Hui J., "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", *RFC 4944*, *IETF*, September 2007.

[14] Hui J. Ed., Thubert P. , "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", *RFC 6282*, *IETF*, September 2011.

[15] Raza Sh., Duquennoy S., "Secure Communication for Internet of Things - A Comparison of Link-Layer Security and IPsec for 6LoWPAN", *Journal of Security and Communication Networks*, Wiley, 2012.

[16] Raza Sh., Duquennoy S., Chung T., "Securing Communication in 6LoWPAN with compressed IPsec", *7th IEEE International Conference on Distributed Computing in Sensor Systems*, Barcelona, Spain, 27-29 June 2011.

[17] Winter Ed. T., Thubert Ed., Brandt A., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", *RFC 6550*, *IETF*, March 2012.

[18] Yacine Challal , Hamida Seba, "Group Key Management Protocols: A Novel Taxonomy", *International Journal of Information Technology*, Vol. 2, Number 1, 2005, ISSN:1305-2403.

[19] Rafaeli S. , Hutchison D. , "A Survey of Key Management for Secure Group Communication", Computing Department, *Lancaster University*, 2003.

[20] Mohamed Salah Bouassida, Isabelle Chrisment, "Group Key Management in MANETs", *International Journal of Network Security*, Vol. 6, Number 1, PP. 67-79, Jan. 2008.

[21] Porambage P. , Braeken A., Schmitt C. , "Group Key Establishment for Enabling Secure Multicast Communication in Wireless Sensor Networks Deployed for IoT Applications", *IEEE Access*, September 8, 2015.

[22] Rowles S., Yeung A., "Group Key Management using IKEv2", *Network Wokring Group*, Internet-Draft, March 21, 2016.

[23] Sahar M. Ghanem, Hussein Abdel-Wahab, "A Secure Group Key Management Framework: Design and Rekey Issues", *Proceedings of the Eighth IEEE International Symposium on Computers and Communication*, 2003.

[24] Deuk-Whee Kwak, SeungJoo Lee, "An Efficient LKH Tree Balancing Algorithm for Group Key Management", *IEEE Communications Letters*, Volume 10, Number 3, March 2006.

[25] Liu Haike, Hao Xiaoqiang, "A Novel LKH Tree Structure Based on Heuristic Search Algorithm", *IEEE International Conference on Communication Problem-Solving (ICCP)*, 2014

[26] Balenson D., McGrew D., Sherman A., "Key Management for Large Dynamic Groups: One Way Function Trees and Amortized Initialization", *Advanced Security Research Journal*, Volume 1, Number 1,pp 28-40, 1998.

[27] Khandarkar K., Mapari R., "Analysis of Group Key Management Scheme using One-way Function Tree", *International Journal of Applied Information Systems*, Volume 1, Number 3, Feb. 2012.

[28] Fathirad IR., Devlin J., Atshani S., "Network-Specific Attacks on Diffie-Hellman Key-Exchange in Commercial Protocols", *International Journal of Computer Theory and Engineering*, Vol. 8, No. 2, April 2016.

[29] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", *RFC 5374*, *IETF*, November 2008.

[30] "CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications", *Texas Instruments Incorporated*, 2016.

[31] Bormann C., Ersue M., Keranen A., "Terminology for constrained-node networks", *RFC 7228*, *IETF*, May 2014.

[32] Raza Sh., Wallgren L. , Voigt Th. , "SVELTE: Real-time Intrusion Detection in the INternet of Things", *Ad Hoc Networks Journal*, Elsevier, 2013.

[33] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", *RFC 4307*, *IETF*,December 2005, .

[34] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", *RFC 4309*, *IETF*, December 2005, .

[35] Kent, S., "IP Encapsulating Security Payload (ESP)", *RFC 4303*, *IETF*, December 2005, .

[36] Vilajosana X., Tuset-Peiro Pere, "Openmote: Open-source Prototyping Platform for the industrial IoT" *Conference paper at 7th EAI International Conference on Ad Hoc Networks (AdHocNets)*, At San Remo, Italy.