System Specification

enocean alliance
Building Smarter Connectivity

Security of EnOcean Radio Networks

V 4.03

San Ramon, CA, USA
Original Release 1.0, 2012

Executive Summary

This document is owned by the Technical Working Group (TWG) of the EnOcean Alliance. It is maintained and will be progressed within the authority of the chairman of the TWG.

Changes to this document will have to be proposed to the TWG for decision. A team of senior engineers will then act upon request by the TWG.

REVISION HISTORY

Ver.	Editor	Change	Date
1.0	MF	Creation	05.03.2012
1.1	MF	Inclusion of AES CBC. Secure telegrams with/without non-secure R-ORG.	03.07.2012
1.2	MF	Page numbering	31.07.2012
1.3	AP	Removing ARC4 and editorial changes	31.10.2012
1.4	MH	Editorial on CMAC computing	22.11.2012
1.5	MH	Added PSK Teach-in and VAES extensions above 16 bytes.	17.06.2013
1.6	MF	Abstraction from telegram to message	03.07.2013
1.7	MF	Teach-in unidirectional/bidirectional procedure described	10.07.2013
1.8	MF	RLC synchronization. Message-to- ERP1 and ERP2 conversion	11.07.2013
1.9	MF	PSK CRC explained. Description of Teach-in procedure improved	26.07.2013
2.0	MH	Added High Security definition - renamed public key to VAES INIT Vector - CMAC in Teach-in telegram eliminated	11.08.2014
2.1	FG	Added Test vectors	09.11.2016
2.2	MF	Test vectors encryption and CMAC calculations described in detail	12.07.2017
2.3	MH	Best practice examples, Secure Chaining, 32 bit RLC. Removed Mutual authentication with RLC as NONCE from high security. Changed RLC description.	09.05.2018
2.4	TM	Add Secure Chaining example, updated the SEC-CDM mechanism	27.11.2018
2.5	TM, MH	Based on Alliance TTG Feedback: Recommendations Chapter changed to implementation aspects. Formatting, and wording. Removed AES CBC.	15.01.2019
2.51	MH, FS	Re-enabling the SLF RLC option 0b011 (3) 16 bit RLC explicit with window algorithm and roll over - PRELIMINARY	17.04.2019
2.6	AP	Replaced Secure Chaining test vector in A.4.3	03.12.2020
3.0	AP, MKa	Complete rework of whole document	22.02.2023

3.01	AP	Typos corrected	19.04.2023
3.02	AP	Typo in 3.6.2 corrected	24.05.2024
4.00	MKA	Complete document review and update Addition of CCM security mode Addition of definition for ERP2 variant Removed PSK-based security key update Review of existing test vectors	19.09.2025
4.01	MKA, RS, JB, MH, EO	Resolved feedback from TG document review Improved document structure Improved description of chaining (SEC_CDM) Added examples for UTE, VLD, ADT MSC and SEC_DAT Added examples for ERP2 Added examples for CCM mode	29.10.2025
4.02	MKA, LR	Final version for release following TWG review and prototype implementation	17.07.2026
4.03	MKA	Comment resolution from TWG review	18.09.2026

Copyright © EnOcean Alliance Inc. 2012- 2026. All rights Reserved.

DISCLAIMER

The information within this document is the property of the EnOcean Alliance and its use and disclosure are restricted. Elements of the EnOcean Alliance specifications may also be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of the EnOcean Alliance.) The EnOcean Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. This document and the information contained herein are provided on an "as is" basis and the EnOcean Alliance disclaims all warranties express or implied, including but not limited to

(1) any warranty that the use of the information herein will not infringe any rights of third parties (including any intellectual property rights, patent, copyright or trademark rights, or
(2) any implied warranties of merchantability, fitness for a particular purpose, title or non-infringement.

In no event will the EnOcean Alliance be liable for any loss of profits, loss of business, loss of use of data, interruption of business, or for any other direct, indirect, special or exemplary, incidental, punitive or consequential damages of any kind, in contract or in tort, in connection with this document or the information contained herein, even if advised of the possibility of such loss or damage. All Company, brand and product names may be trademarks that are the sole property of their respective owners.

The above notice and this paragraph must be included on all copies of this document that are made.

The EnOcean Alliance "Security of EnOcean Radio Networks" Specification is available free of charge to companies, individuals and institutions for all non-commercial purposes (including educational research, technical evaluation and development of non-commercial tools or documentation.)

This specification includes intellectual property („IPR") of the EnOcean Alliance and joint intellectual properties („joint IPR") with contributing member companies. No part of this specification may be used in development of a product or service for sale without being a participant or promoter member of the EnOcean Alliance and/or joint owner of the appropriate joint IPR. EnOcean Alliance grants no rights to any third party IP, patents or trademarks.

These errata may not have been subjected to an Intellectual Property review, and as such, may contain undeclared Necessary Claims.

EnOcean Alliance Inc.
2603 Camino Ramon, Suite 200
San Ramon, CA 94583, USA
Graham Martin
Chairman & CEO EnOcean Alliance

1 Introduction 10
1.1 Abbreviations 10
2 ERP Architecture 11
2.1 ERP Topology 11
2.2 ERP Protocol Stack 12
2.3 ERP Telegram Format 13
2.3.1 ERP Telegram Fields 13
2.3.2 ERP1 Telegram Structure 14
2.3.2.1 ERP1 Broadcast Telegram Structure 14
2.3.2.2 ERP1 Addressed Telegram Structure 14
2.3.3 ERP2 Telegram Structure 14
2.3.3.1 ERP2 HEADER Field 15
2.4 ERP Device Addressing 16
2.5 ERP Data Representation 17
2.5.1 Common R-ORG Types 18
2.5.2 Security Processing for ERP2 Messages 19
2.5.3 Secure Message Types 19
3 Security Use Cases 20
3.1 Occupancy Reporting 20
3.2 Access Control 20
3.3 Required Security Functionality 20
4 Security Architecture 21
4.1 Security Modes 21
4.2 Security Key (K) 21
4.2.1 Security Key Assignment 21
4.2.2 Security Key Update and Derivation 21
4.2.3 Security Key Usage in Bi-directional or Multi-party Communication 22
4.2.4 Security Key Update 22
4.2.5 Security Key Derivation 22
4.3 Rolling Code (RLC) 23
4.3.1 Internal RLC Size 23
4.3.2 Transmitted RLC Size 23
4.4 Encryption / Decryption 24
4.5 Message Authentication Code (MAC) 24
4.5.1 MAC Calculation for VAES 24
4.5.2 MAC Calculation for CCM 25
4.6 Security Level Format 25
4.6.1 SECURITY_MODE 25
4.6.2 MAC SIZE 26
4.6.3 RLC SIZE 26
4.6.4 Summary of Security Level Formats and Corresponding SLF Values 27
4.7 Guidance on Usage of Security Parameters 28
4.7.1 Standard Format 28
4.7.2 Standard Format for CCM with Reduced RLC Size 28

4.7.3 Format for Energy-constrained Devices using VAES with CMAC 29
4.7.4 Format for Ultra-low Power Devices using VAES with CMAC 29
5 Security Setup 30
5.1 Security Setup using Out-of-band Mechanisms 30
5.2 Security Setup using QR Code 30
5.3 Security Setup using Security Teach-in Messages 31
5.3.1 Format of Security Teach-in Messages 31
5.3.2 Security Teach-in Message with PSK 32
5.3.2.1 Security Teach-in Message with PSK in VAES mode 32
5.3.2.2 Security Teach-in Message with PSK in CCM mode 33
5.3.3 Security Setup Process 34
5.3.3.1 Uni-directional Security Setup 34
5.3.3.2 Bi-directional Security Setup 35
6 Security Processing 36
6.1 Encryption and Decryption 36
6.2 Authentication 37
6.3 RLC Management 38
6.3.1 Transmission 38
6.3.2 Reception 38
6.3.3 RLC Tracking 39
6.3.3.1 Obtaining the Initial RLC Value 39
6.3.3.2 RLC Tracking for Reduced RLC Size Transmission 40
6.3.3.3 RLC Tracking for Transmission with Implicit RLC 41
6.4 Segmentation (Chaining) and Reassembly (De-chaining) 43
6.4.1 Message Format 44
6.4.1.1 CHAIN_CTRL 44
6.4.1.2 First Telegram in a Secure Message Chain 45
6.4.1.3 Last Telegram in a Secure Message Chain 46
6.4.1.4 Intermediate Telegrams in a Secure Message Chain 47
6.5 ERP1 versus ERP2 Security Processing 47
7 Security Algorithms 48
7.1 VAES Encryption and Decryption 48
7.1.1 Input data 48
7.1.2 VAES encryption process 49
7.1.3 VAES decryption process 50
7.1.4 VAES INIT VECTOR 50
7.2 CMAC Authentication 51
7.2.1 Input Data 51
7.2.2 CMAC calculation for multiples of 16 bytes 52
7.2.3 CMAC calculation for non-multiples of 16 byte 52
7.2.4 Key derivation of K1 and K2 from the Security Key 53
7.3 CCM Algorithm 54
7.3.1 Terminology 54
7.3.2 Input Data 55

7.3.2.1 Nonce N 56
7.3.2.2 Example of Nonce N for a Broadcast Message 56
7.3.2.3 Example of Nonce N for an Addressed Data Message (ADT) 56
7.3.2.4 Authenticated Data *a* 57
7.3.2.5 Data Telegrams 57
7.3.2.6 Security Teach-in Messages with PSK 58
7.3.3 Authenticated and Encrypted Data *e* 58
7.3.3.1 Authenticated and Encrypted Data *e* for Data Messages 58
7.3.3.2 Authenticated and Encrypted Data *e* for Security Teach-in Messages with PSK 58
7.3.4 MAC Size (Size of Authentication Signature *U*) 58
7.3.5 Configuration Parameters 59
7.3.5.1 Encoded Configuration Parameters 59
7.4 CCM Authentication 60
7.4.1 B Block Structure 60
7.4.1.1 B0 Flag 60
7.4.1.2 B0 Block 60
7.4.1.3 B Blocks for Authenticated Data *a* 61
7.4.1.4 B Blocks for Authenticated and Encrypted Data *e* 62
7.4.2 Authentication Processing 63
7.5 CCM Encryption and Decryption 64
7.5.1 A Flag 64
7.5.2 A0 … Ai Blocks 64
7.5.3 Key Stream S 65
7.5.4 Encryption and Decryption of R-ORG and DATA 65
7.5.5 Encryption and Decryption of the Authentication Signature 66
7.6 Key Derivation 67
7.6.1 Input Parameters 67
7.6.2 Internal Parameters 67
7.6.3 Key Derivation Process 68
8 Secure Message Format 69
8.1 Secure Data Message Types 69
8.1.1 Secure Broadcast Message without R-ORG 69
8.1.2 Secure Broadcast Message with R-ORG 70
8.1.3 Secure Addressed Telegram with R-ORG 70
8.1.4 Secure Binary Data Transport Message (SEC_DAT) 71
8.2 Secure Chained Data Messages 72
8.2.1 First Telegram 72
8.2.2 Intermediate Telegrams 72
8.2.3 Last Telegram 73
8.3 Security Teach-in Messages 74
8.3.1 Security Teach-in Message Format 74
8.3.2 Teach-in Info 74
8.3.3 Format of Teach-in Info Field 74
8.3.3.1 Teach-in Info for First Telegram of a Non-PTM Device 75
8.3.3.2 Teach-in Info for First Telegram of a PTM Device 76

8.3.3.3 Teach-in Info for Second Telegram 76
8.3.4 Security Teach-in Message without Chaining 76
8.3.5 Security Teach-in Message with Chaining 77
8.3.6 Security Teach-in Message with PSK 78
9 References 80
Annex A - Secure Message Examples 81
A.1 Secure PTM Message 81
A.1.1 Example parameters 81
A.1.2 Security Teach-in Message 82
A.1.2.1 VAES Mode 82
A.1.2.2 CCM Mode 83
A.1.3 Data Message Processing in VAES Mode 84
A.1.3.1 Data Encryption (VAES) 84
A.1.3.2 Data Authentication (CMAC) 85
A.1.3.3 Resulting Telegrams 86
A.1.4 Data Message Processing in CCM Mode 87
A.1.4.1 CCM Parameters 87
A.1.4.2 Block Calculation 88
A.1.4.3 Authentication Signature Processing 89
A.1.4.4 Encryption Processing 89
A.1.4.5 Resulting Telegrams 90
A.2 Secure 4BS Sensor Message 91
A.2.1 Example parameters 91
A.2.2 Security Teach-in (SEC_TI) Message 91
A.2.2.1 VAES Mode 91
A.2.2.2 CCM Mode 92
A.2.3 Data Message Processing in VAES Mode 93
A.2.3.1 Data Encryption (VAES) 93
A.2.3.2 Data Authentication (CMAC) 94
A.2.3.3 Resulting Telegrams 95
A.2.4 Data Message Processing in CCM Mode 96
A.2.4.1 CCM Parameters 96
A.2.4.2 Block Calculation 97
A.2.4.3 Authentication Signature Processing 98
A.2.4.4 Encryption Processing 98
A.2.4.5 Resulting Telegrams 99
A.3 Secure Chained VLD Data Message (SEC_CDM) with UTE 100
A.3.1 Example parameters 100
A.3.2 Security Teach-in Message 101
A.3.2.1 VAES Mode 101
A.3.2.2 CCM Mode 102
A.3.3 UTE (EEP Teach-in) Message 103
A.3.3.1 UTE Message Content 103
A.3.3.2 UTE Message Processing in VAES Mode 104
A.3.3.3 UTE Message Processing in CCM Mode 108

A.3.4 VLD Data Message 111
A.3.4.1 Data Message Processing in VAES Mode 112
A.3.4.2 Data Message Processing in CCM Mode 116
Annex B - Key Derivation Example 120
B.1 Input Parameters 120
B.2 Key Derivation Process 120
Annex C - Legacy SLF Definition 121
C.1 RLC_TYPE Options 122
C.2 CMAC_TYPE Options 122
C.3 ENCRYPTION_TYPE Options 122

1 Introduction

This document specifies the security concept for the EnOcean Radio Protocol 1/2 (ERP1/2). This concept was specially designed for devices powered by energy harvesting and is based on the ERP. The objective of the design was to keep the energy requirements and µC resources for the implementation of the security as low as possible.

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 [1].

1.1 Abbreviations

Abbreviation	Description
AES	Advanced Encryption Standard
CBC	Cipher Block Chaining (US-Patent 4074066, 1976)
CCM	Counter with CBC MAC (as specified by RFC 3610)
CMAC	Cipher Based Message Authentication Code (as specified by RFC 4493)
EEP	EnOcean Equipment Profile
ERP / ERP 1 / ERP2	EnOcean Radio Protocol version 1 or EnOcean Radio Protocol version 2. If no number is used, then both versions are meant.
EURID	EnOcean Unique Radio Identifier (also known as device ID or radio address)
K	Security Key
LSB / LSb	Least significant byte / Least significant bit
MAC	Message Authentication Code (Authentication Signature)
MSB / MSb	Most significant byte / Most significant bit
PSK	Pre-shared Security Key
RLC	Rolling Code
R-ORG	Radio-Organization (defining the type of an EnOcean radio message)
R-ORG-S	R-ORG for security processing
SLF	Security Level Format (encoding the parameters used for secure communication)
VAES	Variable AES

Table 1. Abbreviations used in this document.

2 ERP Architecture

This specification defines security mechanisms for data exchange using the EnOcean Radio Protocol (ERP). ERP is a radio protocol optimized for the exchange of radio telegrams at lowest power consumption and with a very high quality of service. These goals are achieved by using small telegram sizes in conjunction with redundant transmissions on regulated radio spectrum in the 800 / 900 MHz radio bands.

Common applications of ERP include building automation, industrial sensing and IOT sensor applications.

2.1 ERP Topology

ERP provides a flexible topology for a wide variety of use cases.

The most common topology of ERP systems is uni-directional broadcast where one transmitter (for instance a light switch or a sensor) provides status updates (for instance button events or measured temperature) to one or several receivers (for instance controllers, actuators or gateways) that are within radio range.

The logical link between transmitter and receiver is established on the receiver side by configuring the receiver to accept and react to telegrams from specific transmitters which are identified by their Sender ID. Receivers that are not configured to accept and react to telegrams from a transmitter will ignore telegrams originating from that transmitter.

It is possible for the transmitter to indicate the intended receiver of a telegram by providing the Destination ID as part of the telegram transmission. Such telegrams are called Addressed Data Telegram (ADT) and can be used for instance for the configuration of a specific device.

Devices may respond to received telegrams, for instance when queried for their status or query the other device for its status. This sequence of request and response (or transmission and reception) is called bi-directional communication.

Figure 1 below provides an example of such devices communicating using ERP.

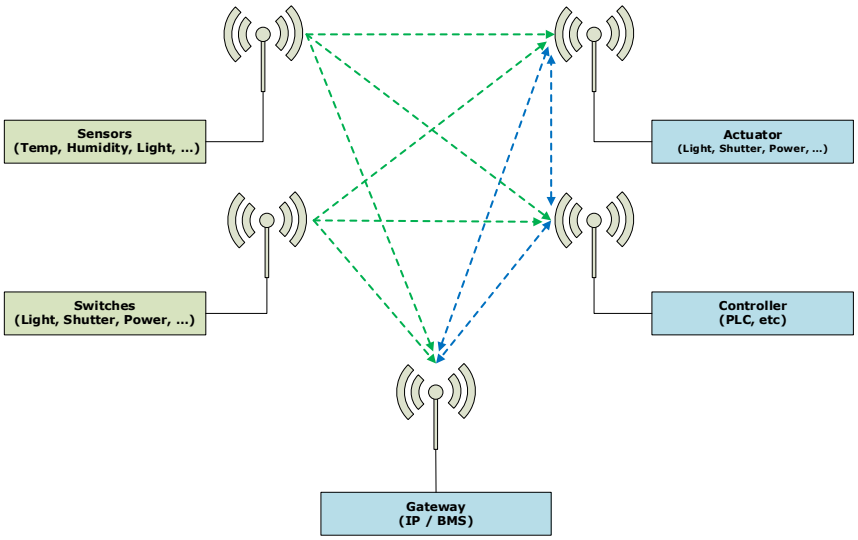


Figure 1. ERP system architecture

2.2 ERP Protocol Stack

ERP is implemented as protocol stack where different functionality is implemented at different, well-defined layers as follows:

- Data representation uses the definitions of EnOcean Equipment Profiles (EEP) to present data messages identified by data addresses to the application
- Security processing provides the required encryption, decryption, authentication and replay-protection mechanisms defined within this specification for secure data message exchange
- Segmentation and reassembly (also called chaining and de-chaining) distribute the content of one secure data message over two or more data telegrams if the content of the secure data message is too large to be transported using one radio telegram as defined by the ERP1 or ERP2 radio protocol
- Radio telegram transmission and reception according to the ERP1 (ISO14543-3-10) or ERP2 (ISO14543-3-11) standards are used for the wireless transport of data telegrams from one sender to one or more receivers

This specification defines the security processing and segmentation / reassembly mechanisms. The following information is provided for reference purposes:

- Chapter 2.3 describes radio communication and addressing using ERP
- Chapter 2.4 describes the addressing mechanisms used to identify the sender and – optionally – the intended destination of an ERP radio telegram
- Chapter 2.5 describes the data representation using EnOcean Equipment Profiles (EEP)

Figure 2 provides a simplified representation of the ERP protocol stack.

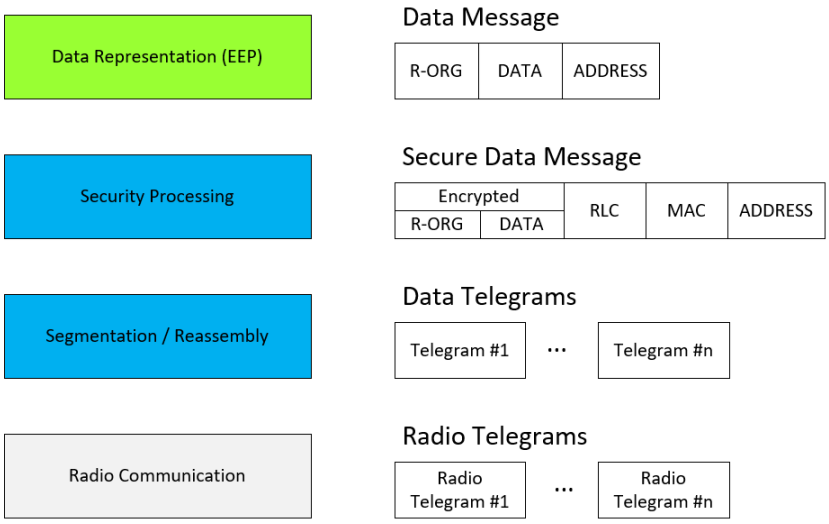


Figure 2. ERP protocol stack

2.3 ERP Telegram Format

ERP is implemented in two variants with slightly different telegram format:

- ERP1 as specified by ISO 14543-3-10 [2]
- ERP2 as specified by ISO 14543-3-11 [3]

ERP1 and ERP2 are commonly referred to as “ERP” throughout the remainder of the document unless functionality is specific to one variant.

The following chapters provide some details about ERP1 and ERP2 where necessary for understanding the context of this specification; consult the respective ISO standard referenced above for detailed information.

2.3.1 ERP Telegram Fields

ERP telegrams contain the following fields:

- R-ORG
  The R-ORG field identifies the high-level telegram type

- DATA
  The DATA field contains the telegram payload

- Security Fields
  Security fields are used for the encryption / decryption and authentication of ERP telegrams

- SENDER ID
  SENDER ID identifies the sender of a telegram

- DESTINATION ID
  Destination ID is an optional field that identifies the intended receiver of a telegram

- Control fields
  Control fields provide information related to the wireless transport of the telegram

The following chapters describe the resulting structure for common ERP1 or ERP2 telegram types. After that, ERP telegram fields that are not relevant for the security processing may be omitted.

2.3.2 ERP1 Telegram Structure

2.3.2.1 ERP1 Broadcast Telegram Structure

ERP1 telegrams combine these fields into the structure shown in Figure 3 below for the case of a broadcast data telegram.

ERP1 Broadcast Telegram				
R-ORG	DATA	SENDER ID	STATUS	CRC / HASH
1 byte	1 … 14-byte	4-byte	1-byte	1-byte

Figure 3. ERP1 broadcast telegram

2.3.2.2 ERP1 Addressed Telegram Structure

Figure 4 shows the structure of an ERP1 addressed data telegram.

ERP1 Addressed Telegram						
ADT	R-ORG	DATA	DESTINATION ID	SENDER ID	STATUS	CRC / HASH
0xA6	1-byte	1 … 9-byte	4-byte	4-byte	1-byte	1-byte

Figure 4. ERP1 addressed telegram structure

2.3.3 ERP2 Telegram Structure

ERP2 telegrams use a more generic structure shown in Figure 5 below. Fields marked with grey background are optional.

Note that the combination of HEADER and EXT_TYPE (if present) encodes the R-ORG that is used by the ERP2 data telegram.

HEADER additionally specifies the addressing mode which can be either broadcast or addressed. If addressed mode is used, then the DESTINATION ID field is present to indicate the intended receiver of a telegram; otherwise, this field is omitted.

The EXT_HEADER field specifies the repeater level and is only present if the telegram has been repeated.

ERP2 Telegram							
LENGTH	HEADER	EXT_HEADER	EXT_TYPE	SENDER ID	DESTINATION ID	DATA	HASH
1-byte	1-byte	0/1-byte	0/1-byte	4/6-byte	0/4-byte	Variable	1-byte

Figure 5. ERP2 telegram structure

2.3.3.1 ERP2 HEADER Field

For ERP2, the HEADER field shown below provides key information about the telegram.

HEADER							
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
ADDRESS CONTROL			EXTENDED HEADER	R-ORG			

Figure 6. ERP2 HEADER structure

The addressing mode is indicated by ADDRESS CONTROL field occupying the most significant 3-bit of the HEADER field. It is therefore not required to use R-ORG 0xA6: ADT in ERP2.

- 0b001: Broadcast (32-bit Sender ID, Destination ID is not present)
- 0b010: Addressed (32-bit Sender ID, 32-bit Destination ID is present)
- 0b011: Broadcast (48-bit Sender ID, Destination ID is not present)

The presence of EXT_HEADER is indicated by the EXTENDED HEADER field. EXT_HEADER indicates the repeater status and is only present if a telegram has been repeated.

- 0b0: EXT_HEADER is not present
- 0b1: EXT_HEADER is present

The 4-bit R-ORG field either directly specifies the most common R-ORG types or is set to 0xF to indicate that the R-ORG is specified by the EXT_TYPE field of the ERP2 telegram.

- 0x0 … 0xE: R-ORG encoded in this field (as specified in Table 2)
- 0xF: R-ORG provided by EXT_TYPE field (as specified in Table 2)

2.4 ERP Device Addressing

ERP devices are identified by their unique device ID, the so-called EnOcean Unique Radio Identifier (EURID). Each ERP device is assigned one EURID, and it is ensured that EURIDs are globally unique.

Additionally, ERP devices may use one or several Base Addresses selected from the Base Address Range which comprises a defined subset of the total address space. Base Addresses are not assured to be globally unique. The Virtual Broadcast ID may be used as Sender ID to indicate that the telegram does not have an intended destination.

Finally, ERP devices use the Broadcast ID as Destination ID to designate telegrams that are not intended for specific receivers (identified by their Destination ID). This is the default behaviour of ERP devices; therefore, the Broadcast ID is not present as Destination ID in ERP radio telegrams.

Figure 7 below illustrates the different address types.

Address	Usage
FF:FF:FF:FF	BROADCAST ID (Used as Destination ID)
FF:FF:FF:FE	VIRTUAL BROADCAST ID (Used as Sender ID)
FF:FF:FF:FD	RESERVED
FF:FF:FF:00	
FF:FF:FE:FF	BASE ID RANGE (Assigned by customer / application)
FF:80:00:00	
FF:7F:FF:FF	EURID RANGE (Device-unique, assigned by EnOcean Alliance)
00:00:00:01	
00:00:00:00	RESERVED

Figure 7. ERP Device Identification

2.5 ERP Data Representation

ERP devices may encode message data using EnOcean Equipment Profiles (EEP) standardized by EnOcean Alliance. The specific EEP used by a device is identified using three fields:

- R-ORG
  R-ORG identifies the high-level EEP type; common examples are listed in Table 2

- FUNCTION
  FUNCTION identifies the function group to which this EEP belongs, for instance the function group of temperature sensors within the 4BS high-level EEP type

- VARIANT (also called TYPE)
  VARIANT (also called TYPE) identifies the exact EEP variant within the EEP function group, for instance a sensor with a reported range of 0 °C … 40 °C defined within the EEP function group of temperature sensors

R-ORG, FUNCTION and VARIANT (TYPE) fields described above are combined into a 21-bit long EEP identifier as shown in Figure 8 below.

R-ORG	FUNCTION	VARIANT (TYPE)
8-bit (0x00 … 0xFF)	6-bit (0x00 … 0x3F)	7-bit (0x00 … 0x7F)

Figure 8. EEP identifier structure

The complete 21-bit EEP identifier is only transmitted during the initial teach-in (pairing) between devices. For subsequent transmissions, only the R-ORG part of the EEP identifier is transmitted. All transmissions from the same transmitter using the same R-ORG are considered to use the same EEP.

Use of more than one R-ORG is possible, for instance to allow transmission of both DATA and SIGNAL messages.

Note that for VLD messages using universal teach-in (UTE) procedure, the R-ORG, FUNC and VARIANT (TYPE) fields are all 8-bit long.

2.5.1 Common R-ORG Types

Table 2 below provides a list of common R-ORG and the corresponding encoding in the HEADER and EXT_TYPE fields of ERP2 messages.

R-ORG	ERP2 Encoding		Short	Message Type / Content
	HEADER[3:0]	EXT_TYPE		
Data and Teach-in R-ORG				
0xA5	0x2	N.A.	4BS	4-byte Sensor EEP Data
0xC5	0xF	0x00	SYS_EX	Remote Management Commands
0xD0	0x3	N.A.	SIG	Signal Data
0xD1	0x6	N.A.	MSC	Manufacturer-specific Data
0xD2	0x4	N.A.	VLD	Variable Length EEP Data
0xD4	0x5	N.A.	UTE	Universal Teach-in (for VLD)
0xD5	0x1	N.A.	1BS	1-byte Sensor EEP Data
0xD6	0xF	0xD6	DAT	Binary Data Transport
0xF6	0x0	N.A.	RPS	Rocker and Pushbutton Switch Data
Security R-ORG (R-ORG-S)				
0x30	0x7	N.A.	SEC	Secure RPS message that does not include R-ORG
0x31	0x8	N.A.	SEC_R	Secure message that includes the R-ORG of the original message (R-ORG and DATA are encrypted together)
0x32	0xF	0x04	SEC_D	Decrypted RPS message that does not include R-ORG
0x34	0xF	0x34	SEC_MAN	Secure Remote Management Command (Secure representation of R-ORG 0xC5: SYS_EX)
0x35	0x9	N.A.	SEC_TI	Security Teach-in
0x36	0xF	0x36	SEC_DAT	Secure Binary Data Transport (Secure representation of R-ORG 0xD6: DAT)
Transport R-ORG				
0xA6	N.A.	N.A.	ADT	Addressed Message (For ERP1 only, in ERP2, this is specified via HEADER[7:4])
0x40	0xF	0x03	CDM	Chained Data
0x33	0xF	0x33	SEC_CDM	Secure Chained Data
Generic Profile R-ORG				
0xB0	0xF	0x05	GP_LRN_REQ	Generic Profile Teach-in Request
0xB1	0xF	0x06	GP_LRN_RESP	Generic Profile Teach-in Response
0xB2	0xF	0x07	GP_CD	Generic Profile Complete Data
SmartACK R-ORG				
0xC6	0xF	0x01	SMACK_LRN_REQ	Smart ACK Learn Request
0xC7	0xF	0x02	SMACK_LRN_RESP	Smart ACK Learn Response

Table 2. Common R-ORG types and their encoding in the ERP2 HEADER and EXT_HEADER fields

2.5.2 Security Processing for ERP2 Messages

Security processing for ERP2 messages uses the actual R-ORG value (and not the values of the HEADER and EXT_TYPE fields) for security processing in the same way as ERP1.

HEADER and EXT_TYPE are neither modified by security processing nor do they affect the result of the security processing as ERP radio transmission occurs in a layer below the security processing.

This approach ensures that the same security mechanisms can be used for ERP1 and ERP2.

2.5.3 Secure Message Types

ERP data may be provided using one of the following secure message types:

- Secure EEP message with R-ORG and Data
  The most common secure message type contains both the EEP R-ORG and the corresponding EEP data payload. This type of secure message is identified by R-ORG-S 0x31: SEC_R.
  If the size of the message exceeds the maximum size of one radio telegram, then message segmentation (SEC_CDM) may be used.

- Secure EEP message without R-ORG
  Information about the R-ORG may be omitted for size or energy constrained applications where the receiver knows the EEP used by the transmitter. This type of secure message is identified by R-ORG-S 0x30: SEC and it is mainly used by switch modules.
  If such message is received and correctly decrypted and authenticated, then it will be provided using R-ORG 0x32: SEC_D to inform the application that this is an EEP data message which does not include the R-ORG.

- Secure Remote Management message
  Secure remote management messages are identified by R-ORG-S 0x34: SEC_MAN.
  If such message is received and correctly processed, then it will be provided to the application using R-ORG 0xC5: SYS_EX to inform the application that this is a remote management message.

- Secure Binary Data Transport message
  Secure binary data transport messages are identified by R-ORG-S 0x36: SEC_DAT.
  If such message is received and correctly processed, then it will be provided to the application using R-ORG 0xD6: DAT to inform the application that the content of the message is binary data.

- Security Teach-in message
  Security Teach-in messages are identified by R-ORG-S 0x35: SEC_TI and contain the required information (Security Profile, Security Key, RLC) for establishing secure communication.
  If such message is received, then the corresponding security information may be automatically added by the receiver to its Security Link Table used for the security processing of received messages.

3 Security Use Cases

ERP is used in a variety of applications with products designed for building automation being the most common use case. Two examples can be used to illustrate the need for a secure protocol and to derive the required security functionality.

3.1 Occupancy Reporting

Occupancy sensors can be used to automatically switch off the lighting or reduce the heating if a room is not occupied. Monitoring the reports from the occupancy sensors of a private home would allow potential intruders to determine if the home is unoccupied.

This can be avoided by Encryption of the data message content so that it remains unknown for unauthorized receivers.

3.2 Access Control

Wireless access control systems allow flexible installation, quick adaption, and easy control of access systems. It is however essential to prevent unauthorized users from controlling the access system so that an unauthorized user cannot unlock the door.

This can be avoided by Authentication and Integrity Verification of the data message content to ensure that the message was sent by an authorized transmitter and its content has not been modified. In addition to that, it must be ensured by Replay Protection that a previously used – and valid – data message cannot be retransmitted later.

3.3 Required Security Functionality

Based on the discussion in the previous sections, we can identify the required security functionality for ERP systems:

- Confidentiality
  It should not be possible for an unauthorized receiver to determine the content of the data that is exchanged between ERP devices

- Authenticity
  It should not be possible for an unauthorized transmitter to transmit data messages that will be accepted by a receiver.

- Integrity
  It should not be possible to modify the content of a transmitted message

- Originality (Replay protection)
  It should not be possible for an unauthorized transmitter to capture and re-transmit (reuse) a previously transmitted data message

ERP meets these requirements using the mechanisms described in subsequent chapters.

4 Security Architecture

This chapter explains key functionality provided by ERP security processing that is used to meet the security requirements identified in the previous chapter.

4.1 Security Modes

ERP systems may use the following Security Modes:

- VAES Encryption with CMAC Authentication
  This mode may be used where backwards compatibility is required

- CCM Encryption and Authentication
  This mode is recommended for new designs

Details of these Security Modes are described in Chapter 7.

4.2 Security Key (K)

Transmitter and receiver authorize each other by their mutual possession of a shared secret – the Security Key (*K*). For ERP systems, *K* has a size of 128-bits (16-bytes).

If transmitter and receiver both possess the same Security Key *K*, then the receiver will accept messages from the transmitter.

4.2.1 Security Key Assignment

At least one Security Key shall be assigned to an ERP device based on the following guidelines:

1. Each device shall be pre-configured with a unique, randomly generated Security Key

2. The value of the pre-configured Security Key should be provided by QR code of the device label as defined by the Product ID and Standardized Labeling specification by EnOcean Alliance

4.2.2 Security Key Update and Derivation

ERP devices shall be able to modify their Security Key based on the following guidelines:

1. Devices shall have the possibility to change their Security Key unless technical limitations such as unidirectional devices with no external interfaces and no user interface prevent this

2. Devices should be able to derive a temporary key (Session Key) from a pre-configured Security Key if required for the intended use case

4.2.3 Security Key Usage in Bi-directional or Multi-party Communication

For the case of bi-directional communication between two or more ERP devices, the following rule for key selection shall apply depending on the Security Mode that is used for communication:

- VAES Encryption with CMAC Authentication
  Different keys shall be used by each party

- CCM Encryption and Authentication
  The same key may be used by more than one party provided that each party uses a different combination of Sender ID (Device EURID or Base ID) and Destination ID (Broadcast ID, Device EURID or Base ID) for its transmission of radio messages

4.2.4 Security Key Update

The Security Key of a device may be changed in various ways, for instance via button press, via the ERP radio interface, via a HW interface (button press, etc.) or via NFC.

The Security Key of a local device may be updated by a remote device via ERP radio protocol in the same way as other device configuration parameters using Secure Remote Commissioning functionality.

4.2.5 Security Key Derivation

The Security Key *K* – acting as shared secret that is known both to the local and the remote device – provides the root of trust for secure communication between these devices.

For uses cases requiring maximum protection of the Security Key *K* against cryptographic analysis based on capture and subsequent analysis exchanged messages (also called “harvest-now-decrypt-later” attacks), it is desirable to limit the use of *K* to communication setup without using it for subsequent communication.

This goal can be achieved by using one or several temporary security keys – so-called Session Keys - which are derived from the primary Security Key *K*. Session Keys may be used for a limited time or for a specific purpose and then be discarded. Session Keys may be derived from the Security Key *K* by using a suitable key derivation function (KDF).

Chapter 7.6 describes the recommended key derivation mechanism that should be used by ERP devices.

4.3 Rolling Code (RLC)

One important requirement for security processing is the need for replay protection meaning that a previously transmitted data message cannot be re-transmitted (replayed) by an attacker. For instance, if the state of a door lock is controlled via ERP messages, then the option to capture (receive) and re-transmit a previously transmitted “door unlock” command message must be prevented.

Techniques used for preventing the re-transmission of previously transmitted messages are called Replay Protection. These techniques usually work by adding additional, unique data to the data message payload to ensure that messages contain different data even when the same command (for instance: door open) is transmitted.

Different mechanisms may be used to generate such additional data depending on the specific use case. These mechanisms must guarantee that each instance of additional data is used only once, meaning that it has not been previously used and will not be reused at another time during the lifetime of the device.

The most common mechanism for the generation of additional data is the use of rolling codes (RLC) due to the simplicity of data generation, data verification and assurance of uniqueness.

RLC are generated by a monotonously incrementing counter of sufficient size (with 4-byte size being a common choice) to guarantee that the same RLC will not re-occur during the device lifetime. RLC will usually be initialized to 0 during production or whenever a new Security Key is used. After that, RLC will increment after each data message that has been transmitted.

The transmitting device will increment by 1 the RLC value that was used for the transmission of a radio message by one and store this value in memory for the transmission of the next message.

Likewise, the receiving device will increment by 1 the RLC value that was used by a message that was successfully authenticated, store this value in memory and subsequently only accept a received data message from the same device if it uses this or a higher RLC value.

4.3.1 RLC Size

The size of the RLC counter may either be 32-bit or 24-bit. New designs shall use 32-bit RLC counter size unless use of 24-bit is required for backwards compatibility. The option of using 16-bit RLC has been deprecated and shall not be used in new designs.

4.3.2 Transmitted RLC Size

In some cases, it may not be necessary to transmit the entire RLC as part of each secure data message as the receiver may be able to keep track of the RLC value using the mechanism defined in Chapter 6.3.3.

Using reduced RLC size for transmitted messages reduces the overhead required for the transmission of the RLC value compared to a scenario where the entire RLC is transmitted. Reduced size RLC transmission is supported in conjunction with CCM as security algorithm.

Implicit RLC (where the RLC is only transmitted in Security Teach-in messages and omitted from secure data messages) shall not be used in new designs.

4.4 Encryption / Decryption

The data payload (EEP data) of ERP messages is encrypted by the sender and decrypted by the receiver using the Security Key to ensure its confidentiality during transmission.

Two algorithms may be used for that purpose:

- VAES
  The Variable AES (VAES) algorithm is used by legacy implementations. The VAES algorithm is described in Chapter 7.1.

- CCM
  The Counter with CBC MAC (CCM) algorithm provides both encryption / decryption and authentication signature generation functionality in one step. CCM is the recommended algorithm for new designs. The CCM algorithm is described in Chapter 7.3 ff.

4.5 Message Authentication Code (MAC)

Authenticity and integrity of a received data message are guaranteed by an authentication signature that is calculated based on message data, RLC and Security Key. In ERP, the authentication signature is called Message Authentication Code or MAC in short.

The sender calculates the MAC based on telegram data, Rolling Code (RLC) and Security Key and appends it to the transmitted message data.

Upon receiving a message and decrypting its content, the receiver will calculate the expected MAC based on message data, Rolling Code (RLC) and Security Key.

If the calculated (expected) MAC matches the transmitted MAC, then the following is true:

- Authenticity
  The data message was transmitted by a transmitter possessing the same key as the receiver

- Integrity
  The data message content has not been modified

- Uniqueness (Replay protection)
  The data message MAC was calculated based on the RLC value that is part of the message

4.5.1 MAC Calculation for VAES

If VAES is used as security algorithm, then MAC is calculated using the CMAC algorithm [7] according to the definition in Chapter 7.2. The transmitted MAC length should be 4-byte; use of 3-byte MAC length is only permitted where required for backwards compatibility or due to energy constraints.

4.5.2 MAC Calculation for CCM

If CCM is used as security algorithm, then MAC is calculated as part of the CCM algorithm as described in Chapter 7.4 and no separate processing step is necessary. The MAC length parameter shall be set to 4-byte within the CCM algorithm, and all 4-byte shall be transmitted.

4.6 Security Level Format

The combination of the selected security parameters defines the Security Level Format (SLF) used for communication between two devices. The SLF is provided by the transmitter to the receiver as part of the Security Teach-in message as described in Chapter 5.3.

If the security mode is VAES with CMAC and bi-directional communication is used, then both devices shall use the same SLF. If the security mode is CCM, then different SLF may be used by the two devices.

The SLF is expressed as a 1-byte value consisting of the following fields:

- SECURITY MODE
  Type of the encryption and authentication algorithms used as described in Chapter 4.1

- MAC SIZE
  Size of the transmitted MAC field as described in Chapter 4.5

- RLC SIZE
  Size of the transmitted RLC field as described in Chapter 4.3

The structure of the 1-byte SLF field is shown in Figure 9 below.

Security Level Format							
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
RLC SIZE			MAC SIZE		RFU	SECURITY MODE	

Figure 9. Structure of Security Level Format (SLF) Register

Permissible and recommended settings for these fields are defined below.

4.6.1 SECURITY_MODE

ERP implementations may support one or both of the following security modes:

- VAES with CMAC (Legacy Mode)
  This mode combines VAES encryption described in Chapter 7.1 with CMAC (RFC 4493) calculation described in Chapter 7.2.
  The SECURITY MODE field shall be set to 0b11 if VAES with CMAC mode is used.

- CCM (New Mode)
  This mode uses the standard CCM (RFC 3610) algorithm described in Chapter 7.3 which provides encryption and authentication within one algorithm.
  The SECURITY MODE field shall be set to 0b01 if CCM mode is used.

4.6.2 MAC SIZE

ERP implementations may use the following MAC sizes:

- 32-bit MAC
  This is the recommended option; the field MAC SIZE shall be set to 0b10 if this option is used.

- 24-bit MAC (only for VAES with CMAC mode)
  This option may only be used if required for compatibility with legacy implementations; it is not supported for CCM security mode. The field MAC SIZE shall be set to 0b01 if this option is used.

4.6.3 RLC SIZE

ERP implementations may use the following RLC sizes:

- 32-bit internal RLC size, 8 LSb transmitted
  This option is only available for CCM security mode; the field RLC SIZE shall be set to 0b000 if this option is used.

- 32-bit internal RLC size, 16 LSb transmitted
  This option is only available for CCM security mode; the field RLC SIZE shall be set to 0b001 if this option is used.

- 24-bit internal RLC size, RLC is not transmitted (implicit RLC, VAES with CMAC mode only)
  This option is only available for VAES with CMAC mode and shall only be used if required for backwards compatibility. The field RLC SIZE shall be set to 0b100 if this option is used.

- 24-bit internal RLC size, 24-bit transmitted (VAES with CMAC mode only)
  This option is only available for VAES with CMAC mode and shall only be used if required for backwards compatibility. The field RLC SIZE shall be set to 0b101 if this option is used.

- 32-bit internal RLC size, 24 LSb transmitted
  This option is available both for CCM and VAES with CMAC mode; the field RLC SIZE shall be set to 0b110 if this option is used.

- 32-bit internal RLC size, 32-bit transmitted
  This option is available both for CCM and VAES with CMAC mode; the field RLC SIZE shall be set to 0b111 if this option is used.

4.6.4 Summary of Security Level Formats and Corresponding SLF Values

Table 3 below summarizes permitted combinations of security parameters (so-called Security Level Formats) and the corresponding SLF values for use in new designs. Annex C provides a summary of legacy security parameters that may be present in older designs.

Security Mode	RLC size (internal)	RLC size (transmitted)	MAC size	SLF Value
VAES with CMAC	32-bit	32-bit	32-bit	0xF3
VAES with CMAC	32-bit	24-bit	32-bit	0xD3
VAES with CMAC	32-bit	24-bit	24-bit	0xCB
VAES with CMAC	24-bit	24-bit	24-bit	0xAB
VAES with CMAC	24-bit	None (Implicit RLC)	24-bit	0x8B
CCM	32-bit	32-bit	32-bit	0xF1
CCM	32-bit	24-bit	32-bit	0xD1
CCM	32-bit	16-bit	32-bit	0x31
CCM	32-bit	8-bit	32-bit	0x11

Table 3. Summary of Security Level Formats and corresponding SLF values

4.7 Guidance on Usage of Security Parameters

Security implementations in ERP devices should select the recommended combination of security parameters (security formats) specified below.

4.7.1 Standard Format

Standard security level format should be used for all line-powered ERP devices and for other ERP devices with sufficient energy budget (including battery-powered devices or energy-harvesting devices with sufficient energy supply) except where backwards compatibility, payload or energy constraints demand otherwise.

- RLC SIZE: 32-bit internal RLC
  32-bit transmitted RLC
- MAC SIZE: 32-bit MAC
- SECURITY_MODE: CCM (recommended for new designs)
  VAES with CMAC (for backwards compatibility)
- SLF: 0xF1 (CCM)
  0xF3 (VAES with CMAC)

4.7.2 Standard Format for CCM with Reduced RLC Size

CCM with reduced RLC size may be used for cases where available energy is limited or the amount of available payload needs to be maximized. This mode uses the following parameters:

- RLC SIZE: 32-bit internal RLC
  8-bit transmitted RLC (Reduced Size RLC)
- MAC SIZE: 32-bit MAC
- SECURITY_MODE: CCM
- SLF: 0x11

4.7.3 Format for Energy-constrained Devices using VAES with CMAC

The format for energy-constrained devices may be used if necessary for legacy devices such as solar-powered sensors or kinetic-harvesting switches using VAES with CMAC security mode.

Note that two different RLC types are supported as certain legacy receivers support only the option of 24-bit internal RLC. These modes use the following parameters:

- RLC SIZE: 32-bit internal RLC and 24-bit transmitted RLC (preferred)
  24-bit internal RLC and 24-bit transmitted RLC (for legacy support)
- MAC SIZE: 24-bit MAC
- SECURITY_MODE: VAES with CMAC
- SLF: 0xCB (if 32-bit internal RLC is used)
  0xAB (if 24-bit internal RLC is used)

4.7.4 Format for Ultra-low Power Devices using VAES with CMAC

The ultra-low power security level format (SLF) shall only be used for ERP devices without internal energy storage (such as kinetic-harvesting switches) that are unable to use the format for energy-constrained devices.

This format is not recommended for new designs and uses the following parameters:

- RLC SIZE: 24-bit internal RLC
  RLC is not transmitted (Implicit RLC)
- MAC SIZE: 24-bit MAC
- SECURITY_MODE: VAES with CMAC
- SLF: 0x8B

5 Security Setup

Security setup (or Security Teach-in) is the procedure of configuring the required parameters for secure communication between transmitter and receiver.

Within this Security Teach-in procedure, transmitter and receiver will agree on the following parameters:

- Device ID (EURID or Base ID) that will used to identify themselves
- Security Level Format (SLF) defining the security parameters
- Security Key that will be used for encryption and authentication of messages
- Next RLC value that will be used for successful message authentication

This chapter explains two options for Security Teach-in.

5.1 Security Setup using Out-of-band Mechanisms

Device ID, Security Key and security mode used by one device can be provided to another device using out-of-band mechanisms such as QR code, NFC, Remote Management or a non-ERP communication interface such as IP or serial communication.

Using such out of band approach avoids the need to transmit the Security Key over the air using a secure teach-in message as described in Chapter 5.3.

If an out-of-band approach is used, then the receiver may initialize the RLC for the transmitter based on the first received secure message that is successfully authenticated and decrypted. This approach requires that the RLC is transmitted as part of the secure message; it is not possible for implicit RLC mode as described in Chapter 4.3.1.

5.2 Security Setup using QR Code

EnOcean devices shall include a QR code as defined by the EnOcean Alliance Product ID and Standardized Labeling Specification [8]. Providing such QR code enables automated device setup including security teach-in while minimizing risks and errors related to manual setup.

The QR Code shall at a minimum contain the fields specified in Table 4 below.

Data Identifier	Data Size (excl. Identifier)	Data Content
30S	12 characters	EURID of the device
1P	12 characters	EnOcean Alliance Product ID of the Device
13Z	32 characters	Security Key of the device

Table 4. Mandatory QR Code Content for EnOcean Alliance Products

To maximize compatibility, the QR code content is expressed as text string, for instance:

"30S000012345678+1P0x000B0000004C+13Z00112233445566778899AABBCCDDEEFF"

This text string example would identify an STM 550 device (Rev. DB) from EnOcean GmbH using EURID 0x000012345678 and security key 0x00112233445566778899AABBCCDDEEFF.

5.3 Security Setup using Security Teach-in Messages

For cases where out-of-band security setup is not possible, it is possible to use a procedure where the security teach-in information is transmitted within a dedicated ERP message. This message is called a Security Teach-in message; it is identified by R-ORG-S 0x35: SEC_TI and uses the format described below.

Note that the Security Teach-in process only provides the required data for the setup of secure communication between two devices. Information about the EEP that is used for communication must be transmitted separately after completion of the Security Teach-in procedure (if required).

5.3.1 Format of Security Teach-in Messages

Security Teach-in messages use the structure shown in Figure 10 below.

Security Teach-in Telegram Structure				
R-ORG-S 0x35: SEC_TI	TEACH-IN INFO (TII)	SECURITY FORMAT (SLF)	ROLLING CODE (RLC)	SECURITY KEY
1-byte	1-byte	1-byte	3/4-byte	16-byte

Figure 10. Security Teach-in message structure

As shown above, the Security Teach-in message contains the following parameters:

- R-ORG 0x35 (SEC_TI)
  Secure teach-in messages are identified by the R-ORG 0x35 (SEC_TI)

- Teach-in Info
  This field contains information about the Security Teach-in messages as described in chapter 8.3

- SLF
  The SLF specifies the type of encryption and authentication used by the device for communication with the remote device as described in 4.6

- Rolling Code (RLC) value
  This field contains the RLC value that will be used by the remote device for the transmission of the next message as described in chapter 4.3.

- Security Key
  The 16-byte security key is used by the sender to encrypt and authenticate transmitted messages and by the receiver to decrypt and authenticate received messages as described in chapter 4.2.

5.3.2 Security Teach-in Message with PSK

Security Key and RLC may be protected during the transmission of the Security Teach-in message using encryption with a Pre-shared Security Key (PSK) which must be known to the receiver of the Security Teach-in message.

Depending on the selected security mode, the content of the Security Teach-in message may either be protected using VAES or CCM security mode as described in the following chapters.

5.3.2.1 Security Teach-in Message with PSK in VAES mode

If VAES mode is selected and Secure Teach-in with PSK is used, then the concatenation of RLC and Security Key shall be encrypted using the VAES encryption process described in Chapter 7.1 using the following parameters:

- DATA = RLC (3 byte or 4 byte) || Security Key (16 byte)
- RLC used for Encryption = 0x00:00:00:00
- SECURITY KEY = PSK

Note that in VAES mode with PSK, the content of the Security Teach-in message (RLC and Security Key) is encrypted, but is not authenticated. Using this mode is therefore only recommended for specific scenarios where authenticity and integrity of the received Security Teach-in message can be assured by other means.

Figure 11 below shows the content of a Security Teach-in message when PSK is used in VAES mode.

Security Teach-in Message with PSK in VAES Mode				
R-ORG-S 0x35: SEC_TI	TEACH-IN INFO (TII)	SECURITY FORMAT (SLF)	ROLLING CODE (RLC)	SECURITY KEY
1-byte	1-byte	1-byte	3/4-byte	16-byte
Plaintext			Encrypted with PSK	

Figure 11. Security Teach-in message format in VAES mode with PSK

5.3.2.2 Security Teach-in Message with PSK in CCM mode

If CCM mode is selected and Secure Teach-in with PSK is used, then the Security Key shall be encrypted using the CCM encryption process described in Chapter 7.5 and authenticated using the CCM authentication process described in Chapter 7.4.

The following parameters shall be used:

- Nonce *N* = SENDER ID || DESTINATION ID || RLC || 0x00
- Authenticated Data *a* = SENDER ID || DESTINATION ID || RLC || R-ORG-S || TEACH_IN_PARAMETERS || SLF
- Encrypted and Authenticated Data *e* = Security Key
- Encryption Key *K* = PSK

Note that:

- Security Key is both encrypted and authenticated
- SENDER ID, DESTINATION ID, RLC, R-ORG-S, TEACH_IN_PARAMETERS and SLF are authenticated, but not encrypted
- For the purpose of authentication, TEACH_IN_PARAMETERS shall be used which contains the least significant 4 bit of Teach-In Info (TII). The upper 4-bit (which are used by Teach-in Info to control the chaining of the Security Teach-in telegram) shall be set to 0x0 for the purpose of authentication.

Figure 12 below shows the payload of a Security Teach-in message with PSK in CCM security mode.

Security Teach-in Message with PSK in CCM Mode					
R-ORG-S	TII	SLF	RLC	SECURITY KEY	MAC
1-byte	1-byte	1-byte	3 / 4-byte	16-byte	4-byte
Authenticated				Authenticated and Encrypted	Authentication Signature

Figure 12. Security Teach-in message content for CCM mode when using PSK

5.3.3 Security Setup Process

Security setup using Security Teach-in messages can be executed in two ways:

- Uni-directional security setup
  Device_A communicates its security information to Device_B

- Bi-directional security setup
  Device_A communicates its security information to Device_B
  Device_B responds by providing its own security information to Device_A

These different setup procedures are described below.

5.3.3.1 Uni-directional Security Setup

To describe the process of uni-directional security setup via Security Teach-in message, we consider Device_A and Device_B shown in Figure 13.

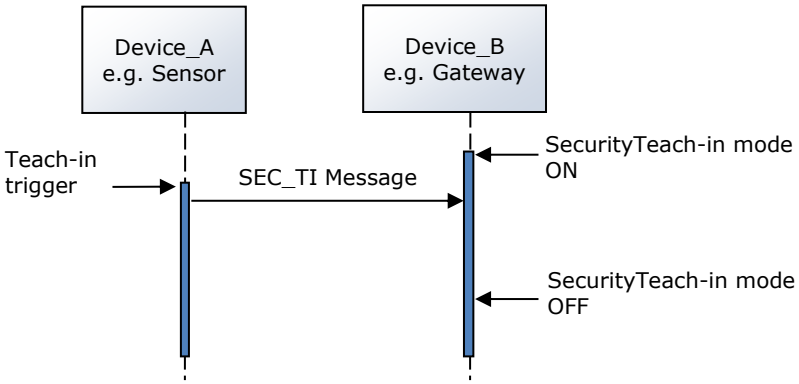


Figure 13. Schematic representation of a uni-directional security setup

To execute the Security Teach-in procedure using a SEC_TI message, Device_B must be set into Security Teach-in Mode (for instance via button press) to accept Security Teach-in messages from other devices. Learn mode is usually time-limited (for instance, to 30 seconds) and Security Teach-in messages are only accepted while learn mode is active. This limits the likelihood of accepting unintended Security Teach-in messages.

Once Device_B is set into learn mode, Device_A must be triggered (for instance via button press) to send a Security Teach-in (SEC_TI) message.

If Device_B received the Security Teach-in message from Device_A while being in Security Teach-in Mode, then it stores the EURID and the security parameters (Security Key, SLF and current RLC) of Device_A that are transmitted in the Security Teach-in message.

5.3.3.2 Bi-directional Security Setup

To describe the process of bi-directional security setup via Security Teach-in message, we again consider Device_A and Device_B shown in Figure 14.

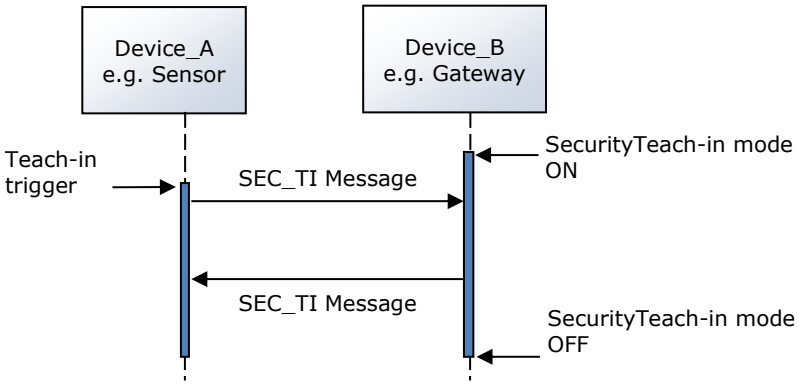


Figure 14. Schematic representation of a bi-directional security setup

The first part of the teaching-in procedure is executed in the same way as for the case of uni-directional security setup.

After Device_B received the Security Teach-in message from Device_A while being in Security Teach-in Mode, Device_B will then respond with its own Security Teach-in message which will be sent as addressed message to the EURID of Device_A.

The Security Teach-in message of Device_B shall be transmitted no later than 500 milliseconds after the reception of the Security Teach-in message from Device_A.

Device_A shall accept Security Teach-in messages no later than 750 milliseconds after the transmission of its own Security Teach-in message within the scope of this procedure.

6 Security Processing

ERP security processes the data content of ERP messages without affecting lower layer specifics such as STATUS, CRC / HASH or physical radio properties. Security processing involves the following steps:

- Encryption / decryption
- Authentication
- RLC management
- Message processing

These steps are described in the following chapters.

6.1 Encryption and Decryption

ERP security processing shall apply encryption by the sender and decryption by the receiver as follows:

- R-ORG
  If the R-ORG is part of the original (unencrypted) message, then it shall be encrypted and as part of the security processing and a dedicated R-ORG-S 0x31: SEC_R (secure message with encrypted R-ORG) shall be used for the resulting secure message
  If the R-ORG is specific either to security processing or the transport of secure messages (e.g. SEC_TI, SEC_DAT or SEC_CDM and ADT), then it shall not affect the security processing

- DATA
  The DATA field of ERP messages shall be encrypted

- SENDER ID, DESTINATION ID, STATUS, CRC / HASH, HEADER, EXT_HEADER, EXT_TYPE
  These fields are required for ERP radio functionality and are therefore not encrypted

Encryption and decryption may either use the VAES algorithm described in Chapter 7.1 or the encryption step of the CCM algorithm described in Chapter 7.5.

6.2 Authentication

ERP security processing shall authenticate the different fields of ERP messages by calculating the message authentication code (MAC) as follows:

- R-ORG-S and R-ORG (if present)
  - R-ORG-S and R-ORG (if present) shall be authenticated
- DATA
  - DATA shall be authenticated
- RLC
  - RLC shall be authenticated
- SENDER ID, DESTINATION ID
  - For CCM mode, SENDER ID and DESTINATION ID shall be authenticated
    DESTINATION ID shall be set to 0xFF:FF:FF:FF for the case of Broadcast messages
  - For VAES mode, SENDER ID and DESTINATION ID shall not be authenticated
- STATUS, CRC / HASH, HEADER, EXT_HEADER, EXT_TYPE
  - These fields are required for ERP radio functionality, and their content may change if repeaters are used; they are therefore not authenticated

The message authentication code (MAC) may either be calculated using the CMAC algorithm described in chapter 7.2 or using the authentication step of the CCM algorithm described in chapter 7.4.

6.3 RLC Management

Devices need to independently manage two different Rolling Code (RLC) values:

- Local RLC (RLC_L)
  RLC_L is the RLC value used for security processing during message transmission to a remote receiver. RLC_L contains the value that will be used for the transmission of the next message to the remote receiver.

- Remote RLC (RLC_R)
  RLC_R is the RLC value used for security processing during message reception from a remote sender. RLC_R contains the lowest RLC value that will be accepted for the reception of a message from the remote sender.

The procedures for managing the RLC values are defined below.

6.3.1 Transmission

The transmitter shall initialize to zero the internal Rolling Code value RLC_L that is used for transmitting the next message to a remote receiver upon security setup with a new Security Key.

The transmitter shall increment the value of RLC_L by 1 after each message transmission to the remote receiver. The combination of a specific Security Key and a specific RLC_L value shall only be used throughout the lifetime of the device.

6.3.2 Reception

The receiver shall initialize to zero the internal Rolling Code value RLC_R that holds the lowest acceptable RLC value for receiving messages from a specific sender for the case of out-of-band device setup. The same Security Key shall not be used more than once for such procedure.

For the case of device setup using a Security Teach-in message, the receiver shall initialize the RLC_R value to the value provided in the Security Teach-in message.

If another Security Teach-in message is received from the same device and the Security Key provided by this Security Teach-in message matches the stored Security Key for that device, then the receiver shall update internal rolling code value to the value provided by that Security Teach-in message.

The receiver shall only accept messages where the rolling code value is equal to or higher than the stored RLC_R value. After successfully authenticating a received message, the receiver shall update RLC_R to the RLC value that was used by this message, incremented by one.

6.3.3 RLC Tracking

As discussed above, using an RLC for replay protection requires that the receiver can determine the RLC that is used by the transmitted message so that only messages with an RLC value that had not been used previously will be accepted.

To determine the RLC value used by the transmitted messages, the simplest approach is to transmit the entire RLC value as part of every data message ("Full RLC Transmission"). This allows the receiver to directly determine the current RLC and process the message accordingly.

For energy-constrained devices or for use cases requiring larger data payload in a message, it may be desirable to only transmit a part of the RLC value ("Reduced RLC Size Transmission"). For such cases, the receiver needs to keep track of the RLC part that is not transmitted using a process called "RLC Tracking". This process relies on knowing the full initial RLC value which is subsequently updated with each successfully authenticated message.

Legacy implementation may not transmit the RLC at all as part of data messages and rely instead on the receiver to determine the RLC value based on the initial RLC value communicated during the security setup and the number of successfully decrypted and authenticated messages. This operation mode is called "Implicit RLC" and is no longer permitted for new designs.

6.3.3.1 Obtaining the Initial RLC Value

The receiver may obtain the initial RLC value for instance by the following mechanisms:

- Security Teach-in message
  The RLC value that is transmitted as part of the Security Teach-in message shall define the next expected (lowest acceptable) RLC value

- NFC
  If the transmitter device provides an NFC interface, then it may be possible to obtain the next expected (lowest acceptable) RLC value using this interface

The receiver shall store this initial RLC value and use it for RLC tracking.

6.3.3.2 RLC Tracking for Reduced RLC Size Transmission

If the transmitted RLC is less than 32-bit, then the receiving device shall determine the missing part of the RLC value based on the content of the internal RLC counter which has been initialized according to one of the mechanisms described above.

If a new message is received, then the receiving device shall compare the RLC value that was received with the least significant part (of same size) of the internal RLC counter. If for instance the received RLC was 1 byte long, then the receiving device would compare the received RLC value with the least significant byte of the internal RLC counter.

If the received RLC value is bigger than or equal to the least significant bytes (same size as received RLC) of the internal RLC counter, then the receiving device shall append the received RLC value to the most significant bytes of the internal RLC counter. If for instance the transmitted RLC is 1-byte, then the receiving device shall append the received RLC value to the three most significant bytes of the internal RLC counter.

If the transmitted RLC value is less than the least significant bytes of the internal RLC counter, then the receiving device shall append the received RLC value to the most significant bytes of the internal RLC counter incremented by 1. If for instance the transmitted RLC is 1 byte, then the receiving device shall append the transmitted RLC value to the three most significant bytes of the internal RLC counter after incrementing those by 1.

The receiving device shall then attempt to decrypt and authenticate the received message using the so-derived RLC value. If decryption and authentication are successful, then the receiving device shall update the stored RLC value with the so-derived RLC value, incremented by one.

This procedure assures that the receiver can always determine the full RLC value used by the sender if it has not missed more than 255 telegrams for the case of 1-byte transmitted RLC or 65535 telegrams for the case of 2-byte transmitted RLC.

If the receiving device may be out of radio range from the sender for an extended period resulting in more telegrams being missed than supported by above procedure, then the receiving device may optionally try additional (incrementing) values for the most significant bytes of its internal RLC counter.

Example 1

For this example, we assume:

- RLC_R (Lowest acceptable RLC) stored at the receiving device is 0x01:02:03:F0
- Transmitting device uses 1-byte RLC transmission (reduced size)
- Receiving device receives a message using 0xF2 as 1-byte RLC

The receiving device shall compare the received RLC (0xF2) with the least significant byte of the stored RLC_R (0xF0) and would determine that the received RLC is bigger.

The receiving device shall therefore append the received RLC to the most significant 3 byte of the stored RLC_R (0x01:02:03) resulting in RLC = 0x01:02:03:F2 and attempt to decrypt and authenticate the received message using the RLC value.

If authentication is successful using RLC = 0x01:02:03:F2, then receiving device shall update its stored RLC_R to 0x01:02:03:F3 which would then become the lowest acceptable RLC value for subsequently received messages.

Example 2
For this example, we assume:

- RLC_R (lowest acceptable RLC) stored at the receiving device is: 0x01:02:03:F2
- Transmitting device uses 1-byte RLC transmission (reduced size)
- Receiving device receives a message using 0x01 as 1-byte RLC

The receiving device shall compare the received RLC (0x01) with the least significant byte of the stored RLC_R (0xF2) and would therefore determine that the received RLC is smaller.

The receiving device would therefore append the received RLC to the most significant 3 byte of the stored RLC_R incremented by 1 (0x01:02:04) resulting in RLC = 0x01:02:04:01 and attempt to decrypt and authenticate the received message using the RLC value.

If authentication is successful using RLC = 0x01:02:04:01, then the receiving device shall update its stored RLC_R to 0x01:02:04:02 which would then become the lowest acceptable RLC value for subsequently received messages.

6.3.3.3 RLC Tracking for Transmission with Implicit RLC

Use of implicit RLC (where no part of the RLC is transmitted) requires that the receiver continuously keeps track of the RLC used by the transmitter.

One key issue with that approach is that the receiver might miss one or several transmitted data messages due to temporary power loss or the transmitter being temporarily out of the radio range of the receiver. In that case, the transmitter will use an RLC that is different (higher) than the RLC value expected by the receiver.

To account for that possibility, it is required that the receiver “tries” a range of possible RLC values – within the so-called RLC Window – if security processing with the expected RLC value fails.

The size of the RLC Window must balance the possibility of extended power or communication loss with the processing overhead associated with trying different possible RLC values for each received message and the possibility of denial-of-service attacks as discussed below.

The most common choice for RLC Window size is 128, which means that the receiver will try both the expected RLC and up to 127 subsequent RLC values until the authentication signature matches the expected value.

Figure 15 below illustrates the use of an RLC Window.

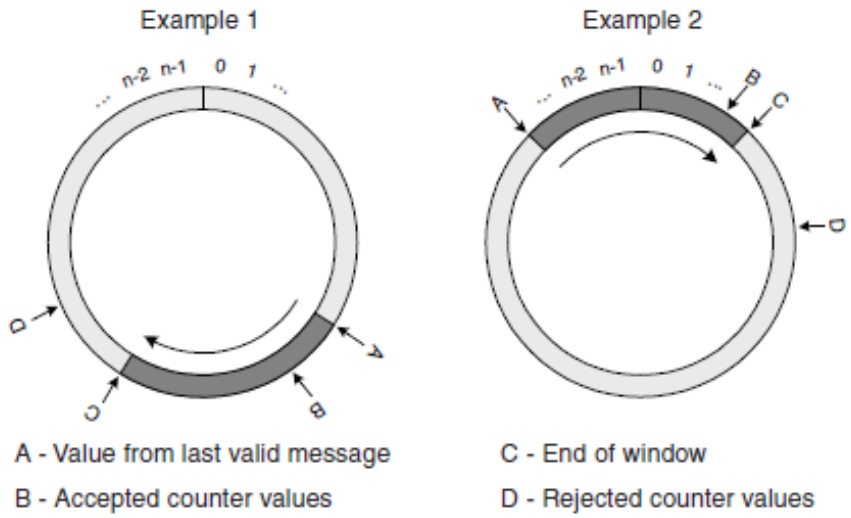


Figure 15. Window of Acceptance for RLC Counter Values

RLC Window is a mechanism that ensures that even if the transmitter and receiver temporarily lost their synchronization (transmitter was operated outside the range of the receiver or messages were lost), messages can still be correctly authenticated and decrypted provided that the difference between the rolling code counters in the transmitter and receiver was not bigger than the rolling code window size.
If the receiver finds the correct RLC within RLC Window (meaning that authentication and decryption of the data message were successful using that RLC), then the receiver will update the previously stored RLC with the new RLC value and thereby resynchronize to the RLC value used by the transmitter.

If the receiver does not find the correct RLC value within the RLC window, then authentication and decryption of the data message fail.

The receiver may then block this specific transmitter (identified by its EURID) so that no security processing is done for subsequently received data messages or enforce a backoff timing whereby security processing is suspended for an application-defined period. This avoids the case where the receiver spends significant processing trying to decrypt and authenticate messages that were sent with incorrect Security Key or incorrect RLC by an unauthorized transmitter.

Note: The implementation of such blocking mechanism could enable denial of service attacks by sending random security messages using the known Sender ID of the attacked transmitter to cause the receiver to temporarily or permanently block the transmitter. Implicit RLC shall therefore not be used in new designs.

If transmitter and receiver RLC were desynchronized with a RLC difference bigger than RLC Window size, then they need to be synchronized again either by means of the Security Teach-in procedure described in Chapter 5.3 or by directly configuring the receiver with the most recent RLC.

Use of implicit RLC is a deprecated feature that is not permitted for new designs.

6.4 Segmentation (Chaining) and Reassembly (De-chaining)

Secure message segmentation (chaining) is a transport layer mechanism where the sender segments the payload of a secure data message into two or more secure chained data message telegrams (SEC_CDM).

Secure message segmentation using SEC_CDM is defined for use with secure messages of type R-ORG-S 0x31: SEC_R, meaning for secure data messages containing an Encrypted R-ORG.

Secure message segmentation is required if the total length of the message payload – consisting of the fields listed below - exceeds the maximum payload size of 14-byte for ERP1 and most ERP2 telegrams:

- R-ORG ADT (1-byte, only present for addressed data telegrams)
- Encrypted R-ORG (1-byte)
- Encrypted DATA
- RLC (1-byte … 4-byte)
- CMAC (3-byte or 4-byte)
- DESTINATION ID (4-byte, only present for addressed data telegrams)

This applies as follows to common security parameter combinations for broadcast data messages:

- 3-byte RLC and 3-byte CMAC: Chaining is required if Encrypted DATA is larger than 7-byte
- 4-byte RLC and 4-byte CMAC: Chaining is required if Encrypted DATA is larger than 5-byte

The first chained data message telegram contains a 2-byte field (CHAIN_LENGTH) indicating the total payload length of the secure data telegram. This field is omitted in subsequent chained data message telegrams.

The number of chained data message telegrams is determined by the size of the payload and the type of transmission (broadcast or addressed data telegram).

The payload shall be distributed across the chained data message telegrams such that all telegrams - except the last one - have maximum size.

The receiver shall reassemble (de-chain) received chained data message telegrams and then provide the content (encrypted R-ORG, encrypted DATA, RLC and MAC) to security processing.

For the case of addressed messages (ADT), the receiver may use the provided DESTINATION ID to determine if the message should be processed.

Segmentation and reassembly of secure data messages is a transport layer mechanism that shall not affect the security processing (encryption or authentication) of data messages. Security processing shall be performed by the sender before segmentation and by the receiver after reassembly.

Appendix A.3 provides an example for the processing of a secure chained data message.

6.4.1 Message Format

Secure chained data messages are identified by R-ORG-S: 0x33 (SEC_CDM) which during transport replaces R-ORG-S 0x31 (SEC_R) of the secure data message.

6.4.1.1 CHAIN_CTRL

Each secure chained data message telegram shall include the 1-byte CHAIN_CTRL field with the content shown in Table 5 below.

Position	Content	Supported Values
Bit 6:7	CHAIN_ID	0b00: RFU (Do Not Use) 0b01: Chain 1 0b10: Chain 2 0b11: Chain 3
Bit 0:5	INDEX	0b000000: Segment #0 0b000001: Segment #1 … 0b111111: Segment #63

Table 5. Structure of the CHAIN_CTRL field

CHAIN_ID shall be the same for each telegram belonging to the same chain and shall identify telegrams that belong to the same chain. CHAIN_ID shall increase whenever a new message chain is transmitted and cannot be equal to zero. This means that CHAIN_ID can be either 0b01, 0b10 or 0b11 which allows distinguishing up to three different message chains.

INDEX shall indicate the order of the segments (telegrams) within a specific message chain. INDEX starts at zero for the first telegram and is incremented for each subsequent telegram in this chain.

6.4.1.2 First Telegram in a Secure Message Chain

The first telegram of a message chain (where INDEX = 0b000000) shall additionally contain the 2-byte CHAIN_LENGTH field which indicates the length (in byte) for the combination of the encrypted R-ORG, encrypted DATA, RLC and MAC fields of the secure ERP message.

The CHAIN_LENGTH field shall follow the CHAIN_CTRL field and shall use big-endian byte order.

The PAYLOAD of the first telegram of a broadcast secure message chain shall always be 11-byte. Figure 16 below illustrates the format of the first telegram of such secure message chain.

R-ORG-SEC_CDM	CHAIN_CTRL		CHAIN_LEN	PAYLOAD
	CHAIN_ID	INDEX		
1-byte	1-byte		2-byte	11-byte

Figure 16. Format of the first telegram in a Broadcast Secure Message Chain

The payload of the first telegram of an addressed (ADT) secure message chain shall always be 6-byte. Figure 17 below illustrates the format of the first telegram of such message chain.

R-ORG ADT	R-ORG-SEC_CDM	CHAIN_CTRL		CHAIN_LEN	PAYLOAD	DESTINATION ID
		CHAIN_ID	INDEX			
1-byte	1-byte	1-byte		2-byte	6-byte	4-byte

Figure 17. Format of the first telegram in an Addressed Secure Message Chain

6.4.1.3 Last Telegram in a Secure Message Chain

The last telegram of the secure message chain may contain up to 13-byte of payload (including the RLC and MAC fields) for broadcast data messages as shown in Figure 18.

R-ORG-S SEC_CDM	CHAIN_CTRL		PAYLOAD	RLC	MAC
	CHAIN_ID	INDEX			
1-byte	1-byte		Up to 13-byte		

Figure 18. Format of the last telegram in a Broadcast Secure Message Chain

The format for the last telegram of an addressed secure message chain is shown in Figure 19 below. It may transport up to 8-byte of payload (including the RLC and MAC fields).

R-ORG ADT	R-ORG-S SEC_CDM	CHAIN_CTRL		PAYLOAD	RLC	MAC	DESTINATION ID
		CHAIN_ID	INDEX				
1-byte	1-byte	1-byte		Up to 8-byte			4-byte

Figure 19. Format of the last telegram in an Addressed Secure Message Chain

6.4.1.4 Intermediate Telegrams in a Secure Message Chain

If the total payload exceeds 24-byte, then additional – intermediary - telegrams (after the first telegram, before the last telegram) may be added.

Figure 20 shows the format of an intermediate telegram of a broadcast secure message chain where each telegram transports 13-byte of payload.

R-ORG-S SEC_CDM	CHAIN_CTRL		PAYLOAD
	CHAIN_ID	INDEX	
1-byte	1-byte		13-byte

Figure 20. Format of an intermediate telegram in a Broadcast Secure Message Chain

Figure 21 shows the format of an intermediate telegram of an addressed secure message chain where each telegram transports 8-byte of payload.

R-ORG ADT	R-ORG-S SEC_CDM	CHAIN_CTRL		PAYLOAD	DESTINATION ID
		CHAIN_ID	INDEX		
1-byte	1-byte	1-byte		8-byte	4-byte

Figure 21. Format of an intermediate telegram in an Addressed Secure Message Chain

6.5 ERP1 versus ERP2 Security Processing

Security processing is identical for ERP1 and ERP2 messages using the following fields:

- R-ORG-S
- R-ORG (if present) and DATA
- RLC
- SENDER ID and DESTINATION ID (for CCM mode only)

For security processing of ERP2 data messages, the actual value of the R-ORG-S field (which is either 0x30: SEC or 0x31: SEC_R) is used in the authentication step. The encoding of R-ORG-S into the HEADER field of ERP2 telegrams occurs after security processing as part of the transport layer processing.

Likewise, transport layer fields such as SEC_CDM (for the segmentation of secure data messages) or ADT (ERP1) / HEADER (ERP2) which specify the addressing mode are not part of the security processing.

7 Security Algorithms

This section describes the security algorithms used by the ERP security implementation.

7.1 VAES Encryption and Decryption

The ERP security implementation uses a variable AES (VAES) implementation to encrypt the concatenation of the R-ORG and Data fields of the ERP message using the Security Key *K* in conjunction with the rolling code RLC.

7.1.1 Input data

The input data for the encryption process is as follows:

- Security Key
  128-bit Security Key *K*

- Rolling Code
  24-bit or 32-bit value of RLC

- Security Payload
  Concatenation of the 1-byte R-ORG field (if present) with the content of the Data field

Figure 22 below illustrates the security payload for the VAES encryption process.

R-ORG	DATA
1-byte (Not present in R-ORG-S: SEC)	Variable length

Figure 22. Security payload for the VAES encryption process

7.1.2 VAES encryption process

VAES encryption uses a sequential process of XOR and AES-128 operations for processing the input data together with the Security Key and RLC to produce the encrypted payload as shown in Figure 23 below.

If the size of the input payload is more than 16-bytes, then an intermediary result of the first processing phase is used as input to the second processing phase.

Annex A provides step-by-step processing examples.

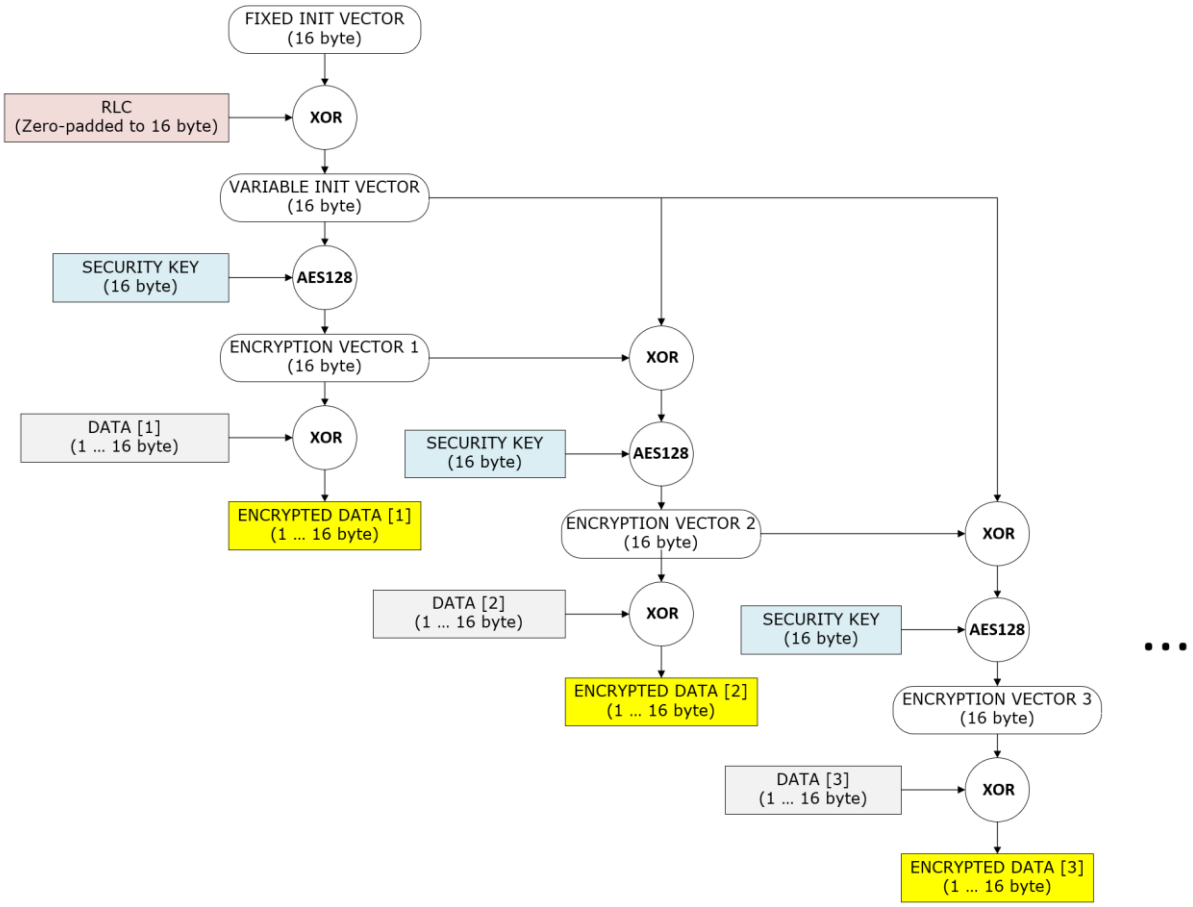


Figure 23. VAES encryption process steps

7.1.3 VAES decryption process

As illustrated in Figure 24 below, the VAES decryption process is the inversion of the VAES encryption process.

Again, if the size of the input data is more than 16-bytes, then an intermediary result of the first processing phase is used as input to the second processing phase.

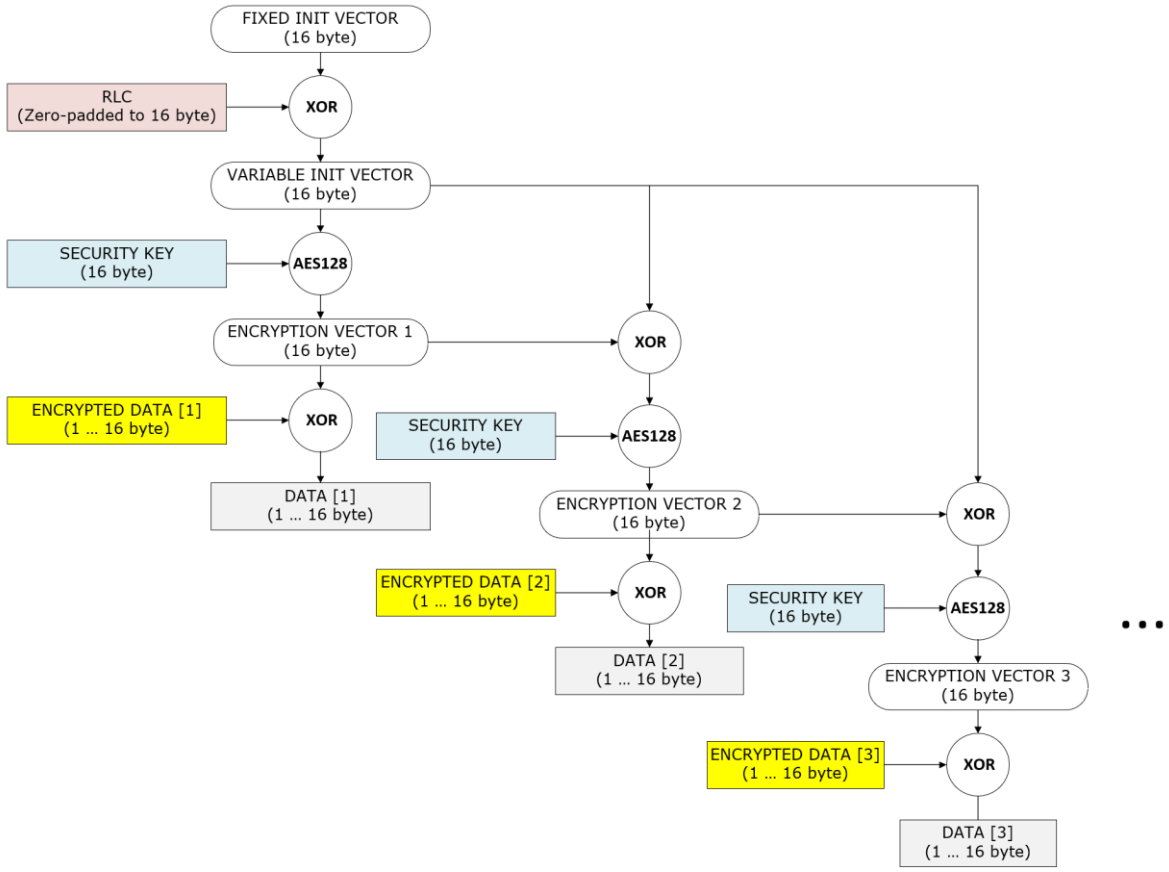


Figure 24. VAES decryption process steps

7.1.4 VAES INIT VECTOR

VAES INIT VECTOR is constant and used as input to the VAES algorithm. For security processing for ERP messages, the following value shall be used:

VAES INIT VECTOR = 0x3410DE8F1ABA3EFF9F5A117172EACABD

7.2 CMAC Authentication

The ERP security implementation uses the authentication mechanism defined by RFC 4493 [7] to generate a CMAC signature to authenticate input data of variable length. The following chapters illustrate this process for reference only.

The execution of the algorithm is slightly different depending on whether the length of the input data defined in Chapter 7.2.1 is a multiple of 16-byte (16-byte, 32-byte, …) or not.

7.2.1 Input Data

The input data for the CMAC algorithm consists of the concatenation of R-ORG-S, the encrypted content of the ERP message DATA field (which may include the encrypted R-ORG of the ERP message) and the RLC as shown in Figure 26 below.

Note that the actual R-ORG-S (0x30 or 0x31) is used for the processing of ERP2 messages, not the encoded representation of R-ORG-S provided by HEADER.

If R-ORG-S = 0x30: SEC, then the input to the CMAC function is the concatenation of R-ORG-S with the encrypted DATA and the currently used RLC value as shown in Figure 25 below.

R-ORG-S	Encrypted DATA	RLC
0x30: SEC	1-byte	3/4-byte

Figure 25. Input data for the CMAC algorithm for R-ORG-S 0x30: SEC

If R-ORG S = 0x31: SEC_R, then the input to the CMAC is the concatenation of R-ORG-S with the encrypted R-ORG, the encrypted DATA and the currently used RLC value as shown in Figure 26 below.

R-ORG-S	Encrypted R-ORG	Encrypted DATA	RLC
0x31: SEC_R	1-byte	Variable	3/4-byte

Figure 26. Input data for the CMAC algorithm for R-ORG-S 0x31: SEC_R

If the input data is more than 16-byte long, then it will be split into several parts INPUT DATA [1], INPUT DATA [2], INPUT DATA [3] and so on where all parts - except the last one - are 16-byte long. The last part might have any size between 1 and 16-byte.

The CMAC processing is slightly different depending on the size of the last part as described in the next two chapters.

7.2.2 CMAC calculation for multiples of 16 bytes

If the length of the input data is exactly a multiple of 16 bytes (16 bytes, 32 bytes, …), then the last part will have a size of 16-byte as well. The CMAC authentication signature is then calculated according to the process shown in Figure 27 below using the Security Key and another key K1 which is derived from the Security Key according to the procedure defined in chapter 7.2.4.

Annex A provides step-by-step examples for the CMAC calculation.

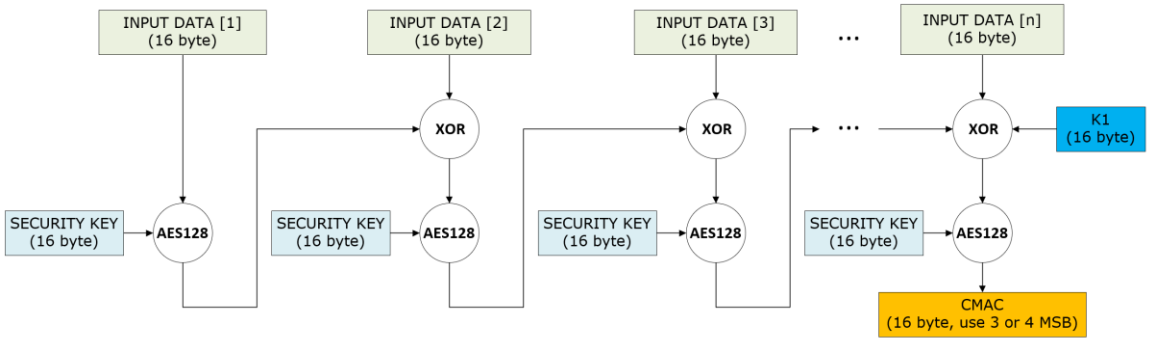


Figure 27. CMAC generation for input data with multiple of 16 bytes size

7.2.3 CMAC calculation for non-multiples of 16 byte

If the length of the input data is not exactly a multiple of 16 bytes, then the last part will have a size of less than 16-byte.

In this case, the last part will be filled with a padding value consisting of one byte of value 0x80 followed by as many bytes of value 0x00 required to reach 16-byte. If for instance the input data would be 13-byte long, then three bytes of padding would be required to reach 16-byte and these three bytes would have the value 0x800000.

The CMAC authentication signature will then be calculated according to the process shown in Figure 28 below using the Security Key and another key K2 which is derived from the Security Key according to the procedure defined in chapter 7.2.4.

Annex A provides step-by-step examples for the CMAC calculation.

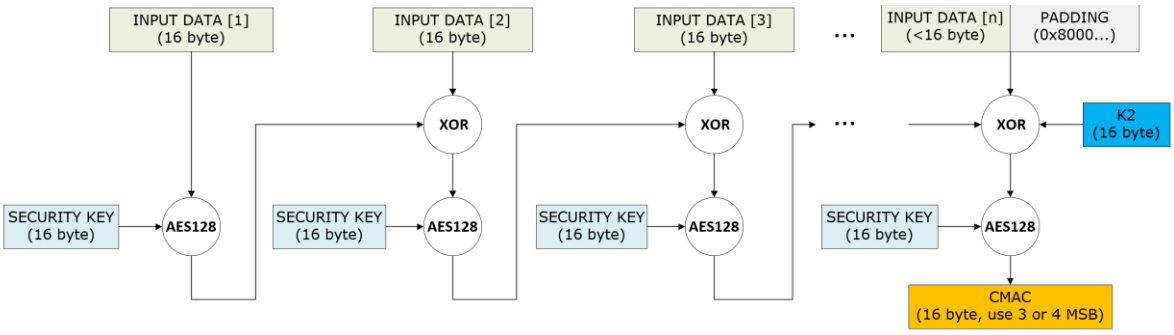


Figure 28. CMAC generation for input data with non-multiple of 16-byte size

7.2.4 Key derivation of K1 and K2 from the Security Key

As discussed above, the calculation of the CMAC authentication signature, which is defined by RFC4493 [7], requires the derivation of the keys K1 and K2 from the Security Key K.

The process for this key derivation is shown in Figure 29 for reference.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                    Algorithm Generate_Subkey                             +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                          +
+   Input    : K (128-bit Security Key)                                        +
+   Output   : K1 (128-bit first subkey)                                   +
+              K2 (128-bit second subkey)                                  +
+--------------------------------------------------------------------------+
+                                                                          +
+   Constants: const_Zero is 0x00000000000000000000000000000000           +
+              const_Rb   is 0x00000000000000000000000000000087           +
+   Variables: L          for output of AES-128 applied to 0^128          +
+                                                                          +
+   Step 1.  L := AES-128(K, const_Zero);                                  +
+   Step 2.  if MostSignificantBit(L) is equal to 0
+
+            then    K1 := L << 1;                                         +
+            else    K1 := (L << 1) XOR const_Rb;                          +
+   Step 3.  if MostSignificantBit(K1) is equal to 0
+
+            then    K2 := K1 << 1;                                        +
+            else    K2 := (K1 << 1) XOR const_Rb;                         +
+   Step 4.  return  K1, K2;                                                  +
+                                                                          +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 29. Key derivation of K1 and K2 from K

7.3 CCM Algorithm

RFC 3610 [5] defines the Counter with CBC-MAC (CCM) encryption and authentication algorithm and this specification should be used as primary source of information. The short term “CCM” is commonly used to describe this algorithm.

CCM implements a generic authenticated encryption block cipher mode. It is defined for use with 128-bit block ciphers and commonly used with the Advanced Encryption Standard (AES-128) [4].

CCM enables encryption, decryption and authentication of variable length payloads. Unlike for block ciphers such as AES-128, it is not required that input and output data lengths are a multiple of 16-byte. This makes CCM a good choice for length-constrained data encryption and decryption applications where the overhead (padding) for reaching a size that is a multiple of 16-byte blocks would not be desirable.

CCM is a well-established and well-researched algorithm that is used in a wide range of industry-leading protocols including Bluetooth, Zigbee, Wi-Fi, IPsec and TLS. CCM has been reviewed and approved by NIST [6].

7.3.1 Terminology

CCM can be used either to encrypt and authenticate or only to authenticate the provided input data. These two modes can be mixed, meaning that it is possible to encrypt and authenticate one part of the payload while another part of the payload is only authenticated.

The following terminology is used for the different parts:

- Authenticated Data *a*
  The Authenticated Data a is provided in plaintext and has a length (in bytes) of *l(a)*.

- Authenticated and Encrypted Data *e*
  The Authenticated and Encrypted Data *e* is the encrypted representation of the plaintext data *m* and has a length (in bytes) of *l(m)*.

- Authentication Signature *U*
  The Authentication Signature *U* authenticates the combination of the Authenticated Data *a* with the Authenticated and Encrypted Data *e* and has the length *M*.

This is illustrated in Figure 30 below.

Authenticated Data *a*	Authenticated and Encrypted Data *e*	Authentication Signature *U*
Plain Text	Encrypted representation of Data *m*	Encrypted representation of T
l(a) bytes	*l(m)* bytes	*M* bytes

Figure 30. Message Format

7.3.2 Input Data

Input data shall be provided to the CCM algorithm according to the following conventions:

- R-ORG-S
  R-ORG-S may either be 0x30 (SEC), 0x31 (SEC_R), 0x35 (SEC_TI) or 0x36 (SEC_DAT)

- R-ORG and DATA
  R-ORG and DATA shall be concatenated such that R-ORG is followed by DATA

- RLC
  RLC shall be provided using big-endian byte order and be 4-byte long.

- SENDER ID
  SENDER ID shall be provided using big-endian byte order and be 4-byte long.
  This means that the most significant two bytes (0x00:00) that are used to extend the 4-byte SENDER ID to a 6-byte SENDER ID in Security Teach-in messages in Japan shall be ignored for the purpose of security processing.

- DESTINATION ID
  DESTINATION ID shall be provided using big-endian byte order and be 4-byte long.
  For broadcast transmission, 0xFF:FF:FF:FF shall be used as DESTINATION ID.

CCM processing shall authenticate the following fields:

- SENDER ID
- DESTINATION ID
- RLC
- R-ORG-S

CCM processing shall encrypt and authenticate the following fields:

- R-ORG and DATA

Authentication and encryption shall be implemented according to the definitions provided in the subsequent chapters.

7.3.2.1 Nonce N

CCM requires a 13-byte initialization value (called "Nonce", abbreviated to "*N*") for each encryption and authentication operation. This Nonce must be unique within the context of the same Security Key meaning that the combination of a specific Nonce value and a specific Security Key must occur not more than once.

Nonce *N* shall be constructed from the concatenation of Source EURID, Destination EURID and RLC value with 1-byte 0x00 padding to reach the required length of 13-byte. This approach ensures that SENDER ID and DESTINATION ID are authenticated by the MAC authentication signature which allows using the same Security Key for more than one {Source EURID; Destination EURID} pair of devices.

Structure of Nonce N												
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
SENDER ID				DESTINATION ID				RLC				00

Figure 31. Structure for Nonce *N*

7.3.2.2 Example of Nonce N for a Broadcast Message

Figure 32 below shows Nonce *N* corresponding to the following parameters of a broadcast message:

- Source EURID: 0x01:02:03:04
- Destination EURID: 0xFF:FF:FF :FF (Broadcast)
- RLC: 0x0A:0B:0C:0D

Example of Nonce *N* for a Broadcast Message												
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
01	02	03	04	FF	FF	FF	FF	0A	0B	0C	0D	00

Figure 32. Example for Nonce *N* for a broadcast message

7.3.2.3 Example of Nonce N for an Addressed Data Message (ADT)

Figure 33 below shows Nonce *N* corresponding to the following parameters of an Addressed Data Message:

- Source EURID: 0x01:02:03:04
- Destination EURID: 0x06:07:08:09 (ADT)
- RLC: 0x0A:0B:0C:0D

Example of Nonce *N* for an Addressed Data Message												
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
01	02	03	04	06	07	08	09	0A	0B	0C	0D	00

Figure 33. Example for Nonce *N* for an Addressed Data Message (ADT)

7.3.2.4 Authenticated Data *a*

Authenticated Data *a* is present in plaintext in the message and authenticated together with the content of the Plaintext Data *m* by the MAC authentication signature.

7.3.2.5 Data Telegrams

For data messages, Authenticated Data *a* shall be 13-byte long and contain the following parameters:

- SENDER ID (4-byte)
- DESTINATION ID (4-byte, set to 0xFFFFFFFF for broadcast messages)
- RLC (4-byte)
- R-ORG-S (1-byte)

Figure 34 below shows the structure of Authenticated Data *a* for data messages.

Authenticated Data *a* for Data Messages												
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
SENDER ID				DESTINATION ID				RLC				R-ORG-S

Figure 34. Structure for Authenticated Data *a* for data messages

7.3.2.6 Security Teach-in Messages with PSK

For Security Teach-in messages using PSK in CCM mode, Authenticated Data shall be 15-byte long and contain the following parameters:

- SENDER ID (4-byte)
- DESTINATION ID (4-byte, set to 0xFFFFFFFF if not present)
- RLC (4-byte)
- R-ORG-S (1-byte, always 0x35 = SEC_TI)
- TEACH_IN_PARAMETERS (least significant 4 bit of Teach-In Info, upper 4 bit set to 0x0)
- Security Level Format (1-byte)

Note that only the content of the TEACH_IN_PARAMETERS field (the least significant 4-bit of the Teach-in Info field) shall be authenticated. The most significant 4-bit of the Teach-in Info field are used for telegram chaining and shall be set to 0b0000 for the purpose of authentication.

Figure 35 below shows the structure of Authenticated Data *a* for Security Teach-in messages using PSK.

Authenticated Data *a* for Security Teach-in Messages with PSK														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
SENDER ID				DESTINATION ID				RLC				R-ORG-S	TI PARAM	SLF

Figure 35. Structure for Authenticated Data *a* for Security Teach-in messages with PSK

7.3.3 Plaintext Data *m*

Plaintext Data *m* is the data that will be authenticated and encrypted resulting in Authenticated and Encrypted Data *e*.

7.3.3.1 Plaintext Data *m* for Data Messages

For data messages, plaintext data *m* shall be formed as combination between the original R-ORG (if present) and the corresponding DATA in the same way as for the VAES encryption algorithm described in Chapter 7.1.1 and shall be encrypted and authenticated using Security Key *K*.

7.3.3.2 Plaintext Data *m* for Security Teach-in Messages with PSK

For Security Teach-in messages using PSK, plaintext data *m* shall consist of the 16-byte Security Key *K* and be encrypted and authenticated using PSK as Security Key.

7.3.4 MAC Size (Size of Authentication Signature *U*)

The MAC size *M* shall always be 4-byte within the scope of security processing of ERP messages.

7.3.5 Configuration Parameters

CCM uses the following configuration parameters:

- Adata (Additional authenticated data present)
  The field Adata is set to 0b1 if data that is only authenticated (but not encrypted) is present and to 0b0 if all data is both encrypted and authenticated.
  Data that is only authenticated is present when CCM is used for security processing of ERP messages; this field is therefore set to 0b1.

- M (Size of authentication signature)
  M designates the size of the authentication signature and may be an even value with 4<=M<=16.
  The length of the authentication signature is present when CCM is used for security processing of ERP messages; this value is therefore set to 4.

- L (Size of the length field l(a) and l(m))
  L designates the size of the length field which may be a value of 2 <= L <= 8
  The minimum value of 2 allows encoding length of up to 65.535 bytes; therefore, this value is used when CCM is used for security processing of ERP messages.

7.3.5.1 Encoded Configuration Parameters

CCM encodes the configuration parameters *M* and *L* listed above as follows:

- M' (Binary encoded authentication signature length)
  M' is the 3 bit encoding of M (length of authentication signature) formed as: M' = M/2 -1
  M' is therefore 0b001 for the authentication signature length of 4-byte.

- L' (Binary encoded length field size)
  L' is the 3 bit encoding of the length of L formed as: L' = L -1
  L' is therefore 0b001 for a 2-byte length of L.

Table 6 below summarizes the encoding of the configuration parameters.

Parameter Encoding	Content	Setting
Adata	Encoded presence of data that will only be authenticated	0b1: Authentication-only data is present
M'	Encoded signature length	0b001: 4-byte signature size
L'	Encoded length field size	0b001: 2-byte length field size

Table 6. Encoded configuration parameters

7.4 CCM Authentication

CCM implements authentication as sequence of AES and XOR operations operating on fixed-size blocks, the so-called *B Blocks* (B_0 … B_i) where the first block B_0 contains initialization data and the subsequent blocks B_1 … B_i contain the input data that shall be authenticated.

The sequence of AES and XOR operations generates a temporary authentication signature *T*, which is then encrypted and truncated resulting in the final authentication signature *U*.

7.4.1 B Block Structure

7.4.1.1 B0 Flag

Adata, *M'* and *L'* are used to form the 1-byte *Flag* used in B_0 block ("*B0 Flag*") as shown in Table 7.

Bit Position	Content	Value
7	Reserved, set to 0b0	0b0 (always)
6	*Adata*	0b1: Authenticated Data *a* is present
5 … 3	*M'*	0b001: 4-byte signature
2 … 0	*L'*	0b001: 2-byte size
7 … 0	Resulting B0 Flag	0x49: 4-byte signature Authenticated Data a is present

Table 7. *B0 Flag* structure

7.4.1.2 B0 Block

B_0 block is used as initialization vector for the subsequent authentication operations. *B0 Flag* is used together with Nonce *N* and the size of the encrypted data *l(m)* to form block B_0 as shown in Table 8.

Byte Position	Content	Value
0	*B0_Flag*	0x49: Not all data encrypted, 4-byte signature
1 … 13	*N*	Variable (Nonce N)
14 … 15	*l(m)*	Variable (Length of Plaintext Data *m*, big endian format)

Table 8. B_0 block structure

7.4.1.3 B Blocks for Authenticated Data *a*

Authenticated Data *a* (data that is only authenticated, but not encrypted) is present as described in chapter 7.3.2.4; Block B_1 is therefore formed as shown in Table 9 and used to authenticate this data.

The size of Authenticated Data *a* is 13-byte for data messages and only one block (B_1) is required for the authentication of this data. One byte of zero padding (using 0x00) will be used to reach the total size of 16-byte for B_1.

Byte Position	Content	Value
0 … 1	*l(a)*	Length of Authenticated Data *a* (0x000D = 13 byte)
2 … 14	*a*	Authenticated Data *a* (13 byte)
15	*ZERO*	Zero padding (1 byte of 0x00)

Table 9. B_1 block structure for Authenticated Data of data messages

The size of Authenticated Data *a* is 15-byte for Security Teach-in messages with PSK meaning that two blocks – B_1 and B_2 - are required. Table 10 shows the format of B_1 for Security Teach-in messages with PSK.

Byte Position	Content	Value
0 … 1	*l(a)*	Length of Authenticated Data *a* (0x000F = 15 byte)
2 … 15	*a*	First part of Authenticated Data *a* (14 byte)

Table 10. B_1 block for first part for Authenticated Data of Security Teach-in messages with PSK

Table 11 shows the format of B_2 for Security Teach-in messages with PSK.

Byte Position	Content	Value
0	*a*	Second part of Authenticated Data *a* (1 byte)
1 … 15	*ZERO*	Zero padding (15 byte of 0x00)

Table 11. B_2 block for second part of Authenticated Data of Security Teach-in messages with PSK

7.4.1.4 B Blocks for Plaintext Data *m*

For data messages, Plaintext Data *m* (which shall be authenticated and encrypted) shall be allocated into block B_2 … Bi starting with B_2.

If the length of Plaintext Data *m* is not a multiple of 16-byte, then the last block shall be filled to 16-byte by padding with Zero (0x00) as shown in Table 12 below.

Byte Position	Content	Value
0 … *n*	*m*	Plaintext Data *m*
n+1 … 15	*Zero*	Padding with Zero (0x00) as required to reach 16-byte

Table 12. B_2 … B*i* blocks for data messages

For Security Teach-in messages with PSK, B_2 shall contain the Security Key *K* as shown in Table 13 and no other B blocks shall be present.

Byte Position	Content	Value
0 … 15	*m*	Security Key *K* (16 byte)

Table 13. B_2 block for Security Teach-in messages with PSK

7.4.2 Authentication Processing

The CCM authentication algorithm is structured such that any change of the input data leads to a change in the authentication data.

The authentication processing is implemented as a sequence of encryption and XOR operations which ensures that the calculated output value *T* is dependent on the entire authenticated payload B_1 … B_i and the initialization vector B_0.

The encryption operations use the AES-128 block cipher to encrypt 16-byte input blocks using the Security Key *K* resulting in 16-byte output blocks.

The variable initialization vector B_0 contains Nonce *N* to ensure that execution of this algorithm on the same input payload, but with different Source Address, Destination Address or RLC value will result in different signatures. This prevents the replay of previously transmitted messages (thanks to RLC tracking) and the modification of Source or Destination Address of a previously transmitted message.

The message fields that are only authenticated (R-ORG-S, Sender ID, Destination ID, RLC) are allocated in Block B_1 and the message payload (R-ORG and DATA fields) that is authenticated and encrypted is allocated in Blocks B_2 … B_i as described in the previous chapter.

Figure 36 below shows the processing of these blocks to generate the Temporary Signature *T* which is subsequently encrypted as described in Chapter 7.5.5.

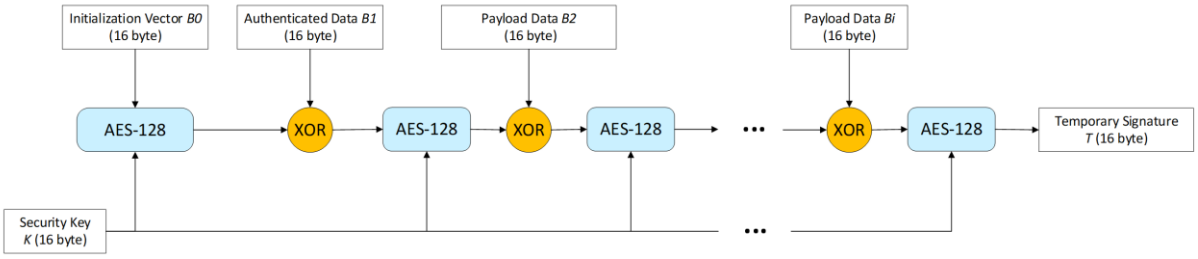


Figure 36. Payload authentication processing

7.5 CCM Encryption and Decryption

CCM implements encryption and decryption of the payload and the authentication signature by generating a keystream *S* {S_0; S_1; …; S_n} which is used as input to an XOR operation that transforms input data to encrypted data and vice versa.

7.5.1 A Flag

The field *L'* is used to form the 1-byte *Flag* value used in all *A* blocks (*"A Flag")* described in Table 14. Within the scope of the EnOcean Alliance Security specification, *A Flag* is always 0x01.

Bit Position	Content	Value
7 … 6	Reserved, must be 0b00	0b00 (always)
5 … 3	Zero (set to 0b000)	0b000 (always)
2 … 0	*L'*	0b001 (always)
7 … 0	Resulting A Flag	0x01

Table 14. *A Flag* structure

7.5.2 A0 … Ai Blocks

The combination of *A Flag,* the Nonce *N* and an incrementing counter *i* is used to form the 16-byte auxiliary data blocks *A0*, *A1*, … *Ai* ("*A Blocks*") as shown in Table 15 which are used to generate the keystream *S*.

Byte Position	Content	Value
0	*A Flag*	0x01
1 … 13	*N* (Nonce)	Variable
14 … 15	*i*	0 … *i*

Table 15. A_0 … A_i block structure

7.5.3 Key Stream S

The auxiliary data blocks A {A_0; A_1; …; A_n} are used together with the Security Key K to generate the key stream S via the AES-128 encryption as shown in Figure 37 below.

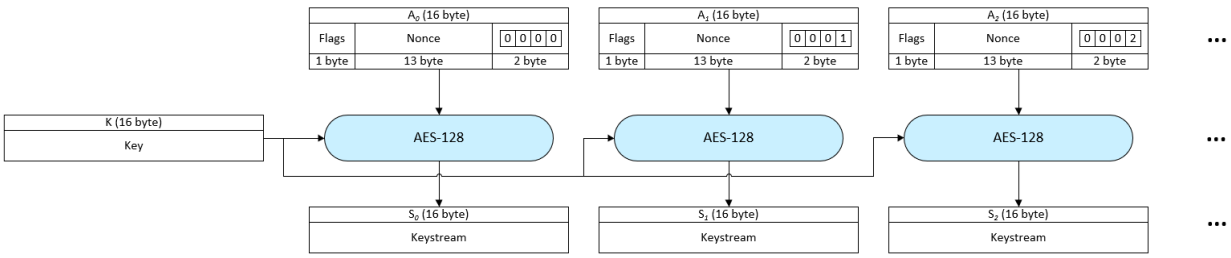


Figure 37. Keystream generation

7.5.4 Encryption and Decryption of R-ORG and DATA

The generated keystream S_1 … S_i is used to encrypt / decrypt the Plaintext Data blocks m_1 … m_i using an XOR operation.

When used for the security processing of ERP radio messages, the Plaintext Data blocks m_1 … m_i is the concatenation of original R-ORG (if present) and DATA.

The XOR operation used for encryption is easily reversible by the receiver (to decrypt the message payload) based on the possession of the same keystream blocks. Figure 38 illustrates this.

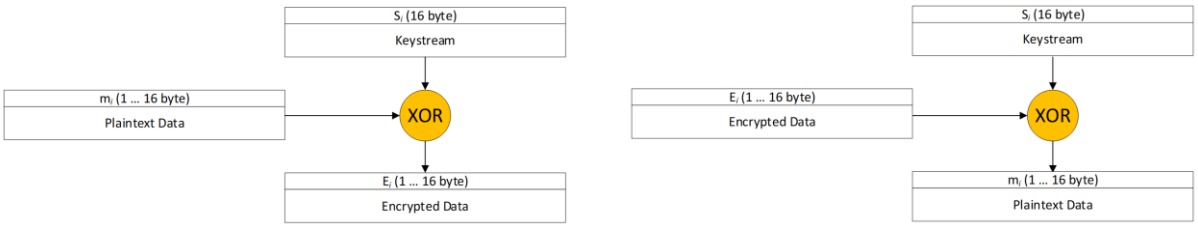


Figure 38. Encryption / decryption of the plaintext payload

7.5.5 Encryption and Decryption of the Authentication Signature

The keystream block S_0 is used to encrypt the 16-byte output *T* of the authentication algorithm resulting in the authentication signature *U.* This is shown in Figure 39 below.

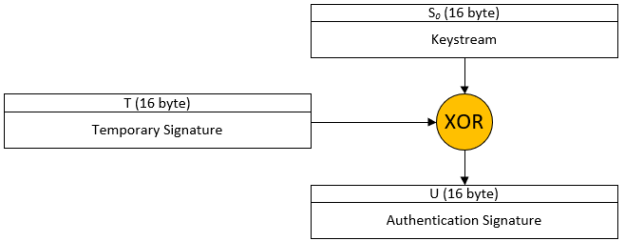


Figure 39. Encryption / decryption of the authentication signature

The calculated authentication signature *U* shall be truncated to the desired length *M* of 4-byte by using the most significant four bytes of *U*. This truncated value shall then be transmitted in the MAC signature field of the ERP radio message.

The receiver shall calculate the expected authentication signature *T* based on the received and decrypted payload data, encrypt it with the keystream block S_0 and compare the four most significant bytes of the result with the signature from the MAC field of the received ERP message to determine the authenticity of the received message.

7.6 Key Derivation

For certain applications with high security requirements, it may be desirable not to widely use the shared Security Key (primary key) *K* to limit the exposure of this key.

Using a secondary (derived) key *K1* that is restricted in time (temporary key) or scope (device-specific key) may be preferable to using the primary key *K* in such cases.

The derivation of a secondary key *K1* from a primary key K is done via the Key Derivation Function (KDF) specified below. This uses a Key Derivation Function (KDF) in counter mode based on the CMAC algorithm defined in [7]. In this implementation, only one output is generated and therefore only the counter value 0x01 is used.

7.6.1 Input Parameters

The KDF defined for ERP messages requires the following two input parameters:

- Primary Security Key *K*
  The primary Key K shall always be 16 byte long

- Context *C* (0 … 16 byte)
  The Context parameter allows deriving different secondary Security Keys from the same primary Security Key depending on context parameters such as addresses or counters.
  If the Context is a constant (including not being present), then the secondary Security Key derived from the primary key will also be constant, i.e. it will always be the same.

7.6.2 Internal Parameters

The KDF defined for ERP messages uses one internal parameter:

- Initialization Value *I* (0 … 16 byte)

The Initialization Value *I* may be set to any value provided that the least significant byte (the counter) is set to 0x01 for consistency with the definition of KDF in counter mode.

For the use of this KDF with ERP messages, the Initialization Value *I* shall always be set as follows:

I = “enocean” || 0x01 = 0x656E6F6365616E01

7.6.3 Key Derivation Process

The Context (Initialization Value) *C* shall be hashed using the CMAC function with a 16-byte all zero key *Z* to create the 16-byte Salt value *S*:

Z = 0x00000000000000000000000000000000

$S = CMAC_Z(C)$

The Salt value *S* shall then be used as Key of the CMAC function which derives a temporary key T from the Long-Term key *K* (which is the shared secret):

$T = CMAC_S(K)$

The temporary key T therefore depends on the Primary key *K* and the context *C*. It is then used for the subsequent derivation of the secondary key *K1*. The derivation of the secondary key *K1* uses one CMAC processing round with the temporary key *T* and the Initialization Value *I*:

$K1 = CMAC_T(I)$

This key derivation process is illustrated in Figure 40 below and a step-by-step example is provided in Appendix Annex B.

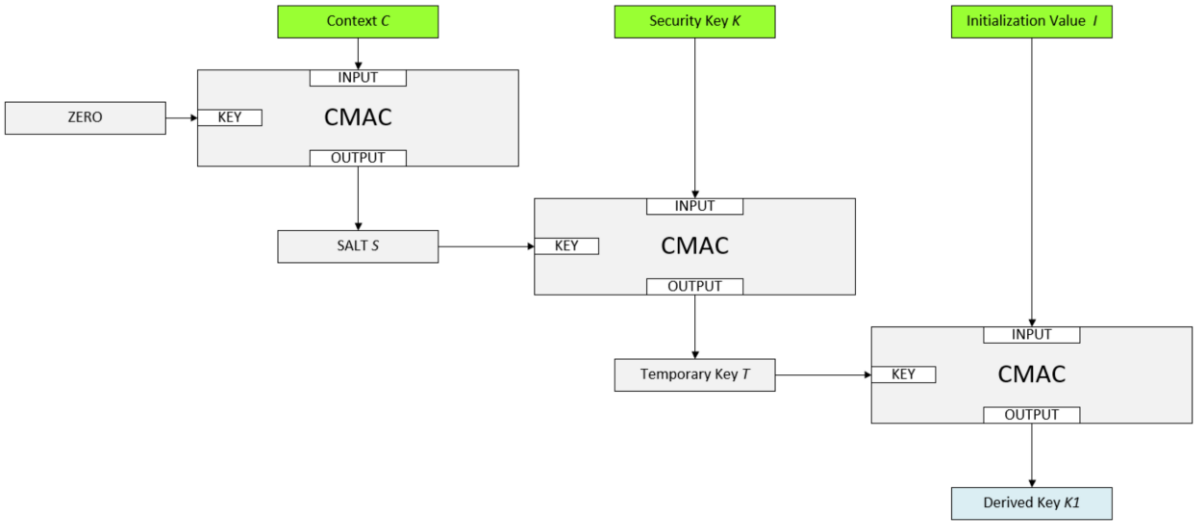


Figure 40. Key Derivation Process

8 Secure Message Format

ERP secure messages may use either ERP1 or ERP2 telegram format depending on the region where the device is used.

Additionally, ERP secure messages may either be transmitted as secure broadcast message or as secure addressed message which identifies the intended receiver of the message. The latter case is not common due to the required overhead for specifying the intended receiver.

The following chapters describe the format of common message types using ERP1 and ERP2 telegram format.

8.1 Secure Data Message Types

The following chapters provide examples for the format of different secure message types using both ERP1 and ERP2 radio mode. Fields such as STATUS, CRC / HASH that are not relevant for security processing are omitted in the illustrations.

8.1.1 Secure Broadcast Message without R-ORG

Figure 41 below illustrates the format of an ERP secure broadcast message without R-ORG and the resulting secure message structure for ERP1 (shown on the top) and ERP2 (shown on the bottom). This format is mainly used for pushbutton switch messages.

Note that the payload of secure broadcast messages does not include the R-ORG of the original message.

The security processing of secure broadcast messages without R-ORG will encrypt the content of the DATA field and authenticate the content of the Encrypted Data and R-ORG-S fields. For the case of CCM security mode, SENDER ID and DESTINATION ID (which is assumed to be 0xFF:FF:FF:FF as it is not present in Broadcast messages) will also be authenticated.

As discussed in Chapter 6.5, security processing for ERP2 messages uses the actual R-ORG-S value (0x30 in this case) for authentication, not its encoded representation (0x7) in HEADER. Security processing of the same input data (R-ORG-S, DATA, RLC, SENDER ID) will therefore result in the same output data (ENCRYPTED DATA and MAC) both for ERP1 and ERP2 messages.

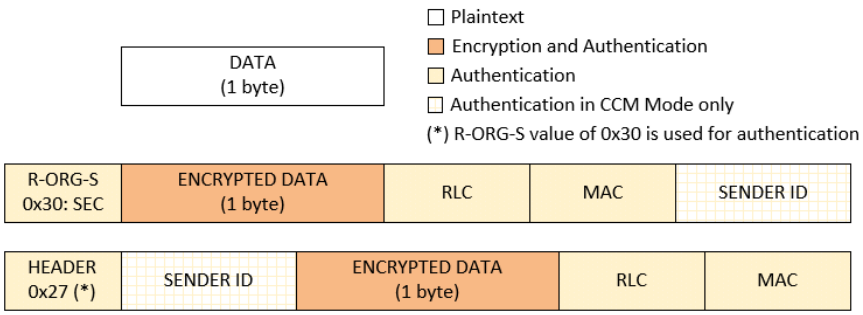


Figure 41. Format of secure RPS broadcast messages (without EEP) for ERP1 (top) and ERP2 (bottom)

8.1.2 Secure Broadcast Message with R-ORG

Figure 42 below illustrates the format of an ERP secure broadcast message with R-ORG and the resulting message structure for ERP1 (shown on the top) and ERP2 (shown on the bottom).

SENDER ID and DESTINATION ID (which is assumed to be 0xFF:FF:FF:FF as it is not present in Broadcast messages) are part of the authenticated payload in CCM security mode only.

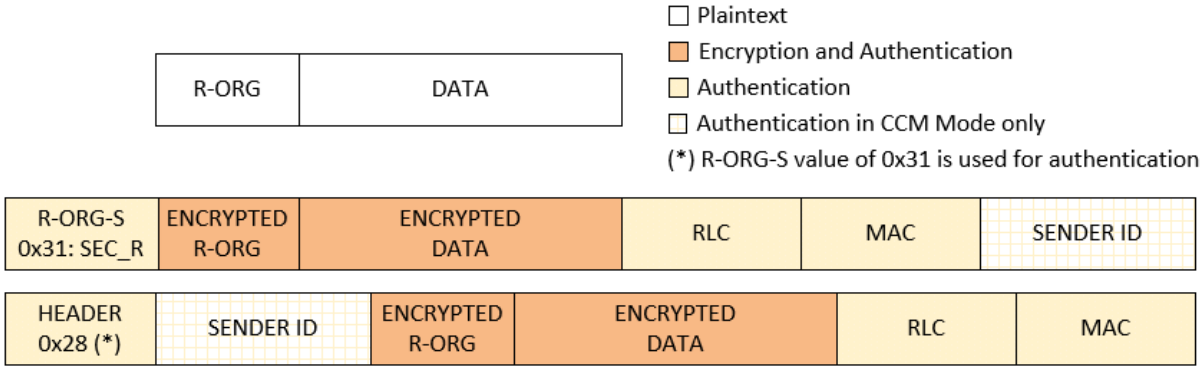


Figure 42. Format of secure broadcast data messages with R-ORG for ERP1 and ERP2

8.1.3 Secure Addressed Telegram with R-ORG

Figure 43 below illustrates the security processing of an ERP addressed message and the message telegram structure for ERP1 (shown on the top) and ERP2 (shown on the bottom).

Note that this message type is not commonly used due to the substantial overhead of the addressing fields.

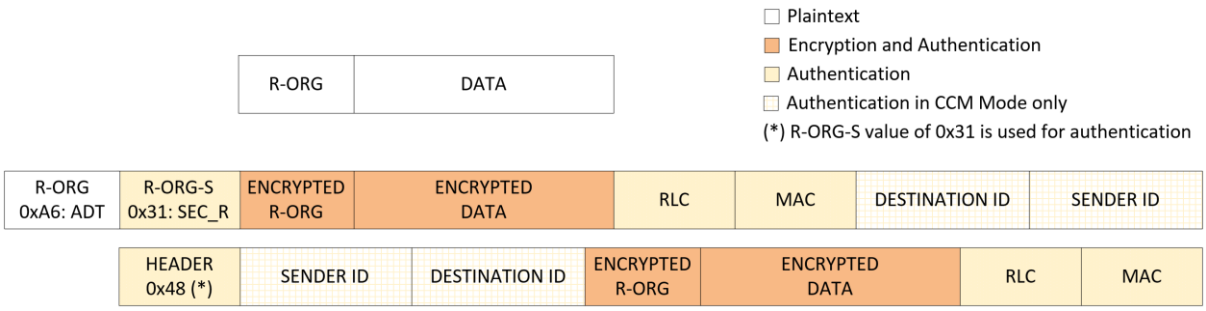


Figure 43. Format of secure addressed data messages with R-ORG for ERP1 and ERP2

8.1.4 Secure Binary Data Transport Message (SEC_DAT)

Secure Binary Data Transport messages are identified by R-ORG-S 0x36: SEC_DAT and are used to transport binary data for purposes such as over the air (OTA) data transfer.

- Security Mode
  Secure Binary Data telegrams shall use CCM as Security Mode; using legacy VAES mode is only permissible for the purpose of the initial discovery of available devices if required for backwards-compatibility.

- RLC Size
  Secure Binary Data telegrams may either contain the full 4-byte RLC or a reduced 1-byte RLC. Transmission of the full 4-byte RLC is mainly intended for setup and control messages while the reduced 1-byte RLC mode is intended for data transport.

- MAC Size
  Secure Binary Data telegrams shall always contain a 4-byte authentication signature (MAC); use of 3-byte authentication signature is not permissible.

- DATA (Payload) Size
  Secure Binary Data telegrams may transport up to 6-byte of DATA Payload when using 4-byte RLC size or up to 9-byte of DATA payload when using 1-byte RLC size. Secure Binary Data messages shall not use chaining (SEC_CDM) to extend this maximum DATA size.

The format of Secure Binary Data Transport messages is shown in Figure 44 below.

R-ORG-S	DATA	RLC	MAC
0x36: SEC_DAT	1 … 6/9-byte	4/1-byte	4-byte

Figure 44. Format of Secure Binary Data Transport (SEC_DAT) messages

8.2 Secure Chained Data Messages

This chapter describe the format of secure chained data messages. Appendix A.3 provides a step-by-step example for the security processing of a secure message chain.

8.2.1 First Telegram

Figure 45 shows the structure for the first telegram of a secure message chain for ERP1 (shown on the top) and ERP2 (shown on the bottom).

As described in Chapter 6.4.1.1, the CHAIN_CTRL field identifies the CHAIN ID and the INDEX of each telegram in the chain using the structure defined in Table 5. The CHAIN_LENGTH field – which is only present in the first telegram of a secure message chain - specifies the combined length (in bytes) of the following fields:

- ENCRYPTED R-ORG (always 1-byte)
- ENCRYPTED DATA (for all parts of the message chain)
- RLC
- MAC

The ENCRYPTED DATA field of the first telegram shall always contain 10-byte of payload.

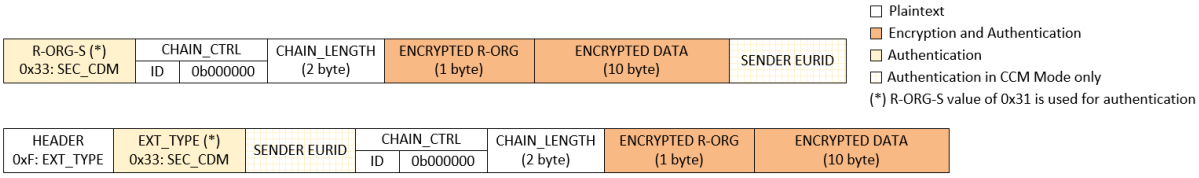


Figure 45. Format for first telegram of a secure message chain

8.2.2 Intermediate Telegrams

Figure 46 shows the structure for an intermediary (not the first and not the last) telegram of a secure message chain. This telegram type is optional and present if the combined length of payload, RLC and MAC is more than 23-byte.

The example below shows uses an INDEX value of 0b000001 as example; the same format applies if more than one intermediate telegram is present.

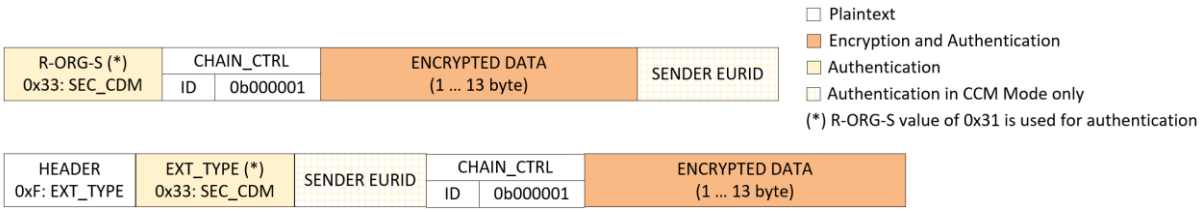


Figure 46. Format for intermediary (not first, not last) telegrams of a secure message chain

8.2.3 Last Telegram

Figure 47 below shows the structure for the last telegram of a secure message chain for ERP1 (shown on the top) and ERP2 (shown on the bottom). This telegram type is mandatory for a secure message chain.

The maximum length of the combination of ENCRYPTED DATA, RLC and MAC fields is 13-byte meaning that the last telegram can carry a payload of up to (13 – MAC SIZE – RLC SIZE) byte.

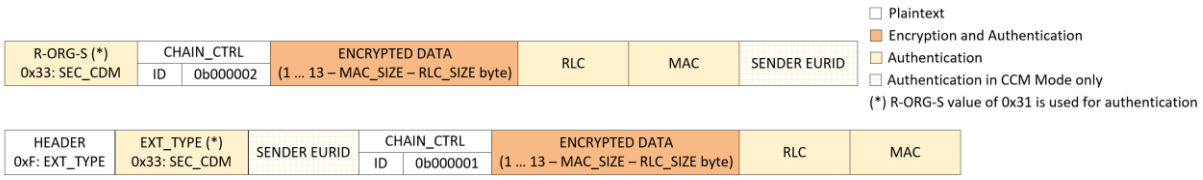


Figure 47. Format of last telegram of a secure message chain

8.3 Security Teach-in Messages

Security Teach-in messages provide the information required for the Security Teach-in process described in Chapter 5.3.

8.3.1 Security Teach-in Message Format

Security Teach-in messages are identified by R-ORG-S 0x35: SEC_TI. For ERP2, this R-ORG-S is encoded by setting HEADER to 0x29. Security Teach-in messages contain the following parameters:

- Security Level Format (SLF) described in Chapter 4.6
- Teach-in Info (TII) described in Chapter 8.3.2
- RLC value (either 3-byte or 4-byte depending on the selected size) that will be used next
- Security Key

The payload size of the Security Teach-in messages is larger than the maximum payload size for ERP1 telegrams; therefore, such messages must be transmitted using two telegrams as shown in the examples provided in Annex A.

For ERP2, the Security Teach-in message may be transmitted as one telegram in some regions (e.g. Japan) while two telegrams are required in other regions depending on local radio.

Some parameters provided by Security Teach-in messages may be protected by a pre-shared security key (PSK) as described in chapter 5.3.2.

8.3.2 Teach-in Info

The Teach-in Info field provides the required control fields for message chaining (if two telegrams are required for transmission of the Security Teach-in message) and specifies certain teach-in parameters.

8.3.3 Format of Teach-in Info Field

Teach-in Info uses the format shown in Figure 48.

<table>
<tr><td colspan="8">Teach-in Info</td></tr>
<tr><td>Bit 7</td><td>Bit 6</td><td>Bit 5</td><td>Bit 4</td><td>Bit 3</td><td>Bit 2</td><td>Bit 1</td><td>Bit 0</td></tr>
<tr><td colspan="4">CHAIN CONTROL</td><td colspan="4">TEACH-IN PARAMETERS</td></tr>
<tr><td colspan="2">INDEX</td><td colspan="2">CNT</td><td>PSK</td><td>TYPE</td><td colspan="2">INFO</td></tr>
</table>

Figure 48. Teach-in Info format

For ERP1 and most ERP2 systems, it is required to split the Security Teach-in message into a chain of two telegrams since the payload is longer than the maximum size of one ERP1 or ERP2 telegram.

In this case, the INDEX field of Teach-in Info will indicate the order of the two telegrams with the first telegram having INDEX = 0b00 and the second telegram having INDEX = 0b01.

The first telegram (with INDEX = 0b00) will provide additional information in the CNT, PSK, TYPE and INFO fields as follows:

- CNT
  This field will identify the total number chained telegrams used for the transmission of the Security Teach-in message. If this field is set to 0b01, then one telegram is sufficient for transmitting the Security Teach-in message. If this field is set to 0b10, then two telegrams are required for transmitting the Security Teach-in message.

- PSK
  If this field is set to 0b1, then a Pre-shared Security Key (PSK) is used to encrypt RLC and Security Key. If this field is set to 0b0, the RLC and Security Key are not encrypted with a Pre-shared Security Key (PSK).

- TYPE
  If this field is set to 0b1, then the Security Teach-in message was sent by a PTM switch module; otherwise, this field is set to 0b0.

- INFO
  If the device is a PTM switch module, then this field will identify which rocker was pressed to trigger the Security Teach-in message transmission: 0b00 indicates that Rocker A was used while 0b01 indicates that Rocker B was used.
  If the device is not a PTM module, then this field will identify the Security Teach-in procedure type: 0b00 indicates that a uni-directional Teach-in is executed while 0b01 indicates that a bi-directional Security Teach-in is executed.

8.3.3.1 Teach-in Info for First Telegram of a Non-PTM Device

Figure 49 below shows the structure of the Teach-in Info field for the first telegram of a Security Teach-in message chain for the case where the device is not a PTM module.

Teach-in Info Field for the First Telegram of a Non-PTM Device							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	
0b00: First message		0b10: 2 chained messages used to transmit the Security Teach-in telegram		0b0: PSK not used 0b1: RLC and Security Key are encrypted with PSK	0b0: Device is not a PTM	0b00: Unidirectional security Teach-in procedure 0b01: Bidirectional Teach-in procedure 0b10, 0b11: Reserved	

Figure 49. Teach-in Info format of first telegram (IDX = 0b00) and TYPE = 0b0 (non-PTM device)

8.3.3.2 Teach-in Info for First Telegram of a PTM Device

Figure 50 below shows the structure of the Teach-in Info field for the first telegram of a Security Teach-in message chain for the case where a device is a PTM module.

Teach-in Info Field for the First Telegram of a PTM Device							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	
0b00: First message		0b10: 2 chained messages used to transmit the Security Teach-in telegram		0b0: PSK not used 0b1: RLC and Security Key are encrypted with PSK	0b1: Device is a PTM	0b00: ROCKER A normal Teach-in 0b01: ROCKER B normal Teach-in 0b10, 0b11: Reserved	

Figure 50. Teach-in Info format of first telegram (IDX = 0b00) and TYPE = 0b1 (PTM device)

8.3.3.3 Teach-in Info for Second Telegram

Figure 51 below shows the structure of the Teach-in Info field for the second telegram of a Security Teach-in message chain. This format applies to all device types.

Teach-in Info Field for the Second Telegram							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	
0b01: Second message		0b00: Reserved		0b0: Reserved	0b0: Reserved	0b00: Reserved	

Figure 51. Teach-in Info format of second telegram (IDX = 0b01)

8.3.4 Security Teach-in Message without Chaining

The ERP2 implementation specific to Japan allows transmission of a Security Teach-in message using only one telegram.

For the case where only one telegram is required, the structure shown in Figure 52 would be used. Note that this case can only occur for ERP2.

HEADER	TII (1-byte)					SLF (1-byte)	RLC (3/4-byte)	Security Key (16-byte)
0x29	0b00	0b01	PSK	TYPE	INFO	Variable	Next RLC	Security Key

Figure 52. Format of an ERP Security Teach-in telegram in VAES mode

8.3.5 Security Teach-in Message with Chaining

Security Teach-in messages must be sent using two telegrams for most implementations of ERP as the message payload size exceeds the maximum payload size of an ERP telegram. The message payload distribution across the two telegrams depends on the size of the transmitted RLC value:

- VAES with CMAC mode
  RLC size can be either 3-byte or 4-byte and therefore the first part of the Security Key that is transmitted in the first SEC_TI telegram is either 9 or 8 bytes long and the second part of the Security Key that is transmitted in the second SEC_TI telegram is either 7 or 8 bytes.

- CCM Mode
  RLC size is always 4-byte and therefore the first part of the Security Key that is transmitted in the first SEC_TI telegram is always 8 bytes long and the second part of the Security Key that is transmitted in the second SEC_TI telegram is always 8 bytes long.

No time-out value is defined for the sequence of these two telegrams in this message chain as long as both telegrams are received by the end of Security Teach-in mode. Partly received messages (where only one of the two telegrams was received by the end of Security Teach-in mode) shall be deleted.

The content of the first telegram is shown in Figure 53 below.

HEADER	TII (1-byte)					SLF (1-byte)	RLC (3/4-byte)	Security Key (9/8-byte)
	IDX	CNT	PSK	TYPE	INFO			
0x35 or 0x29	0b00	0b10	0b0	Var.	Var.	Variable	Next RLC	Security Key (MSB Part)

Figure 53. Format of the first telegram of an ERP Security Teach-in message in VAES mode

The content of the second telegram is shown in Figure 54 below.

HEADER	TII (1-byte)					Security Key (7/8-byte)
	IDX	CNT	PSK	TYPE	INFO	
0x35 or 0x29	0b01	0b00	0b0	0b0	0b00	Security Key (LSB Part)

Figure 54. Format of the second part of an ERP Security Teach-in message in VAES mode

8.3.6 Security Teach-in Message with PSK

Using a PSK allows encrypting (and for the case of CCM security mode also authenticating) the content of Security Teach-in telegrams. This mode is indicated by setting the PSK field of Teach-in Info to 0b1 as described in Chapter 8.3.2.

If the security mode is VAES with CMAC and PSK is used, then the concatenation of RLC and SECURITY KEY is encrypted with PSK and placed into the RLC and SECURITY KEY fields.

If the security mode is CCM and PSK is used, then the content of the Security Teach-in telegram is processed as follows:

- Security Key is encrypted as Authenticated and Encrypted Data *e*
- SENDER ID, DESTINATION ID (0xFF:FF:FF:FF), RLC, R-ORG-S, Teach-in Parameters (the lowest 4-bit of TII with the upper 4-bit set to zero) and SLF are authenticated as Authenticated Data *a*
- The authentication signature is provided by the 4-byte MAC field

For the case where only one telegram is required, the structure shown in Figure 55 would be used. Note that this case can only occur for ERP2.

HEADER	TII (1-byte)					SLF (1-byte)	RLC (4-byte)	Security Key (16-byte)	MAC (4-byte)
	IDX	CNT	PSK	TYPE	INFO				
0x35 or 0x29	0b00	0b01	0b1	Var.	Var.	Variable	RLC	Security Key	MAC

Figure 55. Format of an ERP Security Teach-in message in CCM mode with PSK

If two telegrams must be used to transmit the Security Teach-in message, then the content of the first telegram is shown in Figure 56 below.

HEADER	TII (1-byte)					SLF (1-byte)	RLC (4-byte)	Security Key (8-byte)
	IDX	CNT	PSK	TYPE	INFO			
0x35 or 0x29	0b00	0b10	0b1	Var.	Var.	Variable	RLC	Security Key (MSB Part)

Figure 56. Format of first telegram of an ERP Security Teach-in message in CCM mode with PSK

The content of the second telegram is shown in Figure 57 below.

HEADER	TII (1-byte)					Security Key (8-byte)	Authentication Signature (4-byte)
	IDX	CNT	PSK	TYPE	INFO		
0x35 or 0x29	0b01	0b00	0b0	0b0	0b00	Security Key (LSB Part)	MAC

Figure 57. Format of second telegram of an ERP Security Teach-in message in CCM mode with PSK

No time-out value is defined for the sequence of these two messages in this message chain as long as both messages are received by the end of Security Teach-in mode.

Partly received messages (where only one of the two messages was received by the end of Security Teach-in mode) shall be deleted.

9 References

[1] RFC 2119: Key words for use in RFCs to Indicate Requirement Levels
https://datatracker.ietf.org/doc/html/rfc2119

[2] ISO 14543-3-10
https://webstore.iec.ch/en/publication/63311

[3] ISO 14543-3-11
https://webstore.iec.ch/en/publication/24212

[4] NIST FIPS 197: Advanced encryption standard (AES), 2001
https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

[5] RFC 3610: Counter with CBC-MAC (CCM)
https://www.rfc-editor.org/rfc/rfc3610

[6] NIST SP 800-38C: Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality
https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final

[7] RFC 4493: The AES-CMAC Algorithm
https://www.rfc-editor.org/rfc/rfc4493

[8] Product ID and Standardized Labeling Specification
https://www.enocean-alliance.org/productid/

[9] AVR411: Secure Rolling Code Algorithm for Wireless Link
https://www.microchip.com/content/dam/mchp/documents/OTH/ApplicationNotes/ApplicationNotes/Atmel-2600-AVR411-Secure-Rolling-Code-Algorithm-for-Wireless-Link_Application-Note.pdf

Annex A - Secure Message Examples

All examples use hexadecimal notation unless otherwise specified.

A.1 Secure PTM Message

This example illustrates how a PTM 21x switch module would use VAES and CCM security modes.

A.1.1 Example parameters

This example assumes the following message parameters:

- SENDER ID: FE D8 9C 22
- DESTINATION ID: FF FF FF FF (Broadcast)
- DATA (D2-03-00): 0B (Button B1 is pressed)

Additionally, this example assumes the following security parameters:

- VAES parameters: 3-byte Transmitted RLC / 3-byte CMAC (SLF: AB)
- CCM parameters: 2-byte Transmitted RLC / 4-byte CMAC (SLF: 31)
- Security Key (K): 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C
- Device Type: PTM (switch) module
- Teach-in Action: Rocker A pressed

A.1.2 Security Teach-in Message

A.1.2.1 VAES Mode

The Security Teach-in message encodes the following parameters:

- R-ORG: 35 (SEC_TI)
- HEADER (ERP2): 29 (Broadcast, SEC_TI)
- Teach-Info: 24 (First Part of 2) / 40 (Second Part of 2) / 14 (First Part of 1)
  Uni-directional, PSK Not Used, Is a PTM
- RLC: 00 00 37
- SLF: AB (Mode = VAES, MAC Size = 3 byte, Transmitted RLC Size = 3 byte)
- Security Key: 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C

Transmitted Security Teach-in telegrams for ERP1:

- First telegram: 35 24 AB 00 00 37 8E 02 D4 9A 10 C1 AE 40 45
- Second telegram: 35 40 B5 99 C2 9A 05 A2 7C

Transmitted Security Teach-in telegrams for ERP2 when two telegrams are used:

- First telegram: 29 24 AB 00 00 37 8E 02 D4 9A 10 C1 AE 40 45
- Second telegram: 29 40 B5 99 C2 9A 05 A2 7C

Transmitted Security Teach-in telegram for ERP2 when one telegram is used:

- Telegram: 29 14 AB 00 00 37 8E 02 D4 9A 10 C1 AE 40 45 B5 99
  C2 9A 05 A2 7C

A.1.2.2 CCM Mode

The Security Teach-in message encodes the following parameters:

- R-ORG: 35 (SEC_TI)
- HEADER (ERP2): 29 (Broadcast, SEC_TI)
- Teach-Info: 24 (First Part of 2) / 40 (Second Part of 2) / 14 (First Part of 1)
  Uni-directional, PSK Not Used, Is a PTM
- RLC: 00 00 00 37
- SLF: 31 (Mode = CCM, MAC Size = 4 byte, Transmitted RLC Size = 2 byte)
- Security Key (K): 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C

Transmitted Security Teach-in telegrams for ERP1:

- First telegram: 35 24 31 00 00 00 37 8E 02 D4 9A 10 C1 AE 40
- Second telegram: 35 40 45 B5 99 C2 9A 05 A2 7C

Transmitted Security Teach-in telegrams for ERP2 when two telegrams are used:

- First telegram: 29 24 31 00 00 00 37 8E 02 D4 9A 10 C1 AE 40
- Second telegram: 29 40 45 B5 99 C2 9A 05 A2 7C

Transmitted Security Teach-in telegram for ERP2 when one telegram is used:

- Telegram: 29 14 31 00 00 00 37 8E 02 D4 9A 10 C1 AE 40 45 B5
  99 C2 9A 05 A2 7C

A.1.3 Data Message Processing in VAES Mode

PTM modules use different EEP when communicating in secure mode (EEP D2-03-00) compared to communicating in standard mode (EEP F6-02-01/02/03/04 depending on the selected rocker style).

EEP D2-03-00 is a VLD message using 1-byte of DATA. It encodes supported button actions in the least significant four bits of it (DATA[3:0]).

Security processing uses the data message parameters defined for this example:

- R-ORG-S: 30 (SEC)
- Plaintext data (D2-03-00): 0B (Button B1 is pressed)

Additionally, this assumes the security parameters defined for this example:

- Rolling code (RLC): 00 00 38
- Security Key (K): 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C

A.1.3.1 Data Encryption (VAES)

The encryption process for DATA is executed using the following steps:

```
VAES init vector    34 10 DE 8F 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
XOR
RLC padded          00 00 38 00 00 00 00 00 00 00 00 00 00 00 00 00
-----------------------------------------------------
Input               34 10 E6 8F 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
AES-128
Key                 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C
-----------------------------------------------------
AES128 encrypted    29 D6 00 24 78 01 C1 7D 08 D7 0E 73 1D DF 39 E1
XOR
DATA padded         0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
-----------------------------------------------------
DATA encrypted      22 D6 00 24 78 01 C1 7D 08 D7 0E 73 1D DF 39 E1
AND
Mask                FF
-------------------------------------------------
DATA encrypted      22
```

A.1.3.2 Data Authentication (CMAC)

A.1.3.2.1 Subkey Generation

```
Key                 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C
AES-128
Input               00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
-----------------------------------------------------
AES-128 Output      BC 1C D9 37 03 6D 3B E4 08 49 B9 01 67 D9 33 20


If MSb of the AES-128 Output is not set (does not apply in this example):
 Left shift by 1

Otherwise (applies in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<<1                 78 39 B2 6E 06 DA 77 C8 10 93 72 02 CF B2 66 40
-----------------------------------------------------
XOR                 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 87

K1=                 78 39 B2 6E 06 DA 77 C8 10 93 72 02 CF B2 66 C7

If MSb of K1 is not set (applies in this example):
 Left shift by 1

Otherwise (does not apply in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1
K2=                 F0 73 64 DC 0D B4 EF 90 21 26 E4 05 9F 64 CD 8E
```

A.1.3.2.2 CMAC Calculation

Key	8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C
Subkey K2	F0 73 64 DC 0D B4 EF 90 21 26 E4 05 9F 64 CD 8E
Input Data	R-ORG-S || ENCRYPTED DATA || RLC || PADDING
	30 22 00 00 38 80 00 00 00 00 00 00 00 00 00 00
XOR	
K2	F0 73 64 DC 0D B4 EF 90 21 26 E4 05 9F 64 CD 8E
---	---
Input	C0 51 64 DC 35 34 EF 90 21 26 E4 05 9F 64 CD 8E
AES128	
Key	8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C
---	---
CMAC (Use the 3 MSB)	BF 15 03 9F 7E 23 78 82 42 54 7A 65 8E 47 A0 5A

A.1.3.3 Resulting Telegrams

Message payload:

- R-ORG-S (ERP1): 30 (SEC)
- HEADER (ERP2): 27 (Broadcast telegram using R-ORG-S 0x30: SEC)
- Encrypted DATA: 22
- CMAC: BF 15 03
- RLC: 00 00 38

Transmitted telegram:

- ERP1: 30 22 00 00 38 BF 15 03
- ERP2: 27 22 00 00 38 BF 15 03

Decrypted message payload:

- R-ORG-S (ERP1): 32 (SEC_D)
- HEADER (ERP2): 2F (Broadcast telegram, R-ORG-S specified by EXT_TYPE)
- EXT_TYPE (ERP2): 04 (SEC_D)
- Decrypted DATA: 0B

Decrypted message:

- ERP1: 32 0B
- ERP2: 2F 04 0B

A.1.4 Data Message Processing in CCM Mode

Security processing uses the data message parameters defined for this example:

- Sender ID: FE D8 9C 22
- Destination ID: FF FF FF FF
- R-ORG-S: 30 (SEC)
- R-ORG: Not present
- Plaintext data (D2-03-00): 0B (Button B1 is pressed)

Additionally, this assumes the security parameters defined for this example:

- SLF: 31 (Mode = CCM, MAC Size = 4 byte, Transmitted RLC Size = 2 byte)
- Internal rolling code (RLC): 00 00 00 39
- Security Key (K): 8E 02 D4 9A 10 C1 AE 40 45 B5 99 C2 9A 05 A2 7C

A.1.4.1 CCM Parameters

Since PTM messages do not contain an original R-ORG, authenticated and encrypted data *m* consists only of the content of the DATA field (0x0B in this example).

We derive the CCM parameters from the parameters of the data message as follows:

- Nonce *N*: SENDER ID || DESTINATION ID ||RLC || 00
  FE D8 9C 22 FF FF FF FF 00 00 00 39 00
- Authenticated Data *a*: SENDER ID || DESTINATION ID ||RLC ||R-ORG-S
  FE D8 9C 22 FF FF FF FF 00 00 00 39 30
- Length of *a* = *l(a)*: 00 0D (13-byte)
- Authenticated and Encrypted Data *m*: R-ORG || DATA
  0B
- Length of *m* = *l(m)*: 00 01 (1-byte)

A.1.4.2 Block Calculation

We first determine the content of the B blocks that are used to generate the authentication signature as follows:

- B_0: B0 Flag || Nonce || *l(m)*
  `49 FE D8 9C 22 FF FF FF FF 00 00 00 39 00 00 01`
- B_1: *l(a)* || *a* || ZERO
  `00 0D FE D8 9C 22 FF FF FF FF 00 00 00 39 30 00`
- B_2: *m* || ZERO
  `0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00`

We then determine the content of the A blocks that are used to generate the encryption key stream blocks S as follows:

- A_0: A Flag || Nonce || 0x0000
  `01 FE D8 9C 22 FF FF FF FF 00 00 00 39 00 00 00`
- A_1: A Flag || Nonce || 0x0001
  `01 FE D8 9C 22 FF FF FF FF 00 00 00 39 00 00 01`

We then determine the content of the key stream blocks S as follows:

- S_0: AES-128(Key = K, Data = A_0)
  `F2 A3 A8 78 1D F2 81 DF F2 2F 5C 10 9F 9C 69 B5`
- S_1: AES-128(Key = K, Data = A_1)
  `23 34 87 56 E1 AC F4 E4 DD 09 2C 52 2E 59 6D 7A`

A.1.4.3 Authentication Signature Processing

We calculate the authentication signature *T* as follows:

- T_0 AES-128(Key = *K*, Data = B_0)
  `05 7D AD 56 FD 88 9A 38 EB 3E A4 62 CE A4 1B B6`
- X_0 XOR(T_0, B_1)
  `05 70 53 8E 61 AA 65 C7 14 C1 A4 62 CE 9D 2B B6`
- T_1 AES-128(Key = *K*, Data = X_0)
  `32 AF 5B 6D 7A 2B 95 5A CF 2E 1E 3C 9A 0D 84 24`
- X_1 XOR(T_1, B_2)
  `39 AF 5B 6D 7A 2B 95 5A CF 2E 1E 3C 9A 0D 84 24`
- $T_2 = T$ AES-128(Key = K, Data = X_1)
  `FC E6 AA D2 FC D8 4C 62 AB 10 C0 2B 7A 38 17 22`

We encrypt the authentication signature *T* using keystream block S_0 to obtain the encrypted signature *U*:

- *U* XOR(*T*, S_0)
  `0E 45 02 AA E1 2A CD BD 59 3F 9C 3B E5 A4 7E 97`

The encrypted signature *U* is then truncated to 4-byte by using the most significant four bytes (`0E4502AA`) which are then used as authentication signature.

A.1.4.4 Encryption Processing

We truncate S_1 to *l(m)* using the most significant byte (which is 0x23) and calculate the encrypted payload *e* from *m* (which is 0x0B) as follows:

- e XOR(*m*, S_1)
  `28`

A.1.4.5 Resulting Telegrams

Message payload:

- R-ORG-S (ERP1): 30 (SEC)
- HEADER (ERP2): 27 (Broadcast telegram using R-ORG-S 0x30: SEC)
- Encrypted DATA: 28
- RLC: 00 00 00 39 (only two LSB are transmitted: 00 39)
- MAC: 0E 45 02 AA

Transmitted telegram:

- ERP1: 30 28 00 39 0E 45 02 AA
- ERP2: 27 28 00 39 0E 45 02 AA

Decrypted message payload:

- R-ORG-S (ERP1): 32 (SEC_D)
- HEADER (ERP2): 2F (Broadcast telegram, R-ORG-S specified by EXT_TYPE)
- EXT_TYPE (ERP2): 04 (R-ORG-S 0x32: SEC_D)
- Decrypted DATA: 0B

Decrypted message:

- ERP1: 32 0B
- ERP2: 2F 04 0B

A.2 Secure 4BS Sensor Message

A.2.1 Example parameters

This example assumes the following message parameters:

- SENDER ID: 01 9E B6 3B
- DESTINATION ID: FF FF FF FF (Broadcast)
- R-ORG: A5 (4BS)
- Telegram payload: 08 27 FF 80

This example assumes the following security parameters:

- RLC: 00 C0 FF EE
- Security Key (K): 45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00
- VAES parameters: 3-byte Transmitted RLC / 3-byte CMAC (SLF: AB)
- CCM parameters: 2-byte Transmitted RLC / 4-byte CMAC (SLF: 31)

A.2.2 Security Teach-in (SEC_TI) Message

A.2.2.1 VAES Mode

The Security Teach-in message encodes the following parameters:

- R-ORG: 35 (SEC_TI)
- Teach-Info: 20 (First Part of 2) / 40 (Second Part of 2) / 10 (First Part of 1)
  Uni-directional, PSK Not Used, Not a PTM
- RLC: C0 FF EE
- SLF: AB (Mode = VAES, MAC Size = 3 byte, Transmitted RLC Size = 3 byte)
- Security Key: 45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00

Transmitted telegrams for ERP1:

- First telegram: 35 20 AB C0 FF EE 45 6E 4F 63 65 61 6E 20 47
- Second telegram: 35 40 6D 62 48 2E 31 33 00

Transmitted telegrams for ERP2 when two telegrams are used:

- First telegram: 29 20 AB C0 FF EE 45 6E 4F 63 65 61 6E 20 47
- Second telegram: 29 40 6D 62 48 2E 31 33 00

Transmitted telegram for ERP2 when one telegram is used:

- Telegram: 29 10 AB C0 FF EE 45 6E 4F 63 65 61 6E 20 47 6D 62
  48 2E 31 33 00

A.2.2.2 CCM Mode

The Security Teach-in message encodes the following parameters:

- R-ORG: 35 (SEC_TI)
- Teach-Info: 20 (First Part of 2) / 40 (Second Part of 2) / 10 (First Part of 1)
  Uni-directional, PSK Not Used, Not a PTM
- RLC (internal): 00 C0 FF EE
- SLF: 31 (Mode = CCM, MAC Size = 4 byte, Transmitted RLC Size = 2 byte)
- Security Key: 45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00

Transmitted telegrams for ERP1:

- First telegram: 35 20 31 00 C0 FF EE 45 6E 4F 63 65 61 6E 20
- Second telegram: 35 40 47 6D 62 48 2E 31 33 00

Transmitted telegrams for ERP2 when two telegrams are used:

- First telegram: 29 20 31 00 C0 FF EE 45 6E 4F 63 65 61 6E 20
- Second telegram: 29 40 47 6D 62 48 2E 31 33 00

Transmitted telegram for ERP2 when one telegram is used:

- Telegram: 29 10 31 00 C0 FF EE 45 6E 4F 63 65 61 6E 20 47 6D
  62 48 2E 31 33 00

A.2.3 Data Message Processing in VAES Mode

Security processing uses the data message parameters defined for this example:

- R-ORG-S: 31 (SEC_R)
- R-ORG: A5 (4BS)
- DATA: 08 27 FF 80

Additionally, this assumes the security parameters defined for this example:

- SLF: AB (Mode = VAES, MAC SIZE = 3 byte, Transmitted RLC Size = 3 byte)
- Rolling code: C0 FF EE
- Security Key (K): 45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00

A.2.3.1 Data Encryption (VAES)

```
VAES init vector       34 10 DE 8F 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
XOR
RLC padded             C0 FF EE 00 00 00 00 00 00 00 00 00 00 00 00 00
-------------------------------------------------------
Input                  F4 EF 30 8F 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
AES128
Key                    45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00
-------------------------------------------------------
AES128 encrypted       9B E2 E3 5D 5F E7 85 86 45 20 05 87 DD 3B 51 5C
Truncated to 5-byte    9B E2 E3 5D 5F
XOR
R-ORG||DATA            A5 08 27 FF 80
-------------------------------------------------------
ENCRYPTED R-ORG||DATA  3E EA C4 A2 DF
```

A.2.3.2 Data Authentication (CMAC)

A.2.3.2.1 Subkey Generation

Key 45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00

AES128

Input 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

AES128 encrypted DD 21 AA 89 2D DF 3C B9 67 C3 14 36 9A 27 23 38

If MSb of the AES-128 Output is not set (does not apply in this example):
 Left shift by 1

Otherwise (applies in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1

Result after shift BA 43 55 12 5B BE 79 72 CF 86 28 6D 34 4E 46 70

XOR

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 87

K1= BA 43 55 12 5B BE 79 72 CF 86 28 6D 34 4E 46 F7

If MSb of K1 is not set (does not apply in this example):
 Left shift by 1

Otherwise (applies in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1

Result after shift 74 86 AA 24 B7 7C F2 E5 9F 0C 50 DA 68 9C 8D EE

XOR

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 87

K2= 74 86 AA 24 B7 7C F2 E5 9F 0C 50 DA 68 9C 8D 69

A.2.3.2.2 CMAC Calculation

Input Data	R-ORG-S || ENCRYPTED DATA || RLC
	31 3E EA C4 A2 DF C0 FF EE
Input Data || PADDING	31 3E EA C4 A2 DF C0 FF EE 80 00 00 00 00 00 00
XOR	
K2	74 86 AA 24 B7 7C F2 E5 9F 0C 50 DA 68 9C 8D 69
---	---
Input	45 B8 40 E0 15 A3 32 1A 71 8C 50 DA 68 9C 8D 69
AES128	
Key	45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00
---	---
CMAC (Use 3 MSB)	EA F2 0E ED 28 67 9F 64 1C 15 B1 0B 93 08 D0 D4

A.2.3.3 Resulting Telegrams

Data message payload

- R-ORG-S: 31 (SEC_R: Secure telegram with encrypted original R-ORG)
- HEADER (ERP2): 28 (Broadcast telegram, R-ORG-S 0x31: SEC_R)
- Encrypted R-ORG + DATA: 3E EA C4 A2 DF
- RLC: C0 FF EE
- CMAC: EA F2 0E

Transmitted telegram:

- ERP1: 31 3E EA C4 A2 DF C0 FF EE EA F2 0E
- ERP2: 28 3E EA C4 A2 DF C0 FF EE EA F2 0E

Decrypted data message payload:

- R-ORG (ERP1): A5 (4BS)
- HEADER (ERP2): 22 (Broadcast telegram, 4BS)
- Decrypted DATA: 08 27 FF 80

Decrypted data message:

- ERP1: A5 08 27 FF 80
- ERP2: 22 08 27 FF 80

A.2.4 Data Message Processing in CCM Mode

Security processing uses the data message parameters defined for this example:

- Sender ID: 01 9E B6 3B
- Destination ID: FF FF FF FF
- R-ORG-S: 31 (SEC_R)
- R-ORG: A5 (4BS)
- Plaintext data: 08 27 FF 80

Additionally, this assumes the security parameters defined for this example:

- SLF: 31 (Mode = CCM, MAC Size = 4 byte, Transmitted RLC Size = 2 byte)
- Internal rolling code (RLC): 00 C0 FF EE
- Security Key (K): 45 6E 4F 63 65 61 6E 20 47 6D 62 48 2E 31 33 00

A.2.4.1 CCM Parameters

We derive the CCM parameters from the parameters of the data message as follows:

- Nonce *N*: SENDER ID || DESTINATION ID ||RLC || 00
  01 9E B6 3B FF FF FF FF 00 C0 FF EE 00
- Authenticated Data *a*: SENDER ID || DESTINATION ID ||RLC ||R-ORG-S
  01 9E B6 3B FF FF FF FF 00 C0 FF EE 31
- Length of *a* = *l(a)*: 00 0D (13-byte)
- Authenticated and Encrypted Data *m*: R-ORG || DATA
  A5 08 27 FF 80
- Length of *m* = *l(m)*: 00 05 (5-byte)

A.2.4.2 Block Calculation

We first determine the content of the B blocks that are used to generate the authentication signature as follows:

- B_0: B0 Flag || Nonce || *l(m)*
  `49 01 9E B6 3B FF FF FF FF 00 C0 FF EE 00 00 05`
- B_1: *l(a)* || *a* || ZERO
  `00 0D 01 9E B6 3B FF FF FF FF 00 C0 FF EE 31 00`
- B_2: *m* || ZERO
  `A5 08 27 FF 80 00 00 00 00 00 00 00 00 00 00 00`

We then determine the content of the A blocks that are used to generate the encryption key stream blocks S as follows:

- A_0: A Flag || Nonce || 0x0000
  `01 01 9E B6 3B FF FF FF FF 00 C0 FF EE 00 00 00`
- A_1: A Flag || Nonce ||0x0001
  `01 01 9E B6 3B FF FF FF FF 00 C0 FF EE 00 00 01`

We then determine the content of the key stream blocks S as follows:

- S_0: AES-128(Key = K, Data = A_0)
  `E8 EA B6 43 C2 8B 58 7D EA E5 AB 10 BC 71 14 6E`
- S_1: AES-128(Key = K, Data = A_1)
  `33 C4 06 20 6E A8 A4 53 86 BD A5 B5 78 1B E7 D0`

A.2.4.3 Authentication Signature Processing

We calculate the authentication signature *T* as follows:

- T_0 AES-128(Key = *K*, Data = B_0)
  E3 CA B0 BE 0A 36 FB 99 33 89 32 BA BF 1D E5 05
- X_0 XOR(T_0, B_1)
  E3 C7 B1 20 BC 0D 04 66 CC 76 32 7A 40 F3 D4 05
- T_1 AES-128(Key = *K*, Data = X_0)
  17 CC 96 CB 5B 74 AA 8E 53 7D 54 C3 7F 26 0A 75
- X_1 XOR(T_1, B_2)
  B2 C4 B1 34 DB 74 AA 8E 53 7D 54 C3 7F 26 0A 75
- $T_2 = T$ AES-128(Key = K, Data = X_1)
  49 27 7E E6 BD 55 3B FA 94 21 D7 12 BC A0 EE E1

We encrypt the authentication signature *T* using keystream block S_0 to obtain the encrypted signature *U*:

- *U* XOR(*T*, S_0)
  A1 CD C8 A5 7F DE 63 87 7E C4 7C 02 00 D1 FA 8F

The encrypted signature *U* is then truncated to 4-byte by using the most significant four bytes (A1CDC8A5) which are then used as authentication signature.

A.2.4.4 Encryption Processing

We truncate S1 to l(m) using the most significant byte and calculate the encrypted payload e as follows:

- e XOR(*m*, S_1)
  96 CC 21 DF EE

A.2.4.5 Resulting Telegrams

Data message payload:

- R-ORG-S (ERP1): 31 (SEC_R)
- HEADER (ERP2): 28 (Broadcast telegram using R-ORG-S 0x31: SEC_R)
- Encrypted R-ORG and DATA: 96 CC 21 DF EE
- RLC: 00 C0 FF EE (only two LSB are transmitted: FF EE)
- MAC: A1 CD C8 A5

Transmitted telegram:

- ERP1: 31 96 CC 21 DF EE FF EE A1 CD C8 A5
- ERP2: 28 96 CC 21 DF EE FF EE A1 CD C8 A5

Decrypted message payload:

- R-ORG (ERP1): A5 (4BS)
- HEADER (ERP2): 22 (Broadcast telegram, 4BS)
- Decrypted DATA: 08 27 FF 80

Decrypted message:

- ERP1: A5 08 27 FF 80
- ERP2: 22 08 27 FF 80

A.3 Secure Chained VLD Data Message (SEC_CDM) with UTE

This example illustrates how an STM 550 sensor module using EEP D2-14-41 with the parameters shown below would establish secure communication using VAES and CCM security modes, transmit a UTE message for EEP teach-in followed by a data message containing sensor data.

A.3.1 Example parameters

This example assumes the following message parameters:

- SENDER ID: 04 2C 6A BC
- DESTINATION ID: FF FF FF FF (Broadcast)
- Full EEP: D2-14-41
- R-ORG: D2 (VLD)
- DATA: A0 DA C0 18 6B EE F2 26 E0

Additionally, this example assumes the following security parameters:

- VAES parameters: 4-byte Transmitted RLC / 4-byte CMAC (SLF: F3)
- CCM parameters: 4-byte Transmitted RLC / 4-byte CMAC (SLF: F1)
- RLC: 00 00 F3 91
- Security Key (K): 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF

A.3.2 Security Teach-in Message

A.3.2.1 VAES Mode

The Security Teach-in message encodes the following parameters in VAES mode:

- R-ORG: 35 (SEC_TI)
- Teach-Info: 20 (First Part of 2) / 40 (Second Part)
  Uni-directional, PSK Not Used, Is Not a PTM
- RLC: 00 00 F3 91
- SLF: F3 (Mode = VAES, MAC Size = 4 byte, Transmitted RLC Size = 4 byte)
- Security Key: 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF

Transmitted telegrams for ERP1:

- First telegram: 35 20 F3 00 00 F3 91 92 6E 21 44 9E 72 A2 67
- Second telegram: 35 40 3D 85 E0 97 EF 8A 50 CF

Transmitted telegrams for ERP2 when two telegrams are used:

- First telegram: 29 20 F3 00 00 F3 91 92 6E 21 44 9E 72 A2 67
- Second telegram: 29 40 3D 85 E0 97 EF 8A 50 CF

Transmitted telegram for ERP2 when one telegram is used:

- Telegram: 29 10 F3 00 00 F3 91 92 6E 21 44 9E 72 A2 67 3D 85
  E0 97 EF 8A 50 CF

A.3.2.2 CCM Mode

The Security Teach-in message encodes the following parameters in CCM mode:

- R-ORG: 35 (SEC_TI)
- Teach-Info: 20 (First Part of 2) / 40 (Second Part)
  Uni-directional, PSK Not Used, Not a PTM
- SLF: F1 (Mode = CCM, MAC SIZE = 4 byte, Transmitted RLC SIZE = 4 byte)
- RLC (internal): 00 00 F3 91
- Security Key: 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF

Transmitted telegrams for ERP1:

- First telegram: 35 20 F1 00 00 F3 91 92 6E 21 44 9E 72 A2 67
- Second telegram: 35 40 3D 85 E0 97 EF 8A 50 CF

Transmitted telegrams for ERP2 when two telegrams are used:

- First telegram: 29 20 F1 00 00 F3 91 92 6E 21 44 9E 72 A2 67
- Second telegram: 29 40 3D 85 E0 97 EF 8A 50 CF

Transmitted telegram for ERP2 when one telegram is used:

- Telegram: 29 10 F1 00 00 F3 91 92 6E 21 44 9E 72 A2 67 3D 85
  E0 97 EF 8A 50 CF

A.3.3 UTE (EEP Teach-in) Message

UTE (Universal Teach-in of EnOcean Equipment Profiles) is a message type that is used by the transmitter to inform the receiver about which EEP is used by the transmitter.

UTE messages are identified by R-ORG = 0xD4: UTE. For ERP2 telegrams, EEP = 0xD4: UTE is encoded by setting HEADER[3:0] to 0x5.

UTE messages are sent over the encrypted radio channel, i.e. after secure communication has been established by means of a Security Teach-in message or by other means as described in Chapter 5. Figure 58 below shows the format of UTE messages.

R-ORG	DATA						
UTE	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[5]	DATA[6]
0xD4	CONTROL	CHANNEL	MAN_ID_LSB	MAN_ID_MSB	VARIANT	FUNCTION	R-ORG

Figure 58. Format of UTE messages

A.3.3.1 UTE Message Content

For this example, we assume an UTE message which encodes the following parameters:

- R-ORG: D4 (UTE)
- CONTROL: 40 (Uni-directional Teach-in Request)
- CHANNEL: FF (Any Channel)
- MAN_ID_LSB: 0B (LSB of 000B = EnOcean GmbH)
- MAN_ID_MSB: 00 (MSB of 000B = EnOcean GmbH)
- VARIANT (TYPE): 41 (Temperature, Humidity, XYZ Acceleration, Illumination Sensor)
- FUNCTION: 14 (Multi-function Sensor)
- R-ORG: D2 (VLD)

The input data to the security processing would therefore be as follows:

- RLC: `00 00 F3 91`
- KEY: `92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF`
- R-ORG-S: `31` (SEC_R)
- R-ORG: `D4` (UTE)
- DATA: CONTROL ||CHANNEL || MANUFACTURER_ID || VARIANT || FUNCTION || R-ORG
  `40 FF 0B 00 41 14 D2`

Security processing will encrypt and authenticate this input data using the security settings (SLF) defined for the transmission of telegrams.

Note that transmission of the resulting secure telegram will require segmentation (chaining) across two telegrams for most ERP implementations since the combined size of original R-ORG, DATA, RLC and CMAC (which is 16 byte) exceeds 14-byte.

A.3.3.2 UTE Message Processing in VAES Mode

A.3.3.2.1 Data Encryption (VAES)

```
VAES init vector          34 10 DE 8F 1A BA 3E FF 9F 5ª 11 71 72 EA CA BD
XOR
RLC padded                00 00 F3 91 00 00 00 00 00 00 00 00 00 00 00 00
--------------------------------------------------------
Input                     34 10 2D 1E 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
AES128
Key                       92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
--------------------------------------------------------
AES128 encrypted          B0 F8 C2 27 E7 D9 7A 2F 58 7D 1D CF BB 53 5F 57
Truncated to 8-byte       B0 F8 C2 27 E7 D9 7A 2F
XOR
R-ORG || DATA             D4 40 FF 0B 00 41 14 D2
--------------------------------------------------------
ENCRYPTED R-ORG || DATA   64 B8 3D 2C E7 98 6E FD
```

A.3.3.2.2 Data Authentication (CMAC)

Subkey Generation

Key 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF

AES128

Input 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

AES128 encrypted 43 9C 63 4B D5 F1 42 FB 0B E8 32 39 0A 58 68 AB

If MSb of the AES-128 output is not set (applies in this example):
 Left shift by 1

Otherwise (does not apply in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1

K1= 87 38 C6 97 AB E2 85 F6 17 D0 64 72 14 B0 D1 56

If MSb of K1 is not set (does not apply in this example):
 Left shift by 1

Otherwise (applies in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1

Result after shift 0E 71 8D 2F 57 C5 0B EC 2F A0 C8 E4 29 61 A2 AC

XOR

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 87

K2= 0E 71 8D 2F 57 C5 0B EC 2F A0 C8 E4 29 61 A2 2B

CMAC Calculation

Input Data	R-ORG-S || ENCRYPTED DATA || RLC
	31 64 B8 3D 2C E7 98 6E FD 00 00 F3 91
Input Data || PADDING	
	31 64 B8 3D 2C E7 98 6E FD 00 00 F3 91 80 00 00
XOR	
K2	**0E 71 8D 2F 57 C5 0B EC 2F A0 C8 E4 29 61 A2 2B**
---	---
Input	3F 15 35 12 7B 22 93 82 D2 A0 C8 17 B8 E1 A2 2B
AES128	
Key	92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
---	---
CMAC (Use 4 MSB)	E1 90 62 31 0E 9F 0F 50 54 EB E3 BE E2 2A EA F9

A.3.3.2.3 Resulting Telegrams

The resulting data message payload is 16-byte as shown below.

- R-ORG-S: 31 (SEC_R: Secure telegram with encrypted original R-ORG)
- Encrypted R-ORG + DATA: 64 B8 3D 2C E7 98 6E FD
- RLC: 00 00 F3 91
- CMAC: E1 90 62 31

This message payload does not fit into one ERP1 or ERP2 telegram (except for regions where ERP2 telegrams with more than 14-byte of payload are used). The payload therefore must be segmented (chained) into two secure chained telegrams as described in Chapter 6.4.

For this example, we assuming a CHAIN_ID of 0b10 (CHAIN #2) which would result in the following parameters for the chaining process:

- R-ORG-S: 33 (SEC_CDM, this will replace SEC_R during transmission)
- ERP2 HEADER and EXT_TYPE: 2F 33 (Broadcast, Using EXT_TYPE, EXT_TYPE: SEC_CDM)
- CHAIN_CTRL 80 (First Segment) / 81 (Second Segment)
- CHAIN_LENGTH: 00 10 (16 byte)

The resulting secure chained telegrams would be as follows:

- ERP1 (2 chained telegrams): 33 80 00 10 64 B8 3D 2C E7 98 6E FD 00 00 F3
  33 81 91 E1 90 62 31

- ERP2 (2 chained telegrams): 2F 33 80 00 10 64 B8 3D 2C E7 98 6E FD 00 00 F3
  2F 33 81 91 E1 90 62 31

- ERP2 (1 telegram): 28 64 B8 3D 2C E7 98 6E FD 00 00 F3 91 E1 90 62
  31

Decrypted message:

- R-ORG (ERP1): D4 (UTE)
- HEADER (ERP2): 25 (Broadcast telegram, UTE)
- Decrypted DATA: 40 FF 0B 00 41 14 D2

Decrypted message:

- ERP1: D4 40 FF 0B 00 41 14 D2
- ERP2: 25 40 FF 0B 00 41 14 D2

A.3.3.3 UTE Message Processing in CCM Mode

We assume the following parameters for the UTE message:

- RLC: 00 00 F3 91
- KEY: 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
- R-ORG-S: 31 (SEC_R)
- R-ORG: D4 (UTE)
- DATA: CONTROL ||CHANNEL || MANUFACTURER_ID || VARIANT || FUNCTION || R-ORG
  40 FF 0B 00 41 14 D2

A.3.3.3.1 CCM Parameters

We assume similar parameters for CCM mode as for VAES mode, keeping in mind that the internal RLC size is always 4-byte for CCM mode:

- Sender ID: 04 2C 6A BC
- Destination ID: FF FF FF FF
- RLC: 00 00 F3 91
- Security Key (K): 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
- Nonce *N*: SENDER ID || DESTINATION ID ||RLC || 00
  04 2C 6A BC FF FF FF FF 00 00 F3 91 00
- Authenticated Data *a*: SENDER ID || DESTINATION ID ||RLC ||R-ORG-S
  04 2C 6A BC FF FF FF FF 00 00 F3 91 31
- Length of *a* = *l(a)*: 00 0D (13-byte)
- Plaintext Data *m*: R-ORG || DATA
  D4 40 FF 0B 00 41 14 D2
- Length of *m* = *l(m)*: 00 08 (8-byte)

A.3.3.3.2 Block Calculation

We first determine the content of the B blocks that are used to generate the authentication signature as follows:

- B_0: B0 Flag || Nonce || *l(m)*
  49042C6ABCFFFFFFFF0000F391000008
- B_1: *l(a)* || *a* || ZERO
  000D042C6ABCFFFFFFFF0000F3913100
- B_2: *m* || ZERO
  D440FF0B004114D20000000000000000

We then determine the content of the A blocks that are used to generate the encryption key stream blocks S as follows:

- A_0: A Flag || Nonce || 0x0000
  01042C6ABCFFFFFFFF0000F391000000
- A_1: A Flag || Nonce ||0x0001
  01042C6ABCFFFFFFFF0000F391000001

We then determine the content of the key stream blocks S as follows:

- S_0: AES-128(Key = K, Data = A_0)
  67D41545319C8CF5F4F5746F3FBFFEAA
- S_1: AES-128(Key = K, Data = A_1)
  117BEBF4450D03F6E92FD7FB6DF40504

A.3.3.3.3 Authentication Signature Processing

We calculate the authentication signature *T* as follows:

- T_0 AES-128(Key = *K*, Data = B_0)
  8201042217C7AF0F8B320E07F61DE951
- X_0 XOR(T_0, B_1)
  820C000E7D7B50F074CD0E07058CD851
- T_1 AES-128(Key = *K*, Data = X_0)
  25C60431FB02176735DDE5CA04E9C56D

- X_1 XOR(T_1, B_2)
  F186FB3AFB4303B535DDE5CA04E9C56D

- $T_2 = T$ AES-128(Key = K, Data = X_1)
  18FADDBFEC90AEA0AF95B6031B7626C7

We encrypt the authentication signature T using keystream block S_0 to obtain the encrypted signature U:

- U XOR(T, S_0)
  7F2EC8FADD0C22555B60C26C24C9D86D

The encrypted signature U is then truncated to 4-byte by using the most significant four bytes (7F2EC8FA) which are then used as authentication signature.

A.3.3.3.4 Encryption Processing

We calculate the encrypted payload e by truncating S_1 to $l(m)$ using the most significant bytes of S_1 for the following operation:

- e XOR(m, S_1)
  C53B14FF454C1724

A.3.3.3.5 Resulting Telegrams

Data telegram payload:

- R-ORG-S (ERP1): 31 (SEC_R)
- HEADER (ERP2): 28 (Broadcast telegram using R-ORG-S 0x31: SEC_R)
- Encrypted R-ORG and DATA: C5 3B 14 FF 45 4C 17 24
- RLC: 00 00 F3 91
- CMAC: 7F 2E C8 FA

Transmitted secure data telegram:

- ERP1: 31 C5 3B 14 FF 45 4C 17 24 00 00 F3 91 7F 2E C8 FA
- ERP2: 28 C5 3B 14 FF 45 4C 17 24 00 00 F3 91 7F 2E C8 FA

Decrypted data telegram payload:

- R-ORG (ERP1): D4
- HEADER (ERP2): 25
- Decrypted DATA: 40 FF 0B 00 41 14 D2

Decrypted Data Telegram

- ERP1: D4 40 FF 0B 00 41 14 D2
- ERP2: 25 40 FF 0B 00 41 14 D2

A.3.4 VLD Data Message

VLD (Variable Length Data telegram) is a telegram type that is used by complex sensors which often require larger data payload sizes.

VLD telegrams are identified by R-ORG = 0xD2: VLD. For ERP2 telegrams, EEP = 0xD2: VLD is encoded by setting HEADER[3:0] to 0x4.

For this example, we assume the VLD message to contain the following parameters:

- Sender ID: 01 9E B6 3B
- Destination ID: FF FF FF FF
- R-ORG: D2 (VLD)
- DATA: A0 DA C0 18 6B EE F2 26 E0

We assume the security parameters to be as follows:

- RLC: 00 00 F3 92
- KEY: 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
- R-ORG-S: 31 (SEC_R) / 33 (SEC_CDM, since chaining is required)

Security processing will encrypt and authenticate this input data using the security settings (SLF) defined for the transmission of telegrams.

Note that transmission of the resulting secure message will require segmentation (chaining) across two telegrams for most ERP implementations since the combined size of original R-ORG, DATA, RLC and CMAC (which is 16 byte) exceeds 14 byte.

A.3.4.1 Data Message Processing in VAES Mode

Security processing uses the data message parameters defined for this example:

- R-ORG: D2 (VLD)
- DATA: A0 DA C0 18 6B EE F2 26 E0

Additionally, this assumes the security parameters defined for this example:

- SLF: F3 (Mode = VAES, MAC Size = 4 byte, Transmitted RLC Size = 4 byte)
- RLC: 00 00 F3 92
- KEY: 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
- R-ORG-S: 31 (SEC_R) / 33 (SEC_CDM)

A.3.4.1.1 Data Encryption (VAES)

```
VAES init vector          34 10 DE 8F 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
XOR
RLC padded                00 00 F3 92 00 00 00 00 00 00 00 00 00 00 00 00
-----------------------------------------------------
Input                     34 10 2D 1D 1A BA 3E FF 9F 5A 11 71 72 EA CA BD
AES128
Key                       92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
-----------------------------------------------------
AES128 encrypted          A2 59 C7 CA F4 3F 5C 76 FF 83 90 E8 79 A1 A1 EA
Truncated to 10-byte      A2 59 C7 CA F4 3F 5C 76 FF 83
XOR
R-ORG || DATA             D2 A0 DA C0 18 6B EE F2 26 E0
-----------------------------------------------------
ENCRYPTED R-ORG || DATA   70 F9 1D 0A EC 54 B2 84 D9 63
```

A.3.4.1.2 Data Authentication (CMAC)

Subkey Generation

Key 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF

AES128

Input 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

AES128 encrypted 43 9C 63 4B D5 F1 42 FB 0B E8 32 39 0A 58 68 AB

If MSb of the AES-128 output is not set (applies in this example):
 Left shift by 1

Otherwise (does not apply in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1

K1= 87 38 C6 97 AB E2 85 F6 17 D0 64 72 14 B0 D1 56

If MSb of K1 is not set (does not apply in this example):
 Left shift by 1

Otherwise (applies in this example):
 Left shift by 1,
 Trim to 16 byte,
 XOR 00000000000000000000000000000087

<< 1

Result after shift 0E 71 8D 2F 57 C5 0B EC 2F A0 C8 E4 29 61 A2 AC

XOR

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 87

K2= 0E 71 8D 2F 57 C5 0B EC 2F A0 C8 E4 29 61 A2 2B

CMAC Calculation

Input Data	R-ORG-S || ENCRYPTED DATA || RLC
	31 70 F9 1D 0A EC 54 B2 84 D9 63 00 00 F3 92
Input Data || PADDING	
	31 70 F9 1D 0A EC 54 B2 84 D9 63 00 00 F3 92 80
XOR	
K2	**0E 71 8D 2F 57 C5 0B EC 2F A0 C8 E4 29 61 A2 2B**
---	---
Input	3F 01 74 32 5D 29 5F 5E AB 79 AB E4 29 92 30 AB
AES128	
Key	92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
---	---
CMAC (Use 4 MSB)	04 A9 D4 C3 0F 6A A3 8F 9B 49 FB 0A 41 0F 4F 80

A.3.4.1.3 Telegram Chaining

The resulting data telegram payload is 19-byte as shown below.

R-ORG-S: 33 (SEC_CDM: Secure chained message with encrypted R-ORG)

- Encrypted R-ORG + DATA: 70 F9 1D 0A EC 54 B2 84 D9 63
- RLC: 00 00 F3 92
- CMAC: 04 A9 D4 C3

This payload does not fit into one ERP1 or ERP2 telegram (except for regions where ERP2 telegrams with more than 14-byte of payload are used). The payload therefore must be segmented (chained) into two secure chained telegrams as described in Chapter 6.4.

For this example, we assuming a CHAIN_ID of 0b01 (CHAIN #1) which would result in the following parameters for the chaining process:

- R-ORG-S: 33 (SEC_CDM, this will replace SEC_R if chaining is required)
- ERP2 HEADER and EXT_TYPE: 2F 33 (Broadcast, R-ORG in EXT_TYPE, EXT_TYPE = SEC_CDM)
  28 (SEC_R, if no chaining is required)
- CHAIN_CTRL 40 (First Segment) / 41 (Second Segment)
- CHAIN_LENGTH: 00 12 (18 byte)

The resulting secure chained telegrams would be as follows:

- ERP1: 33 40 00 12 70 F9 1D 0A EC 54 B2 84 D9 63 00
  (2 chained telegrams) 33 41 00 F3 92 04 A9 D4 C3

  ERP2: 2F 33 40 00 12 70 F9 1D 0A EC 54 B2 84 D9 63 00
  (2 chained telegrams) 2F 33 41 00 F3 92 04 A9 D4 C3

- ERP2 (1 telegram): 28 70 F9 1D 0A EC 54 B2 84 D9 63 00 00 F3 92 E1 90
  62 31

Decrypted payload:

- R-ORG (ERP1): D2 (VLD)
- HEADER (ERP2): 24 (Broadcast telegram, VLD)
- Decrypted DATA: A0 DA C0 18 6B EE F2 26 E0

Decrypted data message:

- ERP1: D2 A0 DA C0 18 6B EE F2 26 E0
- ERP2: 24 A0 DA C0 18 6B EE F2 26 E0

A.3.4.2 Data Message Processing in CCM Mode

Security processing uses the parameters defined for this example:

- Sender ID: 01 9E B6 3B
- Destination ID: FF FF FF FF
- R-ORG: D2 (VLD)
- DATA: A0 DA C0 18 6B EE F2 26 E0

Additionally, this assumes the security parameters defined for this example:

- SLF: F1 (Mode = CCM, MAC SIZE = 4 byte, Transmitted RLC SIZE = 4 byte)
- RLC: 00 00 F3 92
- KEY: 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF
- R-ORG-S: 31 (SEC_R) / 33 (SEC_CDM)

A.3.4.2.1 CCM Parameters

We assume similar parameters for CCM mode as for VAES mode, keeping in mind that the internal RLC size is always 4-byte for CCM mode:

- RLC: 00 00 F3 92

- Nonce *N*: SENDER ID || DESTINATION ID ||RLC || 00
  01 9E B6 3B FF FF FF FF 00 00 F3 92 00

- Security Key (K): 92 6E 21 44 9E 72 A2 67 3D 85 E0 97 EF 8A 50 CF

- Authenticated Data *a*: SENDER ID || DESTINATION ID ||RLC ||R-ORG-S
  01 9E B6 3B FF FF FF FF 00 00 F3 92 31

- Length of *a* = *l(a)*: 00 0D (13-byte)

- Authenticated and Encrypted Data *m*: R-ORG || DATA
  D2 A0 DA C0 18 6B EE F2 26 E0

- Length of *m* = *l(m)*: 00 0A (10-byte)

A.3.4.2.2 Block Calculation

We first determine the content of the B blocks that are used to generate the authentication signature as follows:

- B_0: B0 Flag || Nonce || *l(m)*
  `49 01 9E B6 3B FF FF FF FF 00 00 F3 92 00 00 0A`
- B_1: *l(a)* || *a* || ZERO
  `00 0D 01 9E B6 3B FF FF FF FF 00 00 F3 92 31 00`
- B_2: *m* || ZERO
  `D2 A0 DA C0 18 6B EE F2 26 E0 00 00 00 00 00 00`

We then determine the content of the A blocks that are used to generate the encryption key stream blocks S as follows:

- A_0: A Flag || Nonce || 0x0000
  `01 01 9E B6 3B FF FF FF FF 00 00 F3 92 00 00 00`
- A_1: A Flag || Nonce || 0x0001
  `01 01 9E B6 3B FF FF FF FF 00 00 F3 92 00 00 01`

We then determine the content of the key stream blocks S as follows:

- S_0: AES-128(Key = K, Data = A_0)
  `0E E6 2B AB 21 3B 58 CE 00 31 DC 5E 2B 0A E9 BE`
- S_1: AES-128(Key = K, Data = A_1)
  `B5 89 9C FB F9 D5 68 E0 FE E2 B6 E0 55 95 E4 06`

A.3.4.2.3 Authentication Signature Processing

We calculate the authentication signature *T* as follows:

- T_0 AES-128(Key = *K*, Data = B_0)
  `03 3C C0 B9 65 DD 49 F2 EC A1 DF B9 51 F0 32 8B`
- X_0 XOR(T_0, B_1)
  `03 31 C1 27 D3 E6 B6 0D 13 5E DF B9 A2 62 03 8B`
- T_1 AES-128(Key = *K*, Data = X_0)
  `E4 6A 02 DA 3E E9 ED B5 56 CF E4 17 11 B5 CB E0`
- X_1 XOR(T_1, B_2)
  `36 CA D8 1A 26 82 03 47 70 2F E4 17 11 B5 CB E0`
- $T_2 = T$ AES-128(Key = K, Data = X_1)
  `C7 2E C5 17 06 CD 20 CF 1F 42 96 31 62 9B D3 F4`

We encrypt the authentication signature *T* using keystream block S_0 to obtain the encrypted signature *U*:

- *U* XOR(*T*, S_0)
  `C9 C8 EE BC 27 F6 78 01 1F 73 4A 6F 49 91 3A 4A`

The encrypted signature *U* is then truncated to 4-byte by using the most significant four bytes (C9 C8 EE BC) which are then used as authentication signature.

A.3.4.2.4 Encryption Processing

We calculate the encrypted payload *e* by truncating S_1 to *l(m)* using the most significant bytes of S_1 for the following operation:

- e XOR(*m*, S_1)
  `67 29 46 3B E1 BE 86 12 D8 02`

A.3.4.2.5 Resulting Telegrams

Data telegram payload:

- R-ORG-S: 33 (SEC_CDM, this will replace SEC_R if chaining is required)
- ERP2 HEADER and EXT_TYPE: 2F 33 (Broadcast, R-ORG in EXT_TYPE, EXT_TYPE = SEC_CDM)
  28 (SEC_R, if no chaining is required)
- CHAIN_CTRL 40 (First Segment) / 41 (Second Segment)
- CHAIN_LENGTH: 00 12 (18 byte)

The resulting secure chained telegrams would be as follows:

- ERP1: 33 40 00 12 67 29 46 3B E1 BE 86 12 D8 02 00
  (2 chained telegrams) 33 41 00 F3 92 C9 C8 EE BC

- ERP2: 2F 33 40 00 12 67 29 46 3B E1 BE 86 12 D8 02 00
  (2 chained telegrams) 2F 33 41 00 F3 92 C9 C8 EE BC

- ERP2 (1 telegram): 28 67 29 46 3B E1 BE 86 12 D8 02 00 00 F3 92 C9
  C8 EE BC

Decrypted payload:

- R-ORG (ERP1): D2 (VLD)
- HEADER (ERP2): 24 (Broadcast telegram, VLD)
- Decrypted DATA: A0 DA C0 18 6B EE F2 26 E0

Decrypted data telegram

- ERP1: D2 A0 DA C0 18 6B EE F2 26 E0
- ERP2: 24 A0 DA C0 18 6B EE F2 26 E0

Annex B - Key Derivation Example

The steps below illustrate how to derive a secondary key *K1* from a primary key *K*, a context *C* and an Initialization Value *I* (which is a constant for key derivation within the scope of ERP telegrams).

B.1 Input Parameters

This example assumes the following parameters:

- Primary Key *K* = 000102030405060708090A0B0C0D0E0F
- Context C = 00FF801234
- Initialization Value I = 656E6F6365616E01 (“enocean” || 0x01)

B.2 Key Derivation Process

Note that all CMAC operations are written using the syntax *CMAC(KEY, Input)*.

Salt *S* = CMAC(*ZERO*, *C*)

S = CMAC(00000000000000000000000000000000, 00FF801234)

S = 34824D2C8ABA959976DF2A81F0769D7C

Temporary Key *T* = CMAC(*S*, *K*)

T=CMAC(34824D2C8ABA959976DF2A81F0769D7C, 000102030405060708090A0B0C0D0E0F)

T = 10ECC0BD44804B5DD6B8014FDF567584

Derived Key K1 = CMAC(*T*, *I*)

K1 = CMAC(10ECC0BD44804B5DD6B8014FDF567584, 656E6F6365616E01)

K1 = AA661C353DED399158C0F04558C31450

Annex C - Legacy SLF Definition

Table 16 below provides a summary of the legacy SLF structure with the corresponding field encodings for information-only purposes.

Note that only a subset of possible combinations has previously been used and that support for specific combinations may not be assumed. New designs shall use SLF as defined in Chapter 4.7.

7	6	5	4	3	2	1	0
RLC_TYPE			**CMAC_TYPE**		**ENCRYPTION_TYPE**		
0b000: RLC not used 0b001: Reserved for future use 0b010: 16-bit internal, 0-bit transmitted [Implicit] Roll Over/ Window Algorithm 0b011: 16 bit internal, 16-bit transmitted Roll Over/ Window Algorithm 0b100: 24-bit internal, 0-bit transmitted [Implicit] Roll Over/ Window Algorithm 0b101: 24-bit internal, 24-bit transmitted No Roll Over / No Window Algorithm 0b110: 32-bit internal, 24-bit transmitted No Roll Over / No Window Algorithm 0b111: 32-bit internal, 32-bit transmitted No Roll Over / No Window Algorithm			0b00: CMAC not used 0b01: CMAC (3-byte transmitted) 0b10: CMAC (4-byte transmitted) 0b11: RFU		0b000: No data encryption 0b001: Reserved for future use 0b010: Reserved for future use 0b011: VAES (using AES-128) 0b100 … 0b111: RFU		

Table 16. Legacy SLF field structure

C.1 RLC_TYPE Options

This field defines the type of RLC used for security processing.

- 0b000: RLC is not used
- 0b001: Reserved for future use
- 0b010: 16-bit implicit RLC (not transmitted in data telegrams)
  Roll over is allowed and a window algorithm must be used
- 0b011: 16-bit explicit RLC (16-bit transmitted in data telegrams)
  Roll over is allowed and a window algorithm must be used
- 0b100: 24-bit implicit RLC (not transmitted in data telegrams)
  Roll over is allowed and a window algorithm must be used
- 0b101: 24-bit explicit RLC (24-bit transmitted in data telegrams)
  Roll over is not allowed
- 0b110: 32-bit explicit RLC (24-bit transmitted in data telegrams)
  Roll over is not allowed
- 0b111: 32-bit explicit RLC (32-bit transmitted in data telegrams)
  Roll over is not allowed

C.2 CMAC_TYPE Options

This field defines the type of CMAC that is provided as part of data telegrams.

- 0b00: CMAC is not used
- 0b01: CMAC is a 3-byte-long code that is calculated using the CMAC algorithm
- 0b10: CMAC is a 4-byte-long code that is calculated using the CMAC algorithm
- 0b11: Reserved for future use

C.3 ENCRYPTION_TYPE Options

This field defines the algorithm used for encryption and decryption of data telegrams.

- 0b000: DATA is not encrypted
- 0b011: DATA will be encrypted/decrypted using VAES
- Others: Reserved for future use