Network Working Group
Internet-Draft
Intended status: Experimental
Expires: November 3, 2017

D. Garcia
R. Marin
University of Murcia
A. Kandasamy
A. Pelov
Acklio

May 2, 2017

LoRaWAN Authentication in RADIUS
draft-garcia-radext-radius-lorawan-03

Abstract

This document describes a proposal for adding LoRaWAN support in RADIUS. The purpose is to integrate the LoRaWAN network join procedure with an Authentication, Authorization and Accounting (AAA) infrastructure based on RADIUS.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 3, 2017.

Table of Contents

1. Introduction

Low Power Wide Area Network (LP-WAN) groups several radio technologies that allow communications with nodes far from the central communication endpoint (base station) in the range of kilometers depending on the specifics of the technology and the scenario. They are fairly recent and the protocols to manage those infrastructures are in continuous development. In some cases they may not consider aspects such as key management or directly tackle scalability issue in terms of authentication and authorization. The nodes to be authenticated and authorized is expected to be considerably high in number. One of the protocols that provide a complete solution is LoRaWAN [LoRaWAN]. LoRaWAN is a MAC layer protocol that use LoRa as its physical medium to cover long range (up-to 20 km depending on the environment) devices. LoRaWAN is designed for large scale networks and currently has a central entity

called Network Server which maintains a pre-configured key named AppKey for each of the devices on the network. Furthermore, session keys such as NwkSKey and AppSKey used for encryption of data messages, are derived with the help of this AppKey. Since each service provider would operate their Network Server individually, authenticating the devices becomes a tedious process because of inter-interoperability or the roaming challenges between the operators. An illustration of the LoRaWAN architecture can be seen in figure Figure 1. As we know the AAA infrastructure provides a flexible, scalable solution. They offer an opportunity to manage all these processes in a centralized manner as happens in other type of networks (e.g. cellular, WiFi, etc...) making it an interesting asset when integrated into the LoRaWAN architecture.

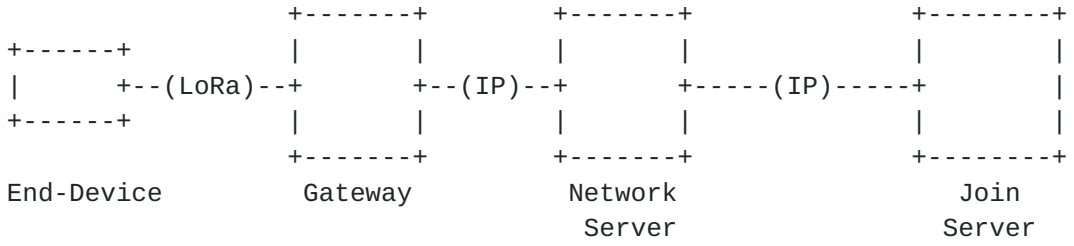


Figure 1: LoRAWAN Architecture

The End-Device communicates with the Gateway by using the LoRa modulation. The Gateway acts as a simple transceiver, which forwards all data do the Network Server, which performs the processing of the frames, network frame authentication (MIC verification), and which serves as Network Access Port. This document describes a way to use standard RADIUS servers as a Join Server, and to use the RADIUS protocol for the interaction between the Network Server and the Application Server. This integration is illustrated in figure Figure 2

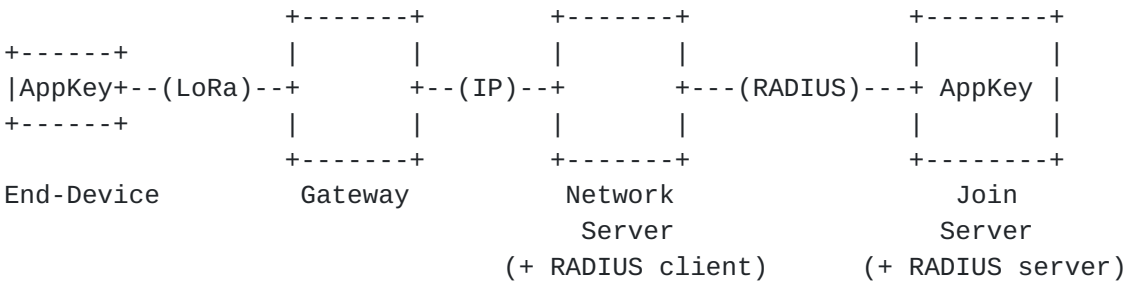


Figure 2: LoRAWAN Architecture with AAA and RADIUS authentication. End-Device and RADIUS server have a shared secret - the AppKey, which is used to derive the session keys (NwkSKey and AppSKey).

The document describes how LoRaWAN join procedure is integrated with AAA infrastructure using RADIUS [RFC2865] by defining the new attributes needed to support the LoRaWAN exchange.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. LoRaWAN support in RADIUS

Regarding the overall functionality, the RADIUS LoRaWAN support defines the new Attributes needed for the management of the join procedure. The Network Server will implement a RADIUS client supporting this specification and therefore, it MUST implement the RADIUS attributes for this service. The NAS-Port-Type specifying the type of port on which the Network Server is authenticating the End-Device in this case MAY be 18 (Wireless - Other) or a new one specifically assigned for LoRaWAN (TBD.).

3. LoRaWAN Overview

3.1. Introduction

The LoRAWAN specification defines how the MAC and PHY layer will be used with the LoRa radio technologies. It defines a process by which the smart objects can securely join the network in an authenticated way and exchange application information ciphered and integrity protected. The focus of this document is to extend how the process of joining is performed by the specification including a AAA infrastructure (RADIUS) to accomplish this. Next we review how the keys, and each message is used in the joining procedure. Then we elaborate some assumptions to design the integration of AAA in the joining procedure possible.

3.2. LoRaWAN join procedure Key Material

The LoRaWAN specification describes 3 keys involved in the joining procedure. One as a root key that will be used to generate the other two, which will be used to secure the message exchanges after the joining procedure success. The AppKey key used to derive the other two keys, NwkSKey and AppSKey:

- The AppKey is an AES-128 application specific key assigned by the owner of the application. This key is derived from an application-specific root key that is only known to the

application owner and is stored in each device and in the Join Server that will perform the authentication.

- The NwkSKey is a network session key that is specific to each End-Device. It is shared between the Network Server and the End-Device and used to calculate and verify the Message Integrity Code (MIC) for each data message, between both entities. Furthermore, it is used to cipher and decipher the payload of MAC-only data message.

- The AppSKey is an application session key specific to each End-Device. It is in charge of ciphering and deciphering the payload of application-specific data messages and is also used to calculate and verify the MIC that may be added to the payload of application-specific data messages.

3.3. LoRaWAN joining procedure

The LoRaWAN joining procedure, as described in the LoRaWAN Specification 1.0 [LoRaWAN], consists on one exchange. The first message of this exchange is called join-request (JR) message and is sent from the End-Device to the Network Server containing the AppEUI and DevEUI of the End-Device with a nonce of 2 octets called DevNonce. Figure 3 summarizes the format.

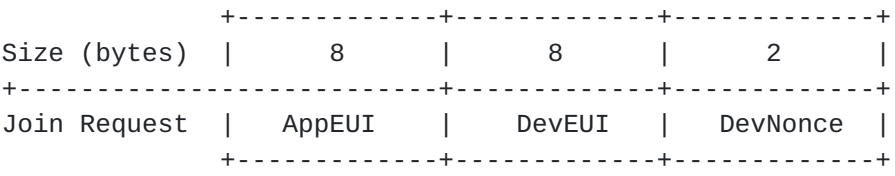


Figure 3: Join Request Message

In response to the join-request, the other endpoint will answer with the join-accept (JA) (Figure 4) if the End-Device is successfully authenticated and authorized to join the network. The join-accept contains a nonce (AppNonce), a network identifier (NetID), an End-Device address (DevAddr), a delay between the TX and RX (RxDelay) and, optionally, the CFList (see LoRaWAN specification [LoRaWAN] section 7).

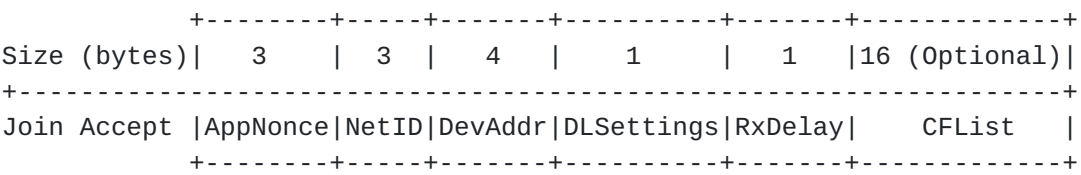


Figure 4: Join Accept Message

Next, we enumerate and describe each field involved in the join procedure message exchange.

- AppEUI: Global application ID in IEEE EUI64 to uniquely identify the application provider.

- DevEUI: Global End-Device ID in IEEE EUI64 to uniquely identify the End-Device

- DevNonce: A random value.

- AppNonce: A random value or some kind of unique ID provided by the Network Server. This value can be also generated by the AAA server, in case the network server wants to rely on the AAA server pseudo random number generation. For this, the AppNonce would be empty (set to zero), signaling the AAA server it has to generate the AppNonce.

- NetID: A network identifier

- DevAddr: A 32 bit identifier of the End-Device in the current network. It is composed of the Network ID and the Network Address.

- DLSettings: 8 bits containing the down-link configuration.

- RxDelay: 8 bits containing the delay between TX and RX.

- CFList (Optional): Channel frequency list.

3.4. LoRaWAN Key Derivation

The keys NwkSKey and AppSKey are derived from the AppKey in both the Join Server and the End-Device according to the LoRaWAN specification [LoRaWAN] as follows:

Derivation of the NwkSkey:

NwkSKey = aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad16)

Derivation of the AppSkey:

AppSKey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad16)

Note: The pad16 function appends octets of containing "zero" so that the length of the data is a multiple of 16.

4. Integration Overview

4.1. Mapping LoRaWAN Entities to AAA Infrastructure

In the current specification of LoRaWAN [LoRaWAN], there is no explicit reference to an external entity to which the Network Server can go to authenticate the End-Device. However, ongoing work related to LoRaWAN, such as the work in the LoRa Alliance [LoRaAllianceSecurity] sketches the use of a new entity, the Join Server, that will be in charge of performing the authentication. This separation of responsibilities is also the aim of our work, where the Join Server acts as an external AAA server in a AAA infrastructure using RADIUS as the protocol to communicate the Network Server and the Join Server. Further, it is under consideration the distribution of the AppSKey to a target application server instead of the Network Server. Therefore, the Join Server would need another protocol to deliver the AppSKey. Another RADIUS interface could be used for this purpose, though this I-D focuses on the joining procedure so far.

4.2. Assumptions

For the integration of LoRaWAN joining procedure with RADIUS next we describe some assumptions regarding the LoRaWAN specification. The first is that the AppKey is only shared between the AAA server (Join Server) and the End-Device. The outcome of the successful join procedure (i.e. NwkSKey and AppSKey) are sent from the AAA server to the network-server. This allows for the End-Device to exchange message with the network-server, once the join procedure is finished, as specified in LoRaWAN [LoRaWAN].

4.3. Protocol Exchange

The join procedure between the End-Device and the network-server entails one exchange consisting on a join-request message and a join-response message. In RADIUS the network-server implements a RADIUS client to communicate with the Join Server, which act as AAA Server.

The protocol exchange is done in the following steps:

1. The End-Device sends the join-request message to to the Network Server.

2. Upon reception of the LoRaWAN join-request message, the Network Server creates a RADIUS Access-Request message, with the Join-Request attribute containing the original message from the End-Device, and the Join-Answer Attribute with all the fields of a join-answer message except for the MIC, which will be calculated

by the AAA Server (Join Server), since is the one that holds the AppKey.

3. Once the AAA Server authenticates and authorizes the End-Device, sends back the Join-Answer with the MIC generated as specified by the LoRaWAN specification. Furthermore, as a consequence of a successful join procedure, the AppSKey (optional) and NwkSKey are generated and sent along in AppSKey and NwkSKey Attributes respectively.

4. The Network Server receives the Access-Accept (if successful), obtains the content of the Join-Request attribute and sends it to the End-Device, storing in association with that End-Device the NwkSKey and the AppSKey.

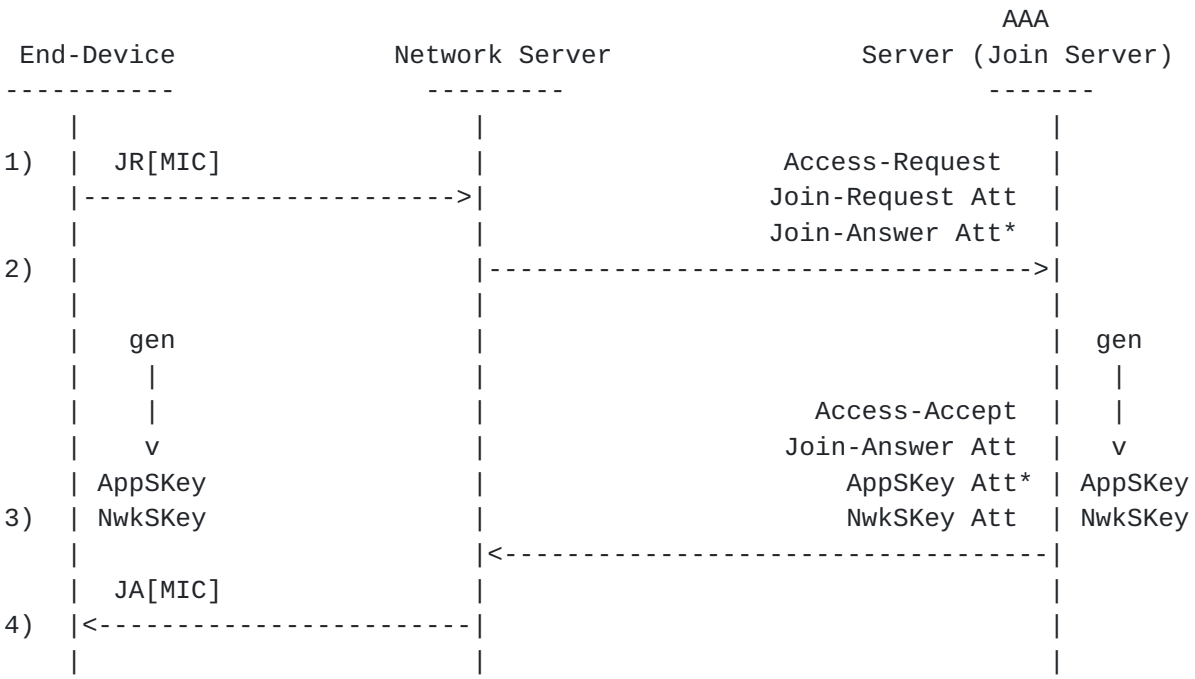
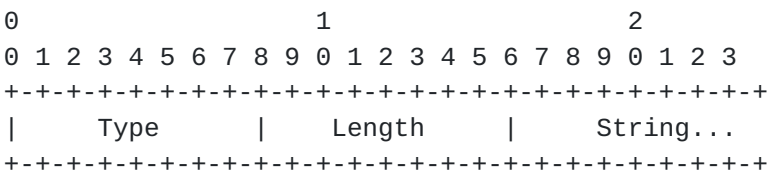


Figure 5: Protocol

4.3.1. Join-Request Attribute

Description

This Attribute contains the original Join-Request message. This attribute will only appear in the Access-Request message. A summary of the Join-Request attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for Join-Request

Length

18

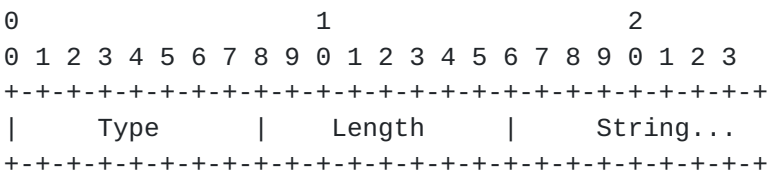
String

The String field contains an octet string with the Join-Request message as received over the network, such as defined in [LoRaWAN].

4.3.2. Join-Answer Attribute

Description

This Attribute is used in both RADIUS Access-Request and RADIUS Access-Accept messages. In the first case, it contains the Join Answer message with all the needed values filled by the network-server except the MIC (this fact is marked with an *). With these values, the Join Server (AAA server) that holds the AppKey is able to create the MIC and compose the final Join Answer message. In the second case, it contains the Join Answer with the MIC generated by the Join Server (AAA server). A summary of the Join-Answer attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for Join-Answer

Length

28

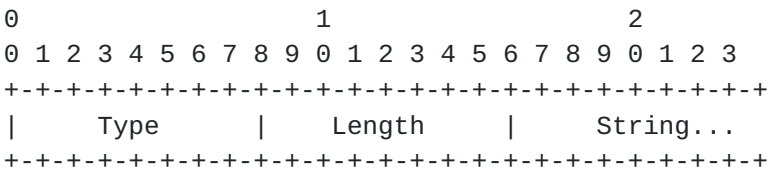
String

The String field contains an octet string with the Join-Answer as received over the network , as defined in [LoRaWAN].

4.3.3. AppSKey Attribute

Description

This Attribute contains the AppSKey, an application session key specific for the End-Device. This attribute is optional, and will only appear in the RADIUS Access-Accept message. A summary of the AppSKey attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for AppSKey

Length

16+

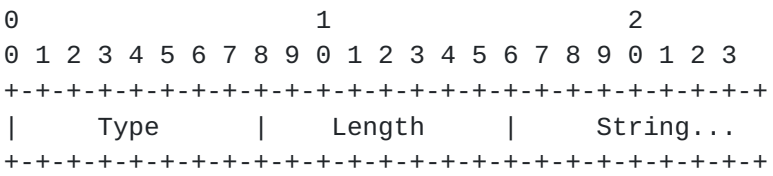
String

The String field contains an octet string containing the Application Session Key, as defined in [LoRaWAN].

4.3.4. NwkSKey Attribute

Description

This Attribute contains the NwkSKey, an network session key specific for the End-Device. This attribute will only appear in the Access-Accept message. A summary of the NwkSKey attribute format is shown below. The fields are transmitted from left to right.



Type

TBD. for NwkSKey

Length

16+

String

The String field contains the octet string of the Network Session Key , as defined in [LoRaWAN].

4.3.5. Table of Attribute

Request	Accept	Reject	Challenge	#	Attribute
1	0	0	0	TBD.	Join-Request
1	1	0	0	TBD.	Join-Answer
0	0-1	0	0	TBD.	AppSKey
0	1	0	0	TBD.	NwkSKey
Request	Accept	Reject	Challenge	#	Attribute

Figure 6: Attributes Table

5. Open Issues

With the purpose of extending the authentication process via AAA infrastructures, and concretely, RADIUS, we have faced a question regarding the relationship between the AppEUI associated to the organization operating the Join Server and the realm used by RADIUS to route the AAA information to the AAA Server (Join Server) of that organization.

In particular, the Network Server knows the AppEUI included in the Join Request, but it needs to discover the realm (Fully Qualified Domain Name) that corresponds to that organizations ID to be able to communicate with the concrete RADIUS server.

NOTE: One option MAY be to use the DNS system to provide the FQDN associated to an AppEUI (which is an EUI64 address). The mapping using DNS to find out the domain name associated to an EUI64 address has been described in [RFC7043]. However, we would need the inverse process. Nevertheless, this needs further discussion.

6. Security Considerations

In the LoRaWAN 1.0 specification, the AppSKey and NwkSKey are not sent over the network, they are derived in each of the endpoints that communicate, namely the End-Device and the Network Server. In this document we propose relegating the responsibility of deriving the Network Session Key and Application Session Key to the RADIUS server (the Join Server). These session keys need to be sent to the Network Server and if necessary to the application server.

To send the messages over the network between the RADIUS server and the RADIUS client (in this case the Network Server). How to provide confidentiality to the key distributed is outside the scope of this document, nevertheless RadSec (RFC6614) or extensions such as those defined in RFC 6218 may be considered to protect the distribution.

The AAA framework and its key management features become increasingly important as the use case of LoRaWAN adds functionality and complexity. This is the case for having the Application Server and Network Server as separate entities and each receive its keys. Although the utility is apparent in that specific case, it has to be considered in any other future use-case that may require key management and key distribution. Another point in favor of using AAA can be also appreciated since the modifications required by this proposal does not imply the modification of the protocols of the constrained link, but the unrestricted network that is used to manage LP-WAN.

7. Proof of concept implementation

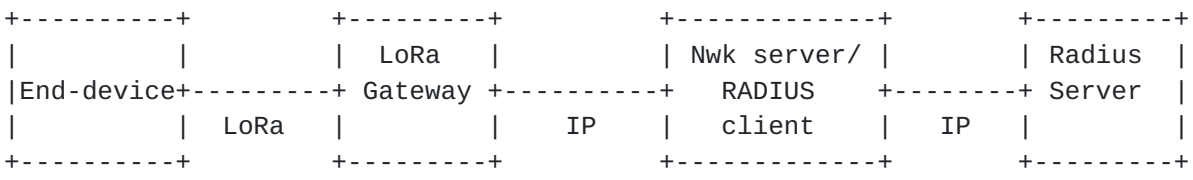
The proof of concept is implemented using the Go programming language, that is well suited for the development of web servers or a network servers as in this case.

The implementation of the network server is from [LoRaSERVER] which is tailored with the features of a RADIUS Client and the RADIUS server implementation from [RADIUSGo] that is modified to handle LoRaWAN attributes.

The LoRa end-device, pre-configured with AppKey, from Nemeus [MK002] is a USB key that can be controlled by UART (AT command) through USB interface. A JAVA application installed on a Linux machine is used to send control and data messages from the End-Device.

The LoRa Gateway is from EXPEMB [EXPEMB] which uses the packet forwarder to forward the LoRa packets to the LoRa Network Server. The Network Server is run in a docker container on a Linux machine transfers the LoRa packets into the RADIUS attributes to be sent to the RADIUS server. For now, the packets are sent to the default RADIUS server but in the future this would be changed as per the discussion in Section 5 in order to redirect the RADIUS request to appropriate RADIUS server.

The RADIUS server is run in a docker container on a Linux machine which contains the mapping between the DevEUI of the End-Device and the AppKey. This AppKey from the map along with the received LoRa attributes is used to derive the session keys, NwkSKey and AppSKey, in the RADIUS server. These keys are transported as RADIUS attributes back to the network server.



A successful authentication would result in the session keys, NwkSKey and AppSKey, being visible on the network server that can be viewed using a web interface and the DevAddr being acquired by the End-Device from the Join Accept Lora message. Running Wireshark on the interface between RADIUS server and the Network Server shows the RADIUS packets with the LoRa attributes.

To simplify the design and implementation, we opted for creating one RADIUS Attribute per message, instead of per each field within the message since only the authenticating module responsible for the Join Procedure in the current network server is delegated to the AAA server and the AAA server would be able to obtain the required fields from this single attribute, i.e either JoinRequest or JoinAccept message. This design choice would follow the RADIUS guidelines given in [RFC6158] identifying it as string for being an opaque encapsulation of data structures defined outside RADIUS. Creating an attribute per field, would be useful in case the AAA infrastructure would change its behavior depending on the specific content of one or more of the fields contained in the message. This could be the case when the LoRaWAN use case becomes more complex and add more functionality.

As future work, we intend to implement the proof of concept in FreeRADIUS

8. Acknowledgments

This work has been possible partially by the SMARTIE project (FP7-SMARTIE-609062 EU Project) and the Spanish National Project CICYT EDISON (TIN2014-52099-R) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

We also wanted to thank for the comments received about this document by Sri Gundavelli, Yeoh Chun-Yeow, Alan DeKok, Stephen Farrell and Mark Grayson.

9. IANA Considerations

In this document we define 4 new RADIUS Attributes that would need actions from IANA to assign the corresponding numbers.

Number	Name	Reference
TBD	Join-Request	Section 4 of this document
TBD	Join-Answer	Section 4 of this document
TBD	AppSKey	Section 4 of this document
TBD	NwkSKey	Section 4 of this document

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC6158] DeKok, A., Ed. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, DOI 10.17487/RFC6158, March 2011, <http://www.rfc-editor.org/info/rfc6158>.

[RFC7043] Abley, J., "Resource Records for EUI-48 and EUI-64 Addresses in the DNS", RFC 7043, DOI 10.17487/RFC7043, October 2013, <http://www.rfc-editor.org/info/rfc7043>.

10.2. Informative References

[EXPEMB] EXPEMB, E., "LoRa MultiConnectivity Service Gateway - Last Accessed:", July 2016, <www.expemb.com/en/product/multi%E2%80%90connectivity-service-gateway-sgwmc%E2%80%90x86lr%E2%80%9012132/>.

[LoRaAllianceSecurity] Girard, P., "LoRaWAN - SECURITY a comprehensive insight - Online Resource: Last Accessed", July 2016, <http://portal.lora-alliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=1085>.

[LoRaSERVER] Acklio, A., "LoRa Server", July 2016, <http://www.ackl.io>.

[LoRaWAN] Sornin, N., Luis, M., Eirich, T., and T. Kramp, "LoRa Specification V1.0", January 2015, <https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf>.

[MK002] Nemesus, N., "MK002-xx-EU - Last Accessed:", July 2016, <http://www.nemeus.fr/en/mk002-usb-key>.

[RADIUSGo]
bronze1man, B., "Radius: A golang radius library - Last Accessed:", July 2016, <https://github.com/bronze1man/radius>.

Authors' Addresses

Dan Garcia-Carrillo (Ed.)
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 78 82
Email: dan.garcia@um.es

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
Email: rafa@um.es

Arunprabhu Kandasamy
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: arun@ackl.io

Alexander Pelov
Acklio
2bis rue de la Chataigneraie
35510 Cesson-Sevigne Cedex
France

Email: a@ackl.io