System Specification

enocean alliance
Building Smarter Connectivity

Security of EnOcean Radio Networks

V 3.01

San Ramon, CA, USA, 2023

Version 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 RORG.	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

EnOcean Alliance Inc.

5000 Executive Parkway, Suite 302

San Ramon, CA 94583
Graham Martin, Chairman & CEO, EnOcean Alliance

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 Terms & Abbreviations

Term / Abbr.	Description
AES	Advanced Encryption Standard
CMAC	Cipher Based Message Authentication Code
Data	Payload of an ERP telegram
Device	Customer end-device with an integrated EnOcean radio module
EEP	EnOcean Equipment Profile
ERP 1/2	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
MSB	Most significant byte
PK	Private key
PSK	Pre-shared key
RECOM	Remote Commissioning
REMAN	Remote Management
RLC	Rolling Code
R-ORG	Radio-Organization (defining the telegram type of an EnOcean telegram)
R-ORG-S	R-ORG for security processing
SLF	Security Level Format (defining the security parameters used during secure communication)

Table 1. Abbreviations used in this document.

1.2 Referenced documents

(1) NIST,FIPS 197, *Advanced encryption standard (AES)*, 2001

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

(2) JH. Song, R. Poovendran, J. Lee, T. Iwata, 2006, *The AES-CMAC Algorithm*

http://www.rfc-editor.org/rfc/rfc4493.txt

(3) Wikipedia, *Block cipher modes of operation*.

http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

(4) 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

2 High-level functionality

This specification defines security mechanisms for data exchange using the EnOcean Radio Protocol (ERP). These mechanisms apply both for the ERP1 (ISO 14543-3-10) and the ERP2 (ISO 14543-3-11) variants of EnOcean Radio Protocol.

2.1 ERP system architecture

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.

2.1.1 ERP device identification

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

2.1.2 ERP data model

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

- R-ORG
  RORG identifies the high-level EEP type. **Table 2** below lists common EEP types and their R-ORG.

R-ORG	Description	Typical Use
0xA5	4 byte sensor telegram (4BS)	Sensor telegrams with 4 byte payload
0xA6	Addressed data telegram (ADT)	Telegrams that specify the intended receiver
0xC5	Remote management telegram (SYS_EX)	Configuration of functional parameters in the receiver
0xD0	Signal telegram (SIGNAL)	Reporting of system parameters
0xD1	Manufacturer-specific content (MSC)	Manufacturer-defined telegrams
0xD2	Variable length telegram (VLD)	Variable length telegrams with more than 4 byte payload
0xD5	1 byte sensor telegram (1BS)	Simple sensors with 1 byte payload such as contact sensors
0xF6	Rocker and pushbutton switches (RPS)	Rocker switches or push buttons

Table 2. Common EEP types and R-ORG

- 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
  VARIANT identifies the exact EEP variant within the EEP function group, for instance a 0 °C – 40 °C EEP that is defined within the EEP function group of temperature sensors

The R-ORG, FUNCTION and VARIANT fields described above are combined into a 21-bit long EEP identifier as shown in **Figure 1** below.

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

Figure 1. EEP identifier structure

The complete 21-bit EEP identifier is only transmitted during the initial teach-in (paring) 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 telegrams.

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

2.1.3 ERP topology

ERP provides a flexible topology for a wide variety of use cases. The most common topology is uni-directional broadcast where one transmitter (for instance a light switch) communicates with one or several receivers (for instance lighting controllers) 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 EURID(s). 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 EURID 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 might respond to received telegrams, for instance when queried for their status. This sequence of request and response (or transmission and reception) is called bi-directional communication.

Figure 2 below illustrates the ERP system architecture.

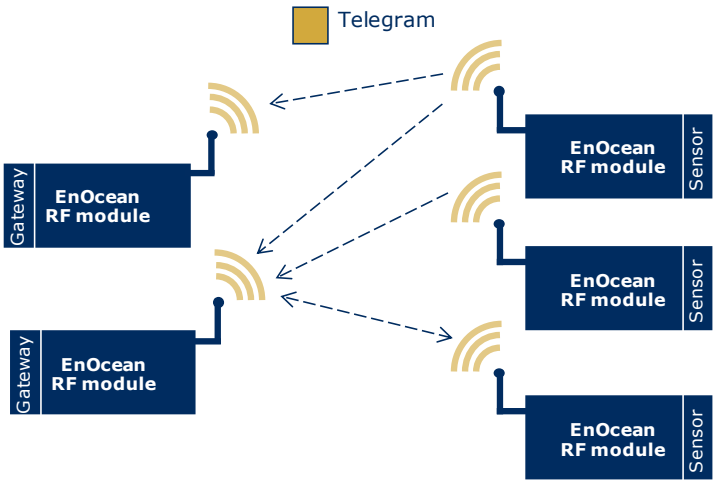

Figure 2. ERP system architecture

2.2 Security use case examples

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.

2.2.1 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 telegram content so that it remains unknown for unauthorized receivers.

2.2.2 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 telegram content to ensure that the telegram 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 telegram cannot be retransmitted later.

2.3 Required security functionality

Based on the discussion of the previous sections, we can identify four required security pillars:

- Confidentiality
  It should not be possible for an unauthorized receiver to understand the data that is exchanged between ERP devices
- Authenticity
  It should not be possible for an unauthorized transmitter to transmit data telegrams that will be accepted by a receiver. For instance, it should be possible only for authorized transmitters to control the lighting or the ventilation system in a building
- Integrity
  It should not be possible to modify the content of a transmitted message by an unauthorized intermediary
- Uniqueness (Replay protection)
  It should not be possible for an unauthorized transmitter to capture and re-transmit (reuse) a previously transmitted data telegram

ERP systems support these four pillars as described in the next chapters.

3 ERP security concepts

This chapter explains key concepts of ERP security used to implement the security requirements identified in the previous chapters.

3.1 Private Key (PK)

Transmitter and receiver authorize each other by their mutual possession of a shared secret – the private key (PK). For ERP systems, the PK has a size of is 128 bits (16 bytes). If transmitter and receiver both possess the same PK, then the receiver will accept telegrams from the transmitter.

For the case of bi-directional communication between two ERP devices, either the same PK or a different PK can be used by the two devices depending on the specific use case. If both devices transmit similar telegrams such as data telegrams where no Message-ID is used, then different PK must be used.

3.1.1 Private Key assignment

The PK shall be assigned to an ERP device based on the following guidelines:

1. Each device shall be pre-configured with a unique, randomly generated PK
2. The PK shall be provided by QR code of the device label as defined by the EnOcean Alliance Product ID and Standardized Labeling specification
3. Each device shall have the possibility to change the PK unless technical limitations such as unidirectional devices with no external interfaces and no user interface prevent this
4. Each PK shall be uniquely generated. The same PK shall not be used for several devices

3.1.2 Private Key usage in bi-directional communication

For bi-directional communication, the following rules apply:

1. If two devices communicate bi-directionally with each other using different R-ORG (asymmetric data content), then the same PK should be used for transmission by each of the two devices.
2. If two devices communicate bi-directionally with each other using the same R-ORG (symmetric data content), then different PK shall be used for transmission by each of the two devices.
3. If a device communicates bi-directionally with more than one communication partner, then different PK shall be used for each communication partner according to the rules above. Broadcast shall be seen for this use case as “one communication” partner.

The examples below illustrate these requirements:

- *Occupancy Sensor* provides status updates via broadcast to all devices within radio range. Occupancy Sensor uses PK1 (which is its own PK) to secure its transmissions. All devices possessing PK1 (in this case, Light Controller and Gateway) can decrypt and authenticate radio telegrams received from Occupancy Sensor.
- *Light Controller* accepts control requests from Gateway using 4BS telegrams and reports status information (dim levels, occupancy state, demand response state) using VLD telegrams
  The same key (PK2) can be used for requests from Gateway to Light Controller and for the reporting from the Light Controller to the Gateway since different R-ORG (4BS and VLD) are used for the two directions.
- *Heating Valve* accepts control requests from Gateway using 4BS telegrams and reports status information (valve position, temperature, offset) also using 4BS telegrams. Different keys (PK3 and PK4) are used since the same R-ORG (4BS) is used in both directions.

Figure 3 shows the PK usage for these communication scenarios.

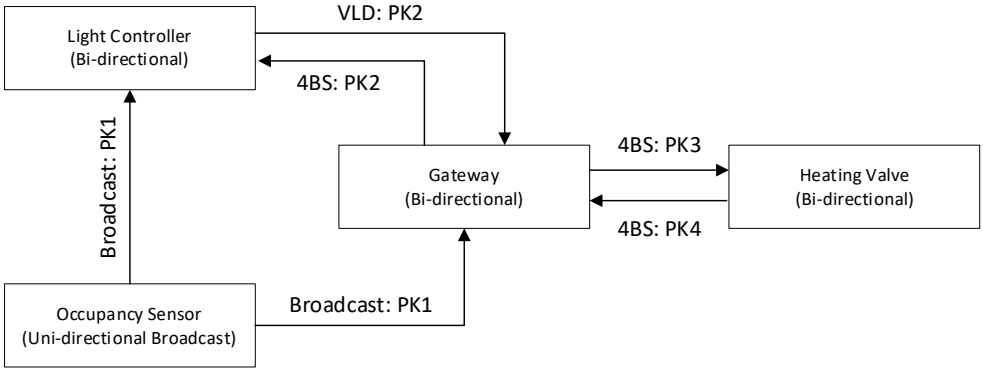

Figure 3. PK usage scenarios

3.1.3 Private Key update

The PK of a device might be changed in various ways, for instance via button press, via the ERP radio interface, via a HW interface or via NFC.

PK update via ERP radio protocol is possible thanks to Secure-REMAN/RECOM using the “Set Device Security Information Content” command. Please refer to the EnOcean Alliance Remote Commissioning specification for details.

Alternatively, a device’s PK can be changed by receiving a SEC-TI telegram (as defined in Chapter **5.2**) which contains the new private key, and RLC set to 0, and both fields are encrypted with the PSK of the device. With this mechanism, the device can authenticate the received private key/RLC update request and can perform the change – even after a factory reset.

After changing the private key, the RLC (described in Chapter **3.2**) shall be set to 0. To communicate the PK update, the device shall either send a data telegram encrypted with the new PK and the new RLC or send a SEC-TI telegram.

3.2 Rolling Code (RLC)

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

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

Different mechanisms can 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 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 to guarantee that the same RLC will not reoccur during the device lifetime. RLC will be initialized to 0 at production or whenever a new PK is used. After that, RLC will be incremented for each data telegram that is transmitted.

The transmitter device will store the RLC that was most recently used for transmission and increment it before the transmission of the next data telegram. Likewise, the receiver device will store the RLC that was most recently received and will only accept data telegrams with a higher RLC value.

3.2.1 Explicit versus implicit RLC

As discussed above, using an RLC for replay protection requires that the receiver can determine the RLC that is used in the transmitted telegram. For that, it is recommended that the RLC will be transmitted as part of every data telegram ("explicit RLC"). This allows the receiver to directly determine the current RLC and process the telegram accordingly.

For extremely energy-constrained devices, it might however be required to omit the transmission of the RLC to achieve a smaller telegram size and thereby conserve energy. For these cases, ERP provides the option of transmitting the RLC only during system setup (teach-in) and omit it during the transmission of normal data telegrams. In this case, the receiver must keep track of the expected RLC value by incrementing its RLC counter for each received telegram. This approach is called "implicit RLC".

The main issue with using an implicit RLC is that the receiver might miss one or several transmitted data telegrams due to temporary power loss or the transmitter being temporarily out of radio range. In that case, the transmitter will use an RLC that is different (higher) than the RLC expected by the receiver. The receiver will therefore always try a range of possible RLC values – within the co-called *RLC Window* – for the case of non-successful security processing with the expected RLC.

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 and the possibility of denial-of-service attacks as discussed below.

The most common choice for RLC window size is 128 meaning that the receiver will try both the expected RLC and up to 127 subsequent RLC values until the authentication signature matches the expected value. The figure below illustrates the use of an RLC window.

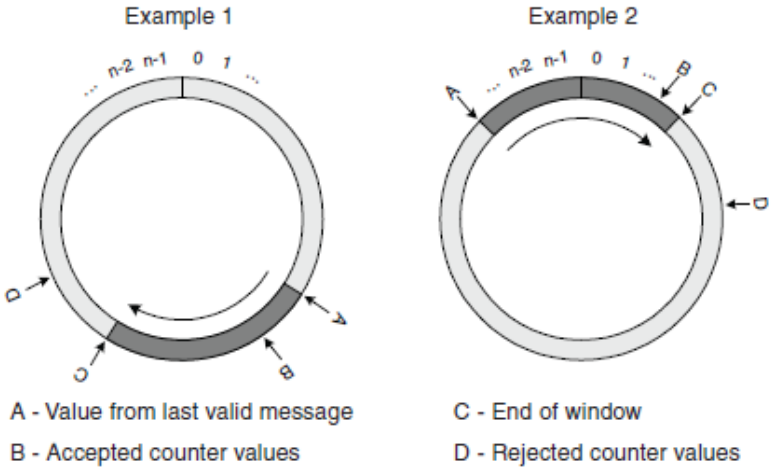

Figure 4. Rolling Window of Acceptance for Counter Values

The 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 telegrams were lost), telegrams 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 the RLC window (meaning that authentication and decryption of the data telegram 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 telegram fail. The receiver may then block this specific transmitter (identified by it EURID) so that no security processing is done for subsequently received data telegrams or enforce a backoff timing whereby security processing is suspended for an application-defined period of time. This avoids the case where the receiver spends significant processing trying to decrypt and authenticate telegrams that were sent with incorrect PK or incorrect RLC by an unauthorized transmitter.

Note: The implementation of such blocking mechanism could enable denial of service attacks by sending random security telegrams using the known EURID of the attacked transmitter to cause the receiver to temporarily or permanently block the transmitter. The use of implicit RLC is therefore strongly discouraged.

If transmitter and receiver RLC were desynchronized with a RLC difference bigger than the RLC window size, then they can be synchronized again either by means of the Teach-in procedure described in Chapter **5** or by directly configuring a new the RLC into the receiver.

ERP supports the use of 24-bit and 32-bit RLC sizes; the use of 32-bit is recommended. The previously existing option of using 16-bit RLC has been deprecated. Use of implicit RLC is not recommended for new designs due to the limitations described above.

3.3 Authentication Signature (CMAC)

Authenticity and integrity of a received data telegram are guaranteed by an authentication signature that is calculated based on telegram data, RLC and PK. In ERP, the authentication signature is called *Cipher-based Message Authentication Code* or *CMAC* in short.

CMAC is calculated using the algorithm described in Chapter **6.2** and appended to the telegram data using big-endian byte order (meaning the most significant byte is transmitted first). 3-byte and 4-byte CMAC lengths are supported by ERP; the previous option of using 2-byte CMAC has been deprecated.

Upon receiving a telegram, the receiver will calculate the expected CMAC based on telegram data, RLC and PK. If the expected CMAC matches the transmitted CMAC then the following is true:

- Authenticity
  The data telegram was transmitted by a transmitter possessing the same PK as the receiver
- Integrity
  The data telegram content has not been modified
- Uniqueness (Replay protection)
  The data telegram CMAC was calculated based on the expected RLC value.

3.4 Encryption (VAES)

The data payload (EEP data) of ERP telegrams is encrypted using the Variable AES (VAES) algorithm in conjunction with the PK as described in Chapter **6.1**. The decryption of the encrypted data payload is described in Chapter **6.1.3**.

3.5 Security Level Format (SLF)

As discussed in the previous chapters, ERP supports different security options which will be selected depending on available energy. The security options used for communication between a transmitter and a receiver are specified during the security setup (teach-in) using the 1-byte Security Level Format (SLF) field transmitted in the secure teach-in telegram or setup by the host. All subsequent data telegrams are required to use this SLF.

The SLF field is organized into three different sub-fields:

- RLC Type
  Type (explicit or implicit) and length (4-byte or 3-byte) of the RLC
- CMAC Type
  Length of the CMAC field
- Encryption Type
  Type of the encryption algorithm used

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

7	6	5	4	3	2	1	0
RLC_TYPE			**CMAC_TYPE**		**ENCRYPTION_TYPE**		

3.5.1 Supported RLC types

The following RLC_TYPE options are supported for current ERP implementations:

- 0b100: 24-bit RLC used, RLC not transmitted in data telegrams (implicit RLC)
- 0b101: 24-bit RLC used, 24-bit RLC transmitted in data telegrams
- 0b110: 32-bit RLC used, 24-bit RLC transmitted in data telegrams (required by legacy receivers)
- **0b111: 32-bit RLC used, 32-bit RLC transmitted in data telegrams**

The last option (marked in bold) is the recommended option.

3.5.2 Supported CMAC types

The following CMAC_TYPE options are supported for current ERP implementations:

- 0b01: 24-bit CMAC
- **0b10: 32-bit CMAC**

The last option (marked in bold) is the recommended option.

3.5.3 Supported Encryption types

Current ERP implementations support only VAES encryption with RLC as defined in Chapter **6.1**.

- **0b011: VAES with RLC**

Previously defined options that are now deprecated are described in Chapter **A.5**.

3.6 Security parameter selection

Security implementations in ERP devices shall use the recommended security parameters specified in the previous sections unless they are constrained by available energy. If ERP devices are constrained by available energy, then they shall select the security parameters for energy-constrained devices.

3.6.1 Standard security level format

The standard security level format shall 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).

The standard security level format uses the following parameters:

- RLC_TYPE 0b111: 32-bit RLC used, 32-bit RLC transmitted in data telegrams

- CMAC_TYPE 0b10: 32-bit CMAC
- ENCRYPTION_TYPE 0b011: VAES with RLC

The resulting SLF value is 0xF3.

3.6.2 Energy-reduced security level format

The energy-reduced security level format shall be used for ERP devices with constrained energy budget such as solar-powered sensors or kinetic-harvesting switches.

The energy-reduced security level format uses the following parameters:

- RLC_TYPE 0b101: 24-bit RLC used, 24-bit RLC transmitted in data telegrams
  0b110: 32-bit RLC used, 24-bit RLC transmitted in data telegrams
- CMAC_TYPE 0b10: 32-bit CMAC
- ENCRYPTION_TYPE 0b011: VAES with RLC

Note that two different RLC types are supported as certain legacy receivers support only the use of 24-bit RLC. The resulting SLF value is therefore either 0xAB (24-bit RLC used) or 0xCB (32-bit RLC used).

3.6.3 Ultra-low power security level format (Not recommended for new designs)

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 energy-reduced security level format. Use of the ultra-low power security level format is not recommended for new designs.

The ultra-low power security level format uses the following parameters:

- RLC_TYPE 0b100: 24-bit RLC used, RLC not transmitted in data telegrams
- CMAC_TYPE 0b01: 24-bit CMAC
- ENCRYPTION_TYPE 0b011: VAES with RLC

The resulting SLF value is therefore 0x8B.

4 Security telegram processing

ERP security is implemented on the OSI presentation layer of the ERP protocol stack meaning that it operates on ERP telegrams without affecting lower layer specifics such as checksums or physical radio properties.

4.1 ERP telegram structure

ERP telegrams consist of the following parts:

- R-ORG
  The R-ORG field identifies the high-level telegram type and allows to distinguish between short messages such as switch messages using RPS R-ORG and more complex messages such as sensor messages using 1BS, 4BS or VLD R-ORG.
- Data fields
  The DATA field contains the telegram payload together with the security data fields RLC (if transmitted) and CMAC. ERP2 telegrams allow the transmission of additional OPTIONAL_DATA in the telegram.
- Control fields
  ERP telegrams contain various control fields such as R-ORG, STATUS and HASH for ERP1 telegrams or LENGTH, HEADER, EXT_HEADER, EXT_TYPE and CRC for ERP2 telegrams.

ERP1 telegram combine these fields into the structure shown in **Figure 5** below.

R-ORG	DATA	SENDER EURID	STATUS	HASH
1 byte	1 ... 14 byte	4 byte	1 byte	1 byte

Figure 5. ERP1 telegram structure

ERP2 telegrams use a more generic structure shown in **Figure 6** below. Note that for ERP2, the R-ORG is for the most common R-ORG types specified by the HEADER field; less common R-ORG types are specified by the EXT_TYPE field. Fields marked with grey background are optional.

LENGTH	HEADER	EXT_HEADER	EXT_TYPE	DESTINATION EURID	SENDER EURID	DATA	OPTIONAL_DATA	CRC
1 Byte	1 Byte	0 / 1 Byte	0 / 1 Byte	0 / 4 Byte	4 / 6 Byte	Variable	0 / Variable	1 Byte

Figure 6. ERP2 telegram structure

4.2 Security processing

ERP security processing will encrypt / decrypt and authenticate the data of ERP telegrams. To do so, the following steps will be used:

- Management of an RLC as described in Chapter **3.2**
- Encryption or decryption of the message data using the VAES algorithm described in Chapter **6.1** and Chapter **6.1.3**
- Authentication based on a CMAC authentication signature described in Chapter **3.3** using the CMAC algorithm described in Chapter **6.2**.
- Secure message chaining as described in Chapter **4.2.5.**

The data used for the security processing is the concatenation of the ERP message R-ORG (if present) and the ERP message DATA. The following chapters provide examples of this processing.

4.2.1 Secure ERP telegram structure

ERP security processing affects the R-ORG and the DATA fields of ERP telegrams:

- The R-ORG field is changed from the original R-ORG type to a new R-ORG type called R-ORG-S to identify a secure ERP telegram. The original R-ORG might be transmitted as part of the subsequent data field.
- The content of the DATA field is encrypted and then combined with the CMAC authentication signature as well as RLC and original R-ORG (if present).

Note that only the DATA and – optionally – the R-ORG fields of ERP telegrams will be affected by the ERP security processing. Other ERP telegram fields such as transmitter EURID, receiver EURID, status and checksum will not be modified. Unmodified fields might therefore not be depicted or discussed when ERP security processing is subsequently described.

The structure of secure ERP telegrams (including the length of the RLC and CMAC fields) is determined during the Teach-in procedure described in Chapter **5** and expressed by the SLF field described in Chapter **3.5**.

Figure 7 below illustrates the security processing of the ERP telegram content.

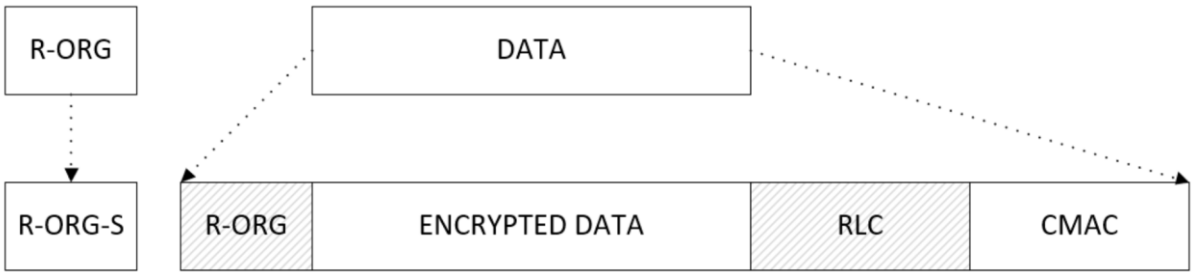

Figure 7. Security processing of ERP telegrams

4.2.2 R-ORG processing

As described in Chapter **2.1.2**, the high-level ERP telegram type is identified by the 8-bit R-ORG field that is transmitted in all ERP telegrams. Specific R-ORG types – called R-ORG-S – are used to identify ERP telegram types related to ERP security processing.

Table 3 below lists the R-ORG-S types defined for the security processing of ERP telegrams.

RORG-S	Description	Usage Examples
0x30 (SEC)	Secure message that does not include R-ORG (Encryption of messages without R-ORG)	Secure PTM switch (RPS) messages
0x31 (SEC_R)	Secure message that includes the R-ORG of the original message (R-ORG and DATA are encrypted together)	Secure sensor (1BS, 4BS, VLD) messages
0x32 (SEC_D)	Decrypted message that does not include R-ORG Result of decryption of a message with R-ORG 0x30 (SEC)	Decrypted PTM switch (RPS) messages
0x33 (SEC_CDM)	Secure chained messages Message content is transmitted using several telegrams	Secure sensor messages requiring chaining due to length
0x35 (SEC_TI)	Secure teach-in telegram	Setup of a secure communication channel

Table 3. R-ORG-S types

As discussed above, R-ORG-S allows distinction between secure telegrams that do not include the R-ORG type of the original telegram (R-ORG-S 0x30: SEC) and secure telegrams that include the R-ORG of the original telegram (R-ORG-S 0x31: SEC_R).

The first case applies to RPS telegrams from PTM (kinetic switch) modules which contain the R-ORG field within the telegram. The processing steps for this case are described in Chapter **4.2.3**.

The second case applies to all other telegram types which do transmit the R-ORG field as part of their telegrams. The processing steps for that case are described in Chapter **4.2.4**.

ERP transmits telegrams using very short, redundant sub-telegrams; therefore, it is required to partition (segment) large telegrams into shorter telegram parts. ERP calls this process *Message Chaining*. Secure ERP telegrams that are part of a message chain are identified using R-ORG-S 0x35: SEC_CDM. The processing of such chained secure messages is described in Chapter **4.2.5**.

4.2.3 Security processing for telegrams without R-ORG field

If ERP telegrams do not include the R-ORG field, then the transmitter will only encrypt and authenticate the ERP telegram DATA. The transmitter will transmit a secure telegram using R-ORG-S 0x30 (SEC) containing the concatenation of the encrypted DATA, the current RLC value and the authentication signature CMAC.

The receiver will decrypt and authenticate the telegram to determine the ERP telegram DATA. The decrypted and authenticated telegram will use R-ORG-S 0x32: SEC_D to inform the receiver that the DATA content of the secure telegram was successfully authenticated and decrypted but that no R-ORG information was provided as part of the secure telegram.

The security processing for such telegrams is illustrated in **Figure 8** below.

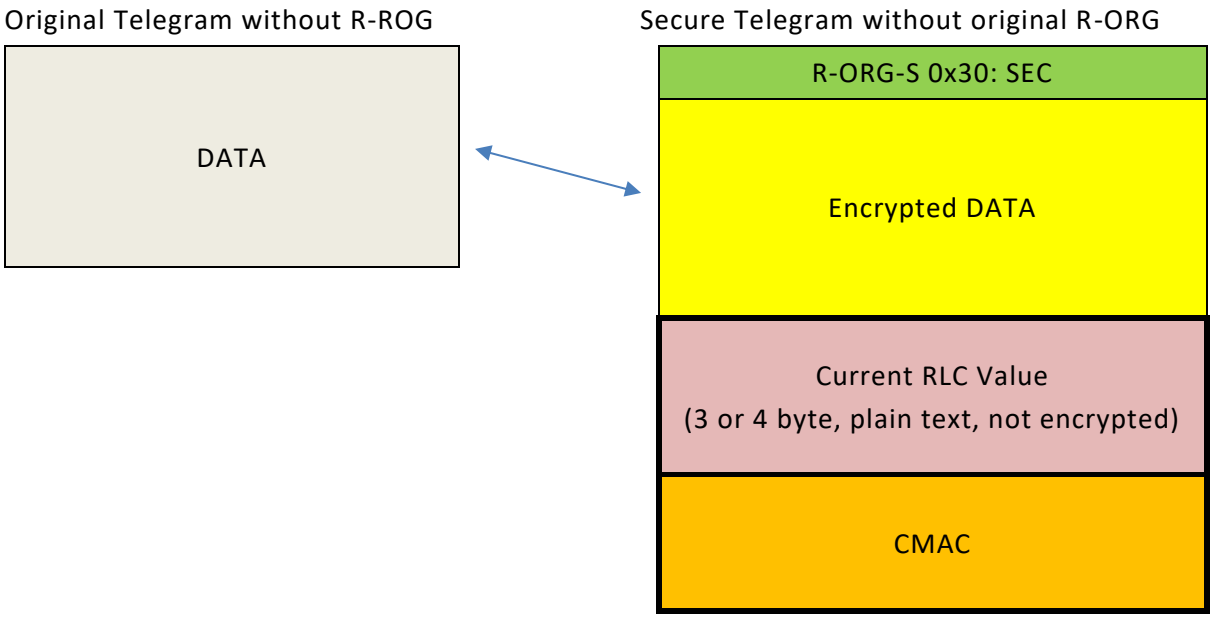

Figure 8. Security processing for ERP telegrams without R-ORG field

4.2.4 Security processing for telegrams with R-ORG field

If ERP telegrams include the R-ORG field, then the transmitter will encrypt and authenticate the R-ORG field of the ERP telegram together with the DATA field of the ERP telegram. The transmitter will transmit a secure telegram using R-ORG-S 0x31: SEC_R containing the concatenation of the encrypted R-ORG field, the encrypted DATA field, the current RLC value and the authentication signature CMAC.

The receiver will decrypt and authenticate such received telegram to determine the R-ORG and DATA content. This process is illustrated in **Figure 9** below.

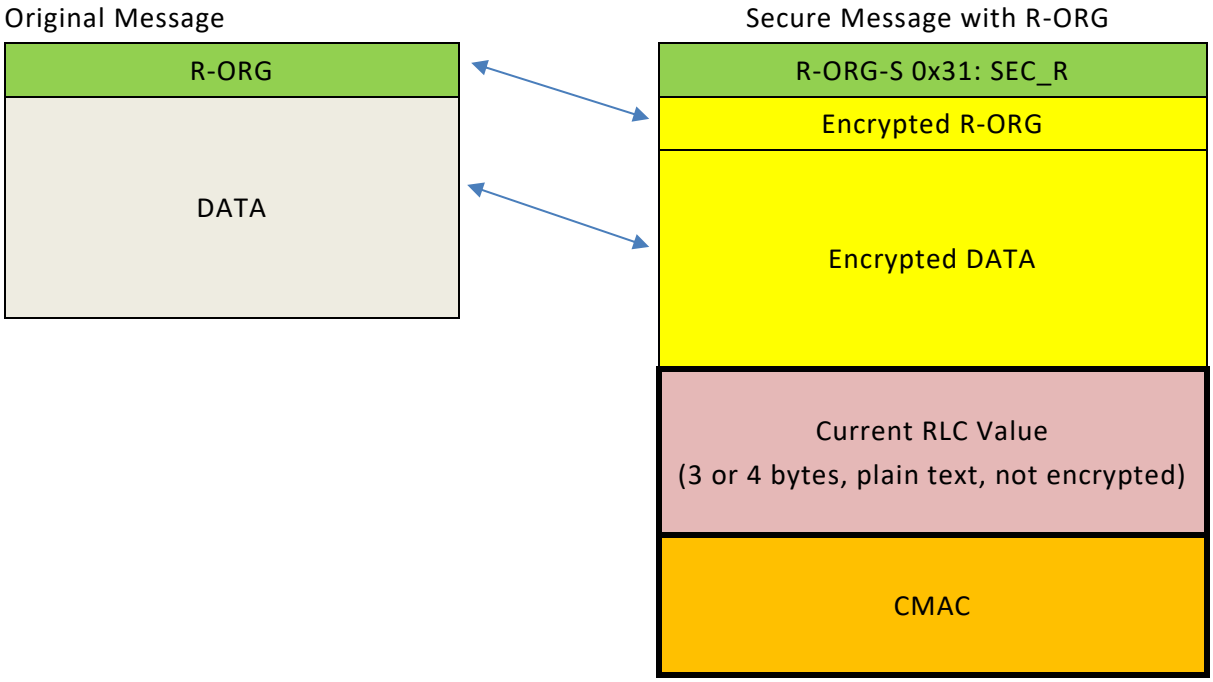

Figure 9. Security processing for ERP telegrams with R-ORG field

4.2.5 Secure message chaining

Secure message chaining is an extension to the ERP message chaining mechanism to allow the transmission of longer data telegrams which exceed the maximum length of an ERP telegram. Note that such telegrams always include the R-ORG field.

Secure message chaining is mainly used for ERP1 telegrams due to their limited telegram size. There, secure message chaining is required if the combined length of the telegram payload, CMAC and RLC fields is more than 14 bytes. For the case of 4 byte RLC size and 4 byte CMAC size, this means that secure message chaining is required if the telegram payload size is more than 6 byte.

ERP2 telegrams can be much longer than ERP1 telegrams and secure message chaining is therefore usually not needed for ERP2 telegrams.

Secure message chaining is done by first processing the ERP telegram according to steps defined in Chapter 4.2.4 using R-ORG-S 0x31: SEC_R for the calculation of the CMAC authentication signature.

The resulting secure telegram will then be chained (partitioned) into a sequence of telegrams which are identified by R-ORG 0x33: SEC_CDM.

This approach has the advantage that the transmission of the CMAC and RLC fields occurs only once (and not for each telegram of the chain) and that only one R-ORG encapsulation takes place.

Message chaining splits one data telegram into several chained messages. Each message in a chain contains a 2-bit sequence number SEQ and a 6-bit index IDX.

The sequence number SEQ is the same for each message in the same chain and identifies messages that belong to the same chain. SEQ increases whenever a new message chain is transmitted and cannot be equal to zero. This means that SEQ can be either 0b01, 0b10 or 0b11 which allows to distinguish up to three different message chains.

Note that many ERP devices have only a limited memory capacity which might restrict their ability to process many large telegrams concurrently. It is therefore strongly recommended to transmit only one chained message to a given device at a time.

The index IDX indicates the order of the messages within the chain. IDX starts at zero for the first message and is incremented for each subsequent message in this chain. The purpose of the index is to indicate the correct order of the messages within the chain.

In addition to that, the first message of a message chain (where IDX = 0b000000) contains the LENGTH field which indicates the length (in byte) of the combination of R-ORG field, telegram payload, RLC field and CMAC field of the secure ERP telegram.

The structures of the messages within a secure ERP1 message chain are shown in **Figure 10**, **Figure 11** and **Figure 12**. Appendix **A.4.3** provides a step-by-step example.

The following figure shows the structure for the first message of an ERP1 message chain.

	7	6	5	4	3	2	1	0
0	R-ORG-S 0x33: SEC-CDM							
1	SEQ		IDX = 0b000000					
2	LENGTH (Sent unencrypted, MSB first) Combined length of R-ORG, Payload, RLC and CMAC							
3								
4	Encrypted R-ORG							
5	First part of Encrypted payload + RLC + CMAC (10 bytes long)							
…								
14								
15	EURID							
16								
17								
18								
19	Status							
20	Checksum							

Figure 10. First message of an ERP1 message chain

The following figure shows the structure for an intermediary (not the first and not the last) message of an ERP1 message chain. This message type is present if the combined length of payload, RLC and CMAC is more than 23 byte.

	7	6	5	4	3	2	1	0
0	R-ORG-S 0x33: SEC-CDM							
1	SEQ		IDX > 0b000000					
2	Encrypted payload + RLC + CMAC (13 bytes long)							
…								
14								
15	EURID							
16								
17								
18								
19	Status							
20	Checksum							

Figure 11. Intermediary (not first, not last) messages of an ERP1 message chain

Finally, the figure below shows the structure for the last message of an ERP1 message chain.

	7	6	5	4	3	2	1	0
0	R-ORG SEC-CDM = 0x33							
1	SEQ			IDX				
2	Last part of Encrypted payload + RLC + CMAC (Variable length, up to 13 bytes)							
…								
length + 1								
	EURID (4 bytes)							
length + 5								
length + 6	Status (1 byte)							
length + 7	Checksum (1 byte)							

Figure 12. Last message of an ERP1 message chain

5 Security teach-in

Security teach-in is the procedure of configuring the parameters for secure communication between transmitter and receiver.

Within this security teach-in procedure, transmitter and receiver will agree on the security level format (SLF) that shall be used for communication between them and exchange the EURID, the PK and the current RLC value that will be used for the transmission of secure telegrams.

This chapter explains different possibilities for the security teach-in process and defines the required procedures.

5.1 Security teach-in using out-of-band mechanisms

EURID and PK of one device can be provided to the other device using out-of-band mechanisms such as 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 PK over the air using a secure teach-in telegram as described in Chapter **5.2**.

If such out-of-band approach is used, then the receiver will initialize the RLC for the transmitter based on the first received secure telegram that is successfully authenticated and decrypted. This approach therefore requires that the RLC is transmitted as part of the secure telegram.

5.2 Security teach-in using a SEC-TI telegram

For cases where out-of-band teach-in is not possible, it is possible to use a teach-in procedure where the security teach-in information is transmitted in-band, meaning within a dedicated ERP data telegram identified by R-ORG-S 0x35: SEC_TI.

Security teach-in using a SEC-TI can either be a uni-directional process (whereby the transmitter communicates its security information to the receiver) or a bi-directional process (whereby first the m transmitter communicates its security information to the receiver and then the receiver responds with its own security information to the transmitter).

To describe this process, we consider two devices named Device A and Device B shown in **Figure 13**.

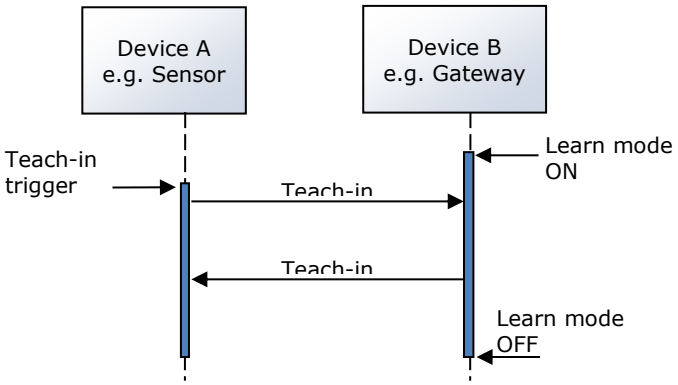

Figure 13. Schematic representation of the teach-in procedure

To execute the security teach-in procedure using a SEC-TI telegram, Device B must be set into *Learn Mode* (for instance via a button press) to accept security teach-in telegrams from other devices. Learn mode is usually time-limited to 30 seconds and security teach-in telegrams are only accepted while learn mode is active. This limits the likelihood of accepting unintended security teach-in telegrams.

Once Device B is set into learn mode, Device A must be triggered (for instance via a button press) to send a *Security Teach-in* (SEC-TI) telegram.

If Device B received the Security Teach-in telegram from Device A while being in learn mode, then it stores the EURID and the security parameters (PK, SLF and current RLC) of Device A that are transmitted in the Security Teach-in message.

For the case of bi-directional teach-in procedure, 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 must be transmitted no later than 500 milliseconds after the reception of the security teach-in message from Device A. Device A will accept security teach-in messages no later than 750 milliseconds after the transmission of its own security teach-in message.

PK and RLC within the security teach-in telegram can be encrypted using a pre-shared key (PSK) as described in Chapter **5.2.3** to prevent unauthorized reception. If Device A uses a PSK to encrypt PK and RLC within its security teach-in telegram and a bi-directional teach-in procedure is executed, then Device B will use the same PSK to encrypt its own PK and RLC within its security teach-in telegram to Device A.

5.2.1 Security teach-in telegram format

The format of the security teach-in telegram is shown in **Figure 14** below. It contains the SLF as described in Chapter **3.5** and the Teach-in Info as described in Chapter **5.2.2**.

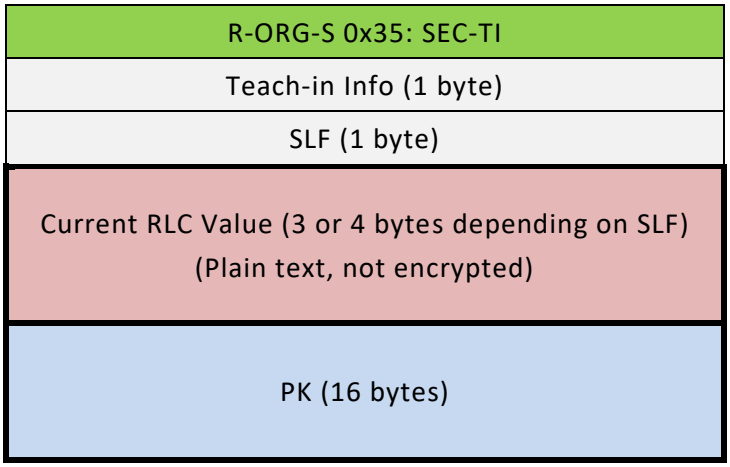

Figure 14. Security teach-in telegram format

The security teach-in telegram provides only security-specific data. This telegram does not contain the information about the EEP used for the data encoding of the ERP telegram (except when identifying the transmitter as PTM module). Information about the EEP must be transmitted separately after the security teach-in procedure as a secure telegram.

5.2.2 Teach-in Info format

The Teach-in Info field uses the format shown in **Figure 15**.

7	6	5	4	3	2	1	0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	

Figure 15. Teach-in Info format

For ERP1 messages, it is required to split the security teach-in telegram into a chain of two messages since the payload is more than 6 bytes long. See Chapter **4.2.5** for details regarding secure message chaining.

In the case of a security teach-in telegram, the IDX field of Teach-in Info will indicate the order of those messages with the first message having IDX = 0b00 and the second message having IDX = 0b01.

The first message (with IDX = 0) will provide additional information in the CNT, PSK, TYPE and INFO fields:

- CNT
  This field will identify the total number chained messages used for the transmission of the security teach-in message.
- PSK
  This field will identify if a pre-shared key (PSK) is used to encrypt RLC and PK
- TYPE
  This field will identify if the device is a PTM module or not
- INFO
  If the device is a PTM module, then this field will identify which rocker was pressed to trigger the security teach-in message transmission.
  If the device is not a PTM module, then this field will identify the security teach-in procedure type

Figure 16 below shows the structure of the Teach-in Info field for the first message of a security teach-in message chain for the case where a device is not a PTM module.

7	6	5	4	3	2	1	0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	
0b00 First message in the chain		0b10 2 chained messages used to transmit the security teach-in telegram		0b0 PSK not used 0b1 RLC and PK 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 16. Teach-in Info format for IDX = 0b00 and TYPE = 0b0 (non-PTM device)

Figure 17 below shows the structure of the Teach-in Info field for the first message of a security teach-in message chain for the case where a device is a PTM module.

7	6	5	4	3	2	1	0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	
0b00 First message in the chain		0b10 2 chained messages used to transmit the security teach-in telegram		0b0 PSK not used 0b1 RLC and PK 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 17. Teach-in Info format for IDX = 0b00 and TYPE = 0b1 (PTM device)

Figure 18 below shows the structure of the Teach-in Info field for the second message of a security teach-in message chain. This format applies to all device types.

7	6	5	4	3	2	1	0
IDX		**CNT**		**PSK**	**TYPE**	**INFO**	
0b01 Second message in the chain		0b00 Reserved		0b0 Reserved	0b0 Reserved	0b00 Reserved	

Figure 18. Teach-in Info format for IDX = 0b01

5.2.3 Secure Teach-in with PSK

PK and RLC can be protected during the transmission of the security teach-in message using encryption with a pre-shared key (PSK). This PSK must be known to the receiver of the security teach-in message and might be setup out-of-band, for instance by a device-specific configuration mechanism or by pre-configuration during device manufacturing.

If Secure Teach-in with PSK is used, then the PSK field of Teach-in Info will be set to 1 and RLC and PK will be encrypted using the VAES encryption process described in Chapter **6.1** with the following parameters:

- VAES DATA = Concatenation of RLC and PK
- VAES RLC = 0x0000
- VAES PRIVATE KEY = PSK

For the case of a bi-directional teach-in procedure, the receiving device will respond to the security teach-in telegram by its own security teach-in telegram communicating its own PK and RLC as described in Chapter **5.2**. The same PSK will be used both in the received request and in the transmitted response.

5.3 Bi-directional security teach-in without learn mode

If a device (Device 1) has obtained the PK of another device (Device 2) via an out-of-band mechanism (for instance a gateway that obtained the PK of another device via QR code scan of the device label), then the following procedure might be used to setup secure communication between the devices without using learn mode and a security teach-in telegram:

1. Device 1 transmits a data telegram encrypted using the PK of Device 2 in conjunction with RLC=0 and the standard SLF defined in Chapter **3.6.1**.

2. Device 2 authenticates the data telegram received from Device 1 using its own PK
   If the authentication is successful, then Device 2 determines that Device 1 has possession of its own PK and is therefore an authorized transmitter.

3. Device 2 will use its own PK for subsequent data telegram transmissions to Device 1. This means that the same PK will be used for both directions of the bi-directional communication.

Note: Different R-ORG shall be used for the different communication directions.

5.4 Determining support for ERP security without manual interaction

Certain legacy devices might not support ERP security; therefore, a process is required whereby one device (Device 1) can determine if another device (Device 2) supports ERP security without requiring manual interaction such as a button press.

The following procedure shall be used for that determination and the subsequent setup of secure communication between the devices:

1. Device 1 advertises its presence and functionality by transmitting non-encrypted, non-authenticated data telegrams.

2. Device 1 advertises its support for ERP security by periodically transmitting a security teach-in (SEC-TI) telegram using a pre-defined PSK and the desired SLF.

3. Device 1 continues to transmit non-encrypted, non-authenticated data telegrams while it does not receive a response from Device 2.

4. If Device 2 supports ERP security and is capable of supporting the SLF specified by Device 1 and knows the PSK used by Device 1, then Device 2 responds to the received SEC-TI telegram with another SEC-TI telegram containing the PK to be used in subsequent telegrams and the current RLC value. PK and RLC value will be protected using the same PSK that was used by Device 1 according to the procedure defined in Chapter 5.2.3.

5. Subsequent communication between Device 1 and Device 2 will use the security level format (SLF) requested by Device 1 and the PK selected by Device 2.

   Note: Different R-ORG shall be used for the different communication directions.

6 Security algorithms

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

6.1 VAES encryption and decryption

The ERP security implementation uses a variable AES (VAES) implementation to encrypt the concatenation of ERP telegram data and RLC using PK.

Encrypting the concatenation of RLC and ERP telegram data ensures that the same input data in different ERP telegrams will be transmitted as different encrypted data due to having different RLC values. As discussed in Chapter 3.2, the RLC will be incremented after each telegram transmission.

Using a VAES-128 algorithm rather than the AES-128 algorithm ensures that input data smaller than 16-byte can be used which is required for energy efficiency. While the AES-128 algorithm requires blocks of 128-bit (16 byte) of input data and produces blocks of 128-bit output data, VAES-128 can operate on input data of any size and will produce output data of the same size. Both AES-128 and VAES-128 use a 128-bit PK.

6.1.1 Input data

The input data for the encryption process is the concatenation of the 1-byte R-ORG field, which identifies the high-level EEP type, with the EEP data. This is illustrated in **Figure 19** below.

Figure 19. VAES-128 input data

6.1.2 VAES encryption process

As illustrated in **Figure 20** below, VAES-128 uses a sequential process of XOR and AES-128 operations for processing the input data together with PK and RLC to produce the encrypted payload.

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.

Appendix **A.4** provides step-by-step processing examples.

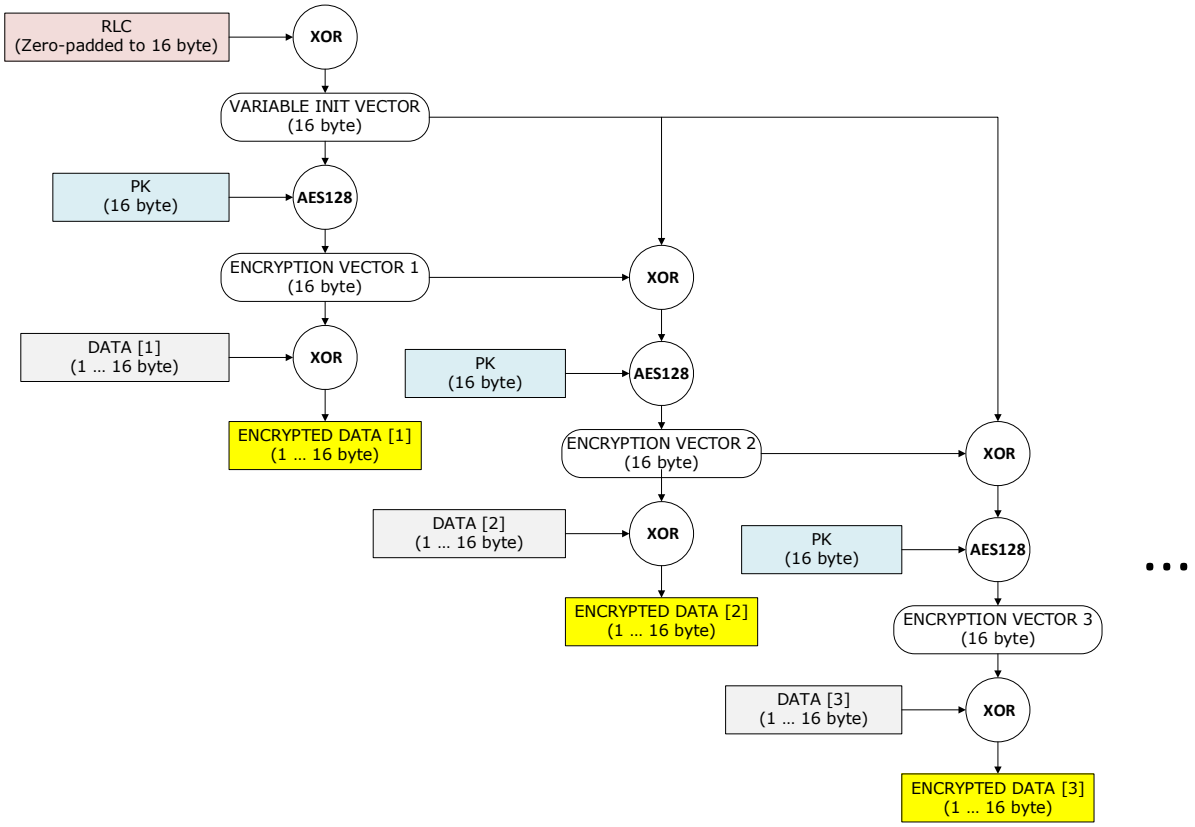

Figure 20. VAES-128 encryption process

6.1.3 VAES decryption process

As illustrated in **Figure 21** below, the VAES-128 decryption process is the inversion of the VAES-128 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.

Appendix **A.4** provides step-by-step processing examples.

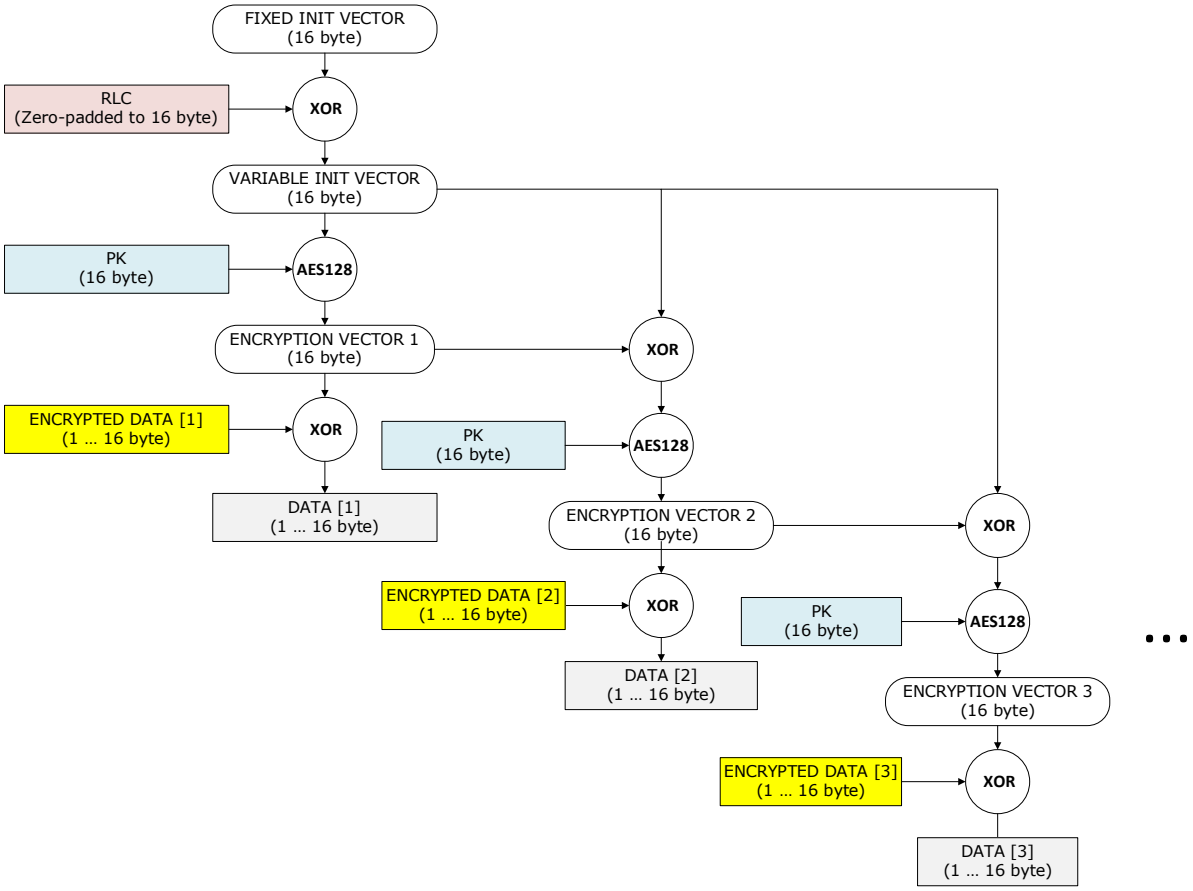

Figure 21. VAES-128 decryption process

6.1.4 VAES INIT VECTOR

VAES INIT VECTOR is a constant used as input to the VAES-128 algorithm. For ERP security processing, the value of VAES INIT VECTOR has been defined as **0x3410DE8F1ABA3EFF9F5A117172EACABD**.

6.2 CMAC authentication

The ERP security implementation uses the authentication mechanism defined by RFC4493 as described in [2] 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 **6.2.1** is a multiple of 16 bytes (16 bytes, 32 bytes, …) or not.

6.2.1 Input data

The input data for the CMAC algorithm consists of the concatenation of R-ORG-S, the encrypted ERP telegram data and the RLC as shown in **Figure 22** below.

Note that if R-ORG S = 0x31: SEC_R, then DATA includes the R-ORG of the data telegram as most significant byte. If R-ORG S= 0x30: SEC, then DATA does not include the R-ORG of the data telegram.

If the input data is more than 16 bytes 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 is 16 bytes long. The last part might have any size between 1 and 16 bytes.

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

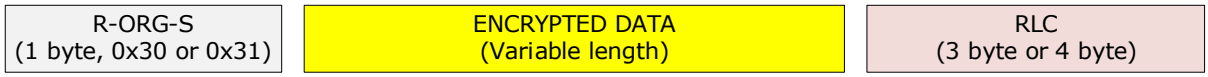

Figure 22. Input data for the CMAC algorithm

6.2.2 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 bytes as well. The CMAC authentication signature is then calculated according to the process shown in **Figure 23** below using PK and another key K1 which is derived from PK according to the procedure defined in Chapter **6.2.4**.

Appendix **A.4** provides step-by-step examples for the CMAC calculation.

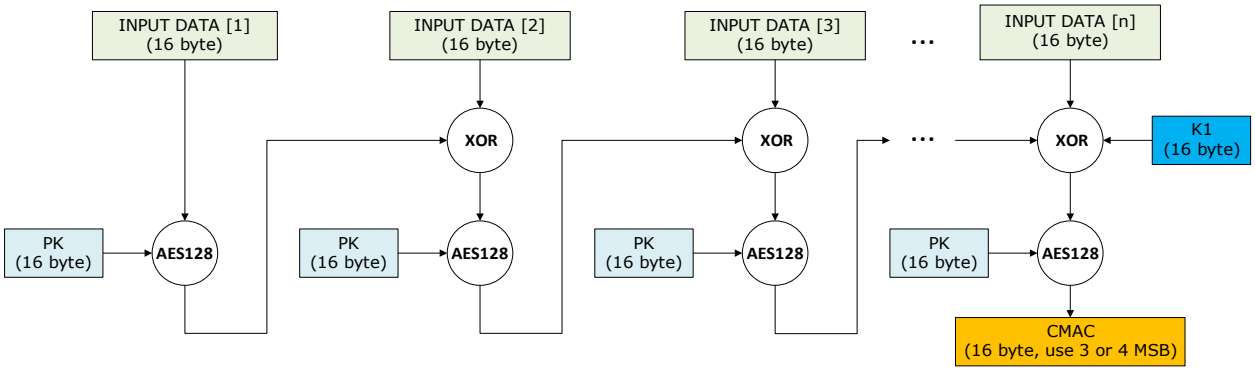

Figure 23. CMAC generation for input data with multiple of 16 bytes size

6.2.3 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 bytes.

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 bytes. If for instance the input data would be 13 bytes long, then three bytes of padding would be required to reach 16 bytes and these three bytes would have the value 0x800000.

The CMAC authentication signature will then be calculated according to the process shown in **Figure 24** below using PK and another key K2 which is derived from PK according to the procedure defined in Chapter **6.2.4**.

Appendix **A.4** provides step-by-step examples for the CMAC calculation.

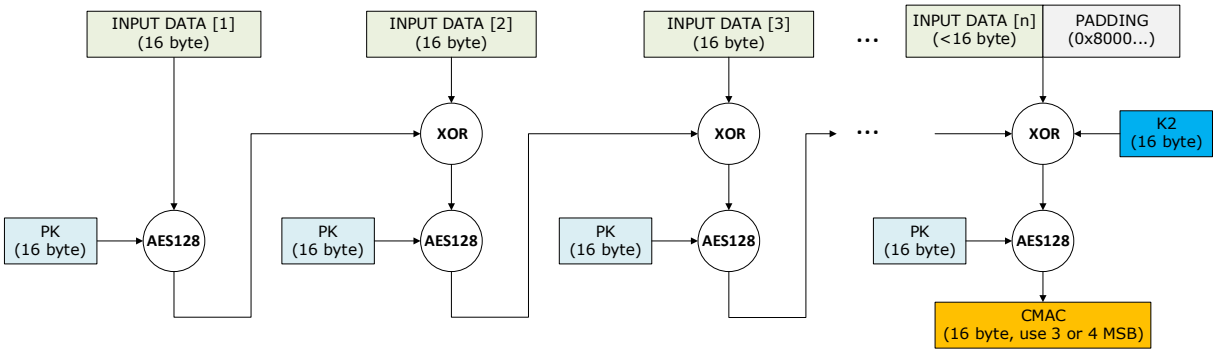

Figure 24. CMAC generation for input data with non-multiple of 16 bytes size

6.2.4 Key derivation of K1 and K2 from PK

As discussed above, the calculation of the CMAC authentication signature, which is defined by RFC4493 as described in [2], requires the derivation of the keys K1 and K2 from PK.

The process for this key derivation is shown in **Figure 25** for reference. Note that PK takes the role of K in this process.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                    Algorithm Generate_Subkey                         +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                                                                      +
+   Input    : K (128-bit 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  K2;                                               +
+                                                                      +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 25. Key derivation of K1 and K2 from PK

Annex A

A.1 ERP1

A.1.1 Secure data telegram format for ERP1

Figure 26 below illustrates the structure of a secure telegram in ERP1 radio protocol.

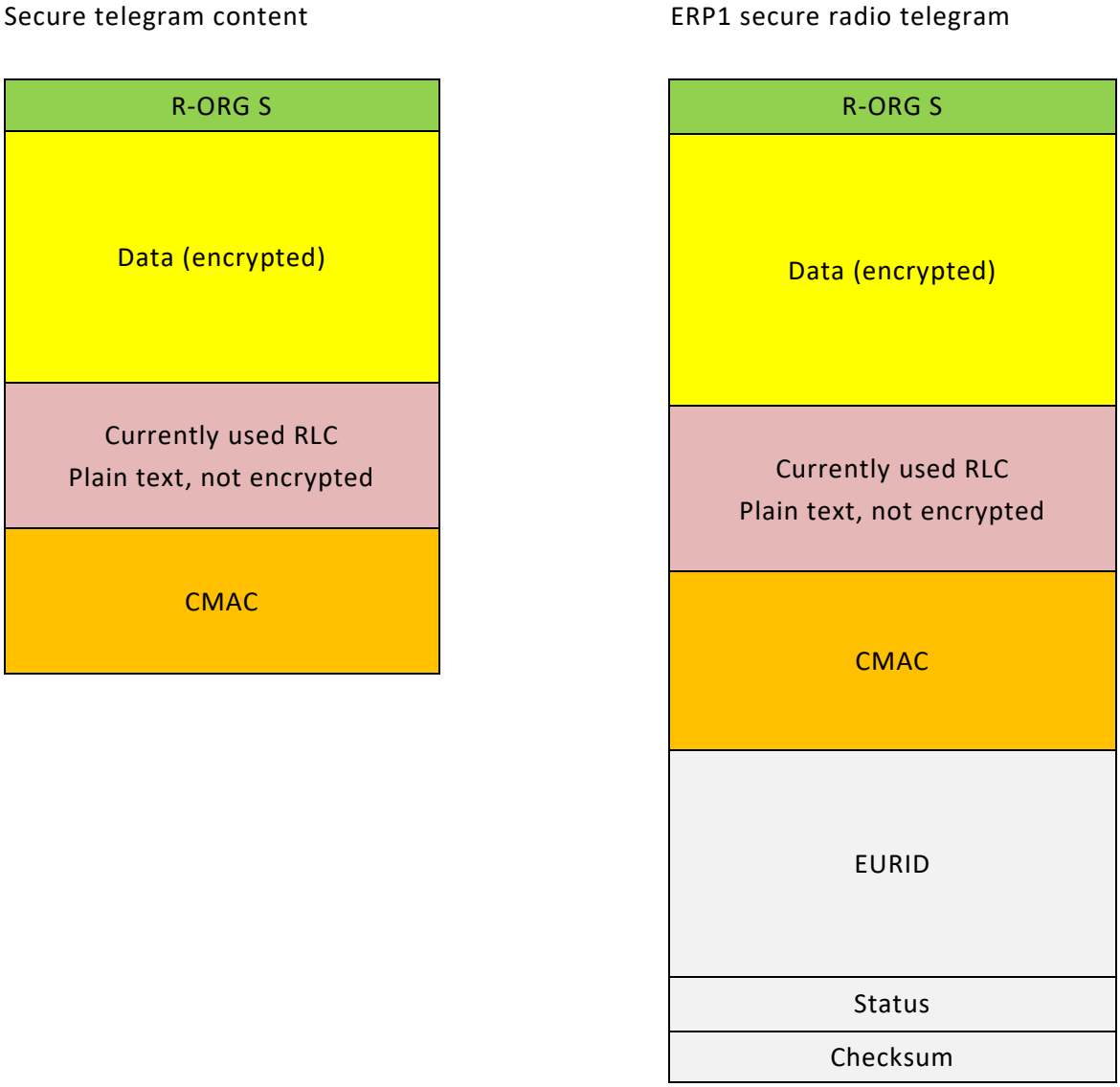

Figure 26. ERP1 secure radio telegram structure

In the most common version of the secure message, the RLC and CMAC fields are present. DATA is encrypted. The fields R-ORG S, DATA, RLC and CMAC are integrated without modification in the ERP1 telegram. The EURID and STATUS fields are not encrypted.

A.1.2 Secure Teach-in chaining with ERP1

The ERP1 limits the size telegrams to 21 bytes. Therefore, the security teach-in telegram identified by R-ORG-S 0x35: SEC-TI must be divided into two ERP1 chained messages. This is shown in **Figure 27** below.

First SEC-TI ERP1 telegram:

	7	6	5	4	3	2	1	0
0	R-ORG-S 0x35: SEC-TI							
1	0b00		0b10		Teach-In Info			
2	SLF							
3	Currently used RLC (3 or 4 bytes) Plain text, not encrypted							
…								
…								
5 or 6								
7 or 8	First part of PK (7 or 8 bytes)							
…								
14								
15	EURID							
16								
17								
18								
19	Status							
20	Checksum							

Second SEC-TI ERP1 telegram:

	7	6	5	4	3	2	1	0
0	R-ORG-S 0x35: SEC-TI							
1	0b01		0b00		0b0000			
2	Second part of PK (9 or 8 bytes)							
…								
9 or 10								
10 or 11	EURID							
11 or 12								
12 or 13								
13 or 14								
14 or 15	Status							
15 or 16	Checksum							

Figure 27. Format of the security teach-in telegram

Depending on the size of the RLC, the first part of the private key which is transmitted in the first SEC-TI telegram is either 7 or 8 bytes long. Accordingly, the second part of the private key which is transmitted in the second SEC-TI telegram is either 9 or 8 bytes long.

No time-out value is defined for the sequence of the two messages in this message chain. Newer messages with the same index will overwrite older messages with the same index. At the end of the learn mode, any partly received messages will be deleted.

A.2 ERP2

A.2.1 Secure data telegram format for ERP2

The following figure indicates the conversion of the secure message into ERP2 protocol. The secure message DATA field contains the information of the concatenated ERP2 telegram Data and Optional Data fields. The RLC and CMAC are placed after the Optional Data. The information of the R-ORG S is contained within the HEADER field. The EXTENDED HEADER depends on the telegram. See ERP2 Specification.

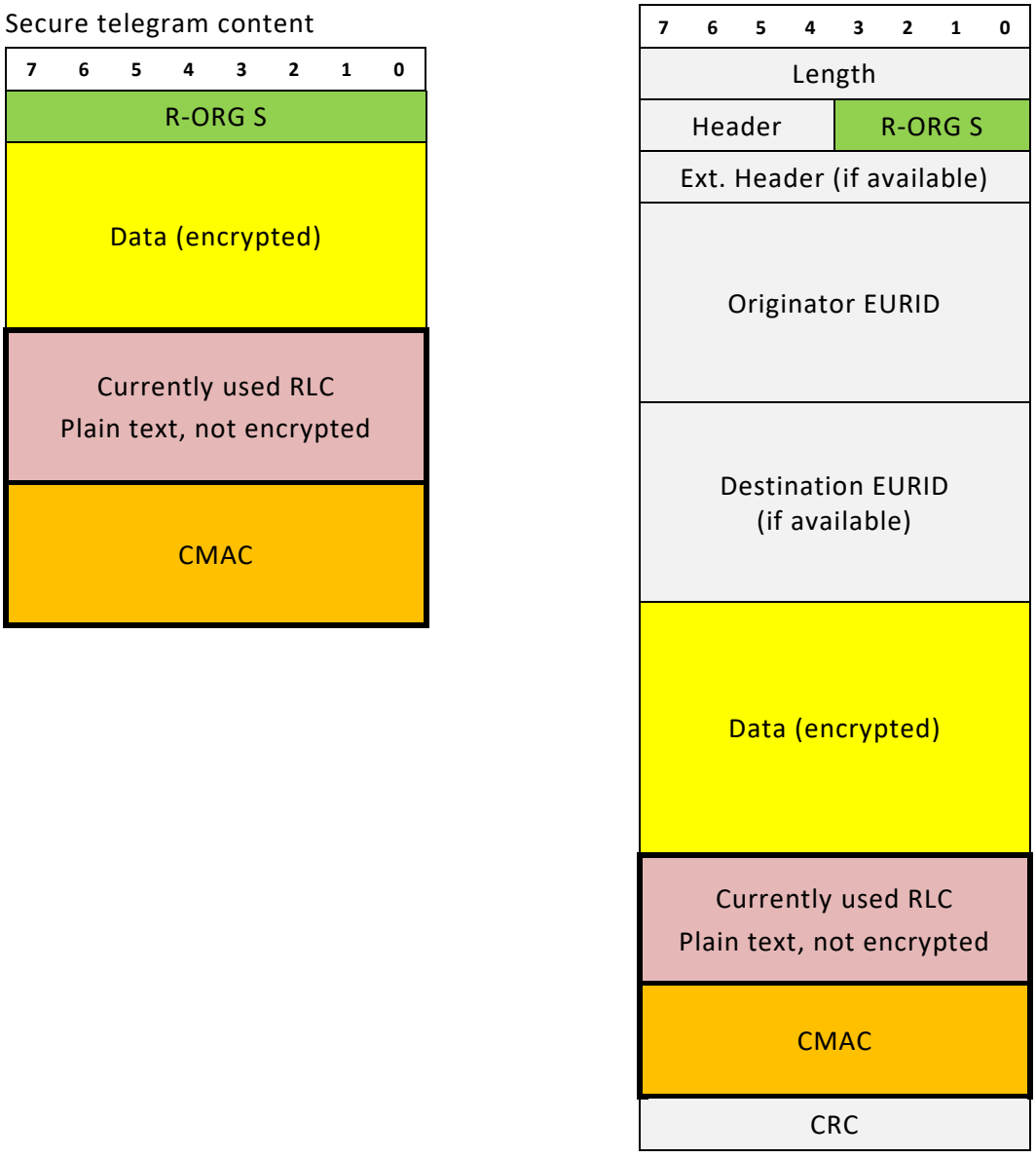

Figure 28. ERP2 secure radio telegram structure

A.2.2 Secure Teach-in with ERP2

A.2.2.1 Secure Teach-In unchained with ERP2

The following figure indicates one unchained SEC-TI telegram for ERP2 protocol:

SEC-TI ERP2 telegram

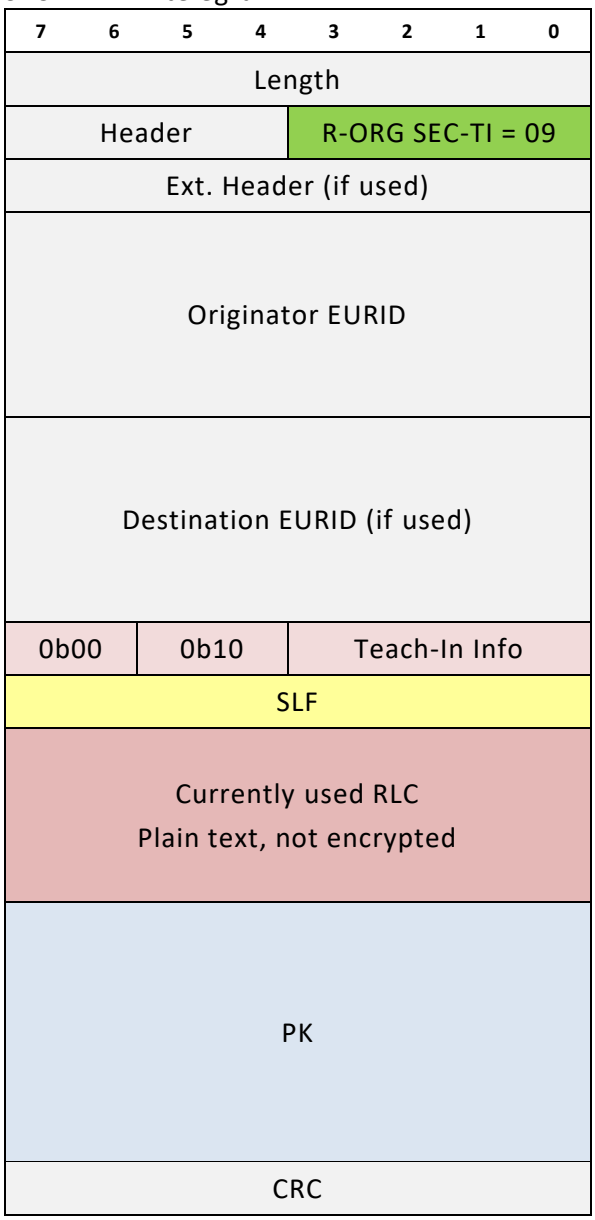

Figure 29. SEC-TI Message ERP2 telegram

The message fields TEACH_IN_INFO, SLF, RLC and KEY are concatenated in the order indicated and placed in the DATA field of the ERP2 telegram. The information of R-ORG TS is contained within the HEADER. The EXTENDED HEADER depends on the telegram.

A.2.2.2 Secure Teach-in chaining with ERP2

Many EnOcean applications have a very limited amount of energy available. For this reason, it is possible to divide the security teach-in telegram into a chain of two messages also for ERP2. This is shown in **Figure 30** below.

SLF, RLC and the first part of PK are transmitted in the first telegram. It recommended to set the length of the first part of PK in the same way as within an ERP1 telegram (9 or 8 bytes depending on RLC size) to allow the conversion from an ERP2 telegram to an ERP1 telegram where needed.

The second telegram contains the remaining part of the PK.

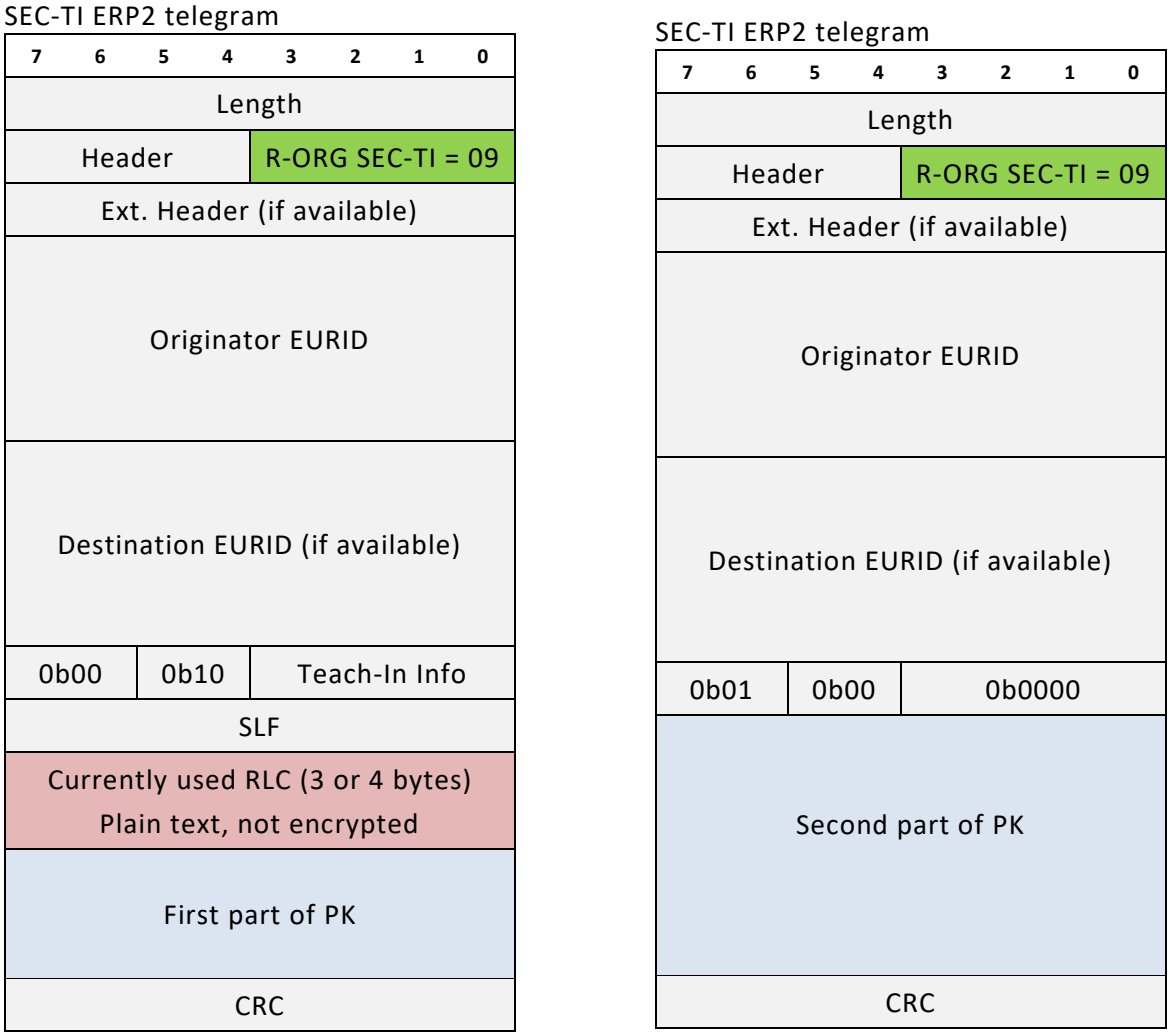

Figure 30. Security teach-in telegram chaining for ERP2

No time-out value is defined for the sequence of the two messages in this message chain. Newer messages with the same index will overwrite older messages with the same index. At the end of the learn mode, any partly received messages will be deleted.

A.3 PSK CRC8 checksum

A.3.1 Algorithm

The polynomial is $P(x) = x^8 + x^2 + x^1 + x^0$

code uint8 u8CRC8Table[256] = {

0x00, 0x07, 0x0e, 0x09, 0x1c, 0x1b, 0x12, 0x15,
0x38, 0x3f, 0x36, 0x31, 0x24, 0x23, 0x2a, 0x2d,
0x70, 0x77, 0x7e, 0x79, 0x6c, 0x6b, 0x62, 0x65,
0x48, 0x4f, 0x46, 0x41, 0x54, 0x53, 0x5a, 0x5d,
0xe0, 0xe7, 0xee, 0xe9, 0xfc, 0xfb, 0xf2, 0xf5,
0xd8, 0xdf, 0xd6, 0xd1, 0xc4, 0xc3, 0xca, 0xcd,
0x90, 0x97, 0x9e, 0x99, 0x8c, 0x8b, 0x82, 0x85,
0xa8, 0xaf, 0xa6, 0xa1, 0xb4, 0xb3, 0xba, 0xbd,
0xc7, 0xc0, 0xc9, 0xce, 0xdb, 0xdc, 0xd5, 0xd2,
0xff, 0xf8, 0xf1, 0xf6, 0xe3, 0xe4, 0xed, 0xea,
0xb7, 0xb0, 0xb9, 0xbe, 0xab, 0xac, 0xa5, 0xa2,
0x8f, 0x88, 0x81, 0x86, 0x93, 0x94, 0x9d, 0x9a,
0x27, 0x20, 0x29, 0x2e, 0x3b, 0x3c, 0x35, 0x32,
0x1f, 0x18, 0x11, 0x16, 0x03, 0x04, 0x0d, 0x0a,
0x57, 0x50, 0x59, 0x5e, 0x4b, 0x4c, 0x45, 0x42,
0x6f, 0x68, 0x61, 0x66, 0x73, 0x74, 0x7d, 0x7a,
0x89, 0x8e, 0x87, 0x80, 0x95, 0x92, 0x9b, 0x9c,
0xb1, 0xb6, 0xbf, 0xb8, 0xad, 0xaa, 0xa3, 0xa4,
0xf9, 0xfe, 0xf7, 0xf0, 0xe5, 0xe2, 0xeb, 0xec,
0xc1, 0xc6, 0xcf, 0xc8, 0xdd, 0xda, 0xd3, 0xd4,
0x69, 0x6e, 0x67, 0x60, 0x75, 0x72, 0x7b, 0x7c,
0x51, 0x56, 0x5f, 0x58, 0x4d, 0x4a, 0x43, 0x44,
0x19, 0x1e, 0x17, 0x10, 0x05, 0x02, 0x0b, 0x0c,
0x21, 0x26, 0x2f, 0x28, 0x3d, 0x3a, 0x33, 0x34,
0x4e, 0x49, 0x40, 0x47, 0x52, 0x55, 0x5c, 0x5b,
0x76, 0x71, 0x78, 0x7f, 0x6A, 0x6d, 0x64, 0x63,
0x3e, 0x39, 0x30, 0x37, 0x22, 0x25, 0x2c, 0x2b,
0x06, 0x01, 0x08, 0x0f, 0x1a, 0x1d, 0x14, 0x13,

0xae, 0xa9, 0xa0, 0xa7, 0xb2, 0xb5, 0xbc, 0xbb,
0x96, 0x91, 0x98, 0x9f, 0x8a, 0x8D, 0x84, 0x83,
0xde, 0xd9, 0xd0, 0xd7, 0xc2, 0xc5, 0xcc, 0xcb,
0xe6, 0xe1, 0xe8, 0xef, 0xfa, 0xfd, 0xf4, 0xf3

```
crc=0;
for (i=0; i<sizeof(key); i++)
{
   crc  = u8CRC8Table[crc ^ key[i]];
}
```

A.3.2 Test vector

PSK = 0x3410de8f1aba3eff9f5a117172eacabd

crc8 = 0x07

A.4 Security Test vectors

The following test vectors are useful to check the user implementation for secure applications.

A.4.1 Secure STM

The first example uses variable AES128 as encryption algorithm, 3 bytes CMAC and 3 bytes rolling code as security parameters. The secure message is with R-ORG encapsulation.

Teach-in telegram:

- Transmitted ERP1-Telegrams over the air:
    1. 35 20 AB C0 FF EE 45 6E 4F 63 65 61 6E 01 9E B6 3B 00
    2. 35 40 20 47 6D 62 48 2E 31 33 00 01 9E B6 3B 00
- Teach-Info : 0x00
- Security Layer format: 0xAB
  "SLF_DATA_ENC_VAES128 | SLF_MAC_3BYTE | SLF_RLC_ALGO_24BIT"
- Security Key: 0x45, 0x6E, 0x4F, 0x63, 0x65, 0x61, 0x6E,
  0x20, 0x47, 0x6D, 0x62, 0x48, 0x2E, 0x31, 0x33, 0x00
- Rolling code: 0xC0FFEE

Data telegram content:

- Transmitted ERP1-Telegram over the air: A5 08 27 FF 80 01 9E B6 3B 00
- R-ORG: 0xA5
- DATA: 0x08, 0x27, 0xFF, 0x80
- EURID: 01 9E B6 3B

Data encryption:

```
VAES init vector    0x3410DE8F1ABA3EFF9F5A117172EACABD
XOR
RLC padded          0xC0FFEE00000000000000000000000000
-----------------------------------------------------
Input               0xF4EF308F1ABA3EFF9F5A117172EACABD
AES128
Key                 0x456E4F6365616E20476D62482E313300
-----------------------------------------------------
AES128 encrypted    0x9BE2E35D5FE7858645200587DD3B515C
XOR
Data padded         0xA50827FF80000000000000000000000
-----------------------------------------------------
Data encrypted      0x3EEAC4A2DFE7858645200587DD3B515C
```

Subkey generation for CMAC calculation:

Key	0x456E4F6365616E20476D62482E313300
AES128	
Input	0x00000000000000000000000000000000
---	---
AES128 encrypted	0xDD21AA892DDF3CB967C314369A272338
<< 1	
---	---
Result after shift	0xBA4355125BBE7972CF86286D344E4670
XOR	
	0x00000000000000000000000000000087
---	---
K1=	**0xBA4355125BBE7972CF86286D344E46F7**
<< 1	
---	---
Result after shift	0x7486AA24B77CF2E59F0C50DA689C8DEE
XOR	
	0x00000000000000000000000000000087
---	---
K2=	**0x7486AA24B77CF2E59F0C50DA689C8D69**

CMAC calculation:

Message padded	0x313EEAC4A2DFC0FFEE80000000000000
XOR	
K2	0x7486AA24B77CF2E59F0C50DA689C8D69
---	---
Input	0x45B840E015A3321A718C50DA689C8D69
AES128	
Key	0x456E4F6365616E20476D62482E313300
---	---
CMAC(24 bits)	0xEAF20EED28679F641C15B10B9308D0D4

Resulting secure telegram

Payload with encapsulated R-ORG + Data + RLC + CMAC

- Transmitted ERP1-Telegram over the air:
  31 3E EA C4 A2 DF C0 FF EE EA F2 0E 01 9E B6 3B 00
- R-ORG: 0x31 (secure R-ORG with encrypted original R-ORG)
- DATA: 0x3E, 0xEA, 0xC4, 0xA2, 0xDF
- CMAC: 0xEA, 0xF2, 0x0E
- RLC: 0xC0, 0xFF, 0xEE

A.4.2 Secure PTM

This example uses variable AES128 encryption algorithm, 3 bytes CMAC and 3 bytes rolling code.

Security teach-in telegram:

- Transmitted ERP1 chained messages:

  Message 1: 35 24 4B 3E 2D 45 6E 4F 63 65 61 6E 01 85 E1 77 00

  Message 2: 35 40 20 47 6D 62 48 2E 31 33 00 01 85 E1 77 00

- Teach-in info: 0x04 "TEACH_INFO_PTM | TEACH_INFO_PTM_ROCKERA"
- Security Layer format = 0x8B
  "SLF_DATA_ENC_VAES128 | SLF_MAC_3BYTE | SLF_RLC_ALGO_24BIT"
- Security Key: 0x45, 0x6E, 0x4F, 0x63, 0x65, 0x61, 0x6E,
  0x20, 0x47, 0x6D, 0x62, 0x48, 0x2E, 0x31, 0x33, 0x00
- Rolling code: 0x3E2D00

Data telegram content:

- Transmitted ERP1-Telegram over the air: D2 09 01 85 E1 77 00
- R-ORG: 0xF6
- DATA: 0x09
- EURID: 01 85 E1 77

Data encryption:

```
VAES init vector    0x3410DE8F1ABA3EFF9F5A117172EACABD
XOR
RLC padded          0x3E2D00000000000000000000000000
-----------------------------------------------------
Input               0x0A3DDE8F1ABA3EFF9F5A117172EACABD
AES128
Key                 0x456E4F6365616E20476D62482E313300
-----------------------------------------------------
AES128 encrypted    0xC77F3257CA9F626AD8F6ED44DB04D26C
XOR
Data padded         0x09000000000000000000000000000000
-----------------------------------------------------
Data encrypted      0xCE7F3257CA9F626AD8F6ED44DB04D26C
AND
Mask (*)            0x0F
-----------------------------------------------------
Data encrypted      0x0E
```

(*) For PTM secure telegrams the secured data most significant byte is chopped.

CMAC calculation:

```
Key             0x456E4F6365616E20476D62482E313300
Subkey K2       0x7486AA24B77CF2E59F0C50DA689C8D69

Message padded  0x300E3E2D008000000000000000000000
XOR
K2              0x7486AA24B77CF2E59F0C50DA689C8D69
------------------------------------------------
Input           0x44889409B7FCF2E59F0C50DA689C8D69
AES128
Key             0x456E4F6365616E20476D62482E313300
------------------------------------------------
CMAC(24 bits)   0x05E56DD07701302C2DE4C6B8DEF0E508
```

Resulting secure telegram:

Payload with RORG + Payload + CMAC

- Transmitted ERP1-Telegram over the air: 30 0E 3E 2D 00 05 E5 6D 01 85 E1 77 00
- R-ORG: 0x30
- DATA: 0x0E
- CMAC: 0x05, 0xE5, 0x6D
- RLC: 0x3E, 0x2D, 0x00 not transmitted

Decrypted data telegram (after decryption by the receiver):

- Transmitted ERP1-Telegram over the air: 32 09 01 85 E1 77 00
- R-ORG: 0x32
- DATA: 0x09

A.4.3 Secure Chained Data

This example uses variable AES128 as encryption algorithm, 4 bytes CMAC and 4 bytes rolling code as security parameters.

- Security Layer format: 0xF3
- "SLF_DATA_ENC_VAES128 | SLF_MAC_4BYTE | SLF_RLC_ALGO_32BIT | SLF_RLC_TX_ON"
- Security Key: 0xE5, 0x08, 0x80, 0xCF, 0x67, 0x79, 0x0D, 0x5D, 0x66, 0xAA, 0x7F, 0x3B, 0x7A, 0xD7, 0x7A, 0x3F
- Rolling code: 0x01020304

Original telegram content (before security processing and chaining)

- R-ORG: 0xD1 (MSC)
- DATA: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D

Data encryption

1st part

```
VAES init vector    0x3410DE8F1ABA3EFF9F5A117172EACABD
XOR
RLC padded          0x01020304000000000000000000000000
-----------------------------------------------------
Input               0x3512DD8B1ABA3EFF9F5A117172EACABD
AES128
Key                 0xE50880CF67790D5D66AA7F3B7AD77A3F
-----------------------------------------------------
AES128 encrypted    0x6A17C07806CEF0515AEA013A24B97FAE
XOR
Data padded         0xD100010203040506070809 0A0B0C0D0E
-----------------------------------------------------
Data encrypted 1    0xBB17C17A05CAF5575DE208302FB572A0
```

2nd part

VAES init vector 0x3410DE8F1ABA3EFF9F5A117172EACABD

XOR

RLC padded 0x01020304000000000000000000000000

XOR

AES-Result 1st part 0x6A17C07806CEF0515AEA013A24B97FAE

Input 0x5F051DF31C74CEAEC5B0104B5653B513

AES128

Key 0xE50880CF67790D5D66AA7F3B7AD77A3F

AES128 encrypted 0xF22A5526B70483E715FE14D8166B673f

XOR

Data padded 0x0F101112131415161718191A1B1C1D

Data encrypted 2 0xFD3A4434A41096F102E60DC20D777A

Resulting secure telegram:

Payload with encapsulated RORG + Data + RLC + CMAC

- R-ORG-S: 0x31
- Data: BB 17 C1 7A 05 CA F5 57 5D E2 08 30 2F B5 72 A0 FD 3A 44 34 A4 10 96 F1 02 E6 0D C2 0D 77 7A 01 02 03 04 3B 4C 38 0F

Transmitted Telegrams over air (ERP1 example):

First ERP1 telegram:

	7	6	5	4	3	2	1	0
0	R-ORG SEC-CDM = 0x33							
1	SEQ		IDX = 0b000000					
2	LENGTH MSB: 0x00							
3	LENGTH LSB: 0x27 (39 bytes in total)							
4	ENCRYPTED R-ORG: 0xBB							
5	ENCRYPTED PAYLOAD: 0x17							
6	ENCRYPTED PAYLOAD: 0xC1							
7	ENCRYPTED PAYLOAD: 0x7A							
8	ENCRYPTED PAYLOAD: 0x05							
9	ENCRYPTED PAYLOAD: 0xCA							
10	ENCRYPTED PAYLOAD: 0xC5							
11	ENCRYPTED PAYLOAD: 0xF5							
12	ENCRYPTED PAYLOAD: 0x57							
13	ENCRYPTED PAYLOAD: 0x5D							
14	ENCRYPTED PAYLOAD: 0xE2							
15	EURID (4 bytes)							
16								
17								
18								
19	Status (1 byte)							
20	Checksum (1 byte)							

Second ERP1 telegram:

	7	6	5	4	3	2	1	0
0	R-ORG SEC-CDM = 0x33							
1	SEQ		IDX = 0b0000001					
2	ENCRYPTED PAYLOAD: 0x08							
3	ENCRYPTED PAYLOAD: 0x30							
4	ENCRYPTED PAYLOAD: 0x2F							
5	ENCRYPTED PAYLOAD: 0xB5							
6	ENCRYPTED PAYLOAD: 0x72							
7	ENCRYPTED PAYLOAD: 0xA0							
8	ENCRYPTED PAYLOAD: 0xFD							
9	ENCRYPTED PAYLOAD: 0x3A							
10	ENCRYPTED PAYLOAD: 0x44							
11	ENCRYPTED PAYLOAD: 0x34							
12	ENCRYPTED PAYLOAD: 0xA4							
13	ENCRYPTED PAYLOAD: 0x10							
14	ENCRYPTED PAYLOAD: 0x96							
15	EURID (4 bytes)							
16								
17								
18								
19	Status (1 byte)							
20	Checksum (1 byte)							

Figure 31. Chained secure telegrams with IDX=0 and IDX=1

Third ERP1 telegram:

	7	6	5	4	3	2	1	0
0	R-ORG SEC-CDM = 0x33							
1	SEQ		IDX = 0b000010					
2	ENCRYPTED PAYLOAD: 0xF1							
3	ENCRYPTED PAYLOAD: 0x02							
4	ENCRYPTED PAYLOAD: 0xE6							
5	ENCRYPTED PAYLOAD: 0x0D							
6	ENCRYPTED PAYLOAD: 0xC2							
7	ENCRYPTED PAYLOAD: 0x0D							
8	ENCRYPTED PAYLOAD: 0x77							
9	ENCRYPTED PAYLOAD: 0x7A							
10	RLC (Plain Text, MSB first): 0x01							
11	RLC (Plain Text, MSB first): 0x02							
12	RLC (Plain Text, MSB first): 0x03							
13	RLC (Plain Text, MSB first): 0x04							
14	CMAC: 0x3B							
15	EURID (4 bytes)							
16								
17								
18								
19	Status (1 byte)							
20	Checksum (1 byte)							

Fourth ERP1 telegram:

	7	6	5	4	3	2	1	0
0	R-ORG SEC-CDM = 0x33							
1	SEQ		IDX = 0b000011					
4	CMAC: 0x4C							
5	CMAC: 0x38							
6	CMAC: 0x0F							
15	EURID (4 bytes)							
16								
17								
18								
19	Status (1 byte)							
20	Checksum (1 byte)							

Figure 32. Chained secure telegrams with IDX=2 and IDX=3

A.5 Legacy security implementations

This chapter informs about SLF combinations, which were valid in legacy security implementations and are deprecated. Only the SLF combinations of Chapter 3.5 should be used.

7	6	5	4	3	2	1	0
RLC_TYPE[1]			**CMAC_TYPE**		**ENCRYPTION_TYPE**		
0b000: RLC not used 0b001: Reserved for future use 0b010: 16 Bit implicit RLC Roll Over/ Window Algorithm 0b011: 16 Bit explicit RLC Roll Over/ Window Algorithm 0b100: 24 Bit implicit Roll Over/ Window Algorithm 0b101: 24 Bit explicit RLC [24 Bit transmitted] No Roll Over / No Window Algorithm 0b110: 32 Bit explicit RLC [24 Bit transmitted] No Roll Over / No Window Algorithm 0b111: 32 Bit explicit RLC [32 Bit transmitted] No Roll Over / No Window Algorithm			0b00: CMAC not used 0b01: AES128 3 bytes 0b10: AES128 4 bytes 0b11: Reserved for future use		0b000: No data encryption 0b001: Reserved for future use 0b010: Reserved for future use 0b011: VAES – AES128 0b100 … 0b111: Reserved for future use		

Table 4. Legacy Security Level Format fields and its interpretation
Note (1): If RLC_TYPE = 0b001 or 0b010, then the secure Teach-in message contains the rolling code whose size corresponds to the one described by that bit field.

Field RLC_TYPE

This field defines the type of the rolling code algorithm used during secure communication calculation and can have the following values:

0b000: RLC is not used

0b001: Reserved for future use

0b010: 16 Bit implicit RLC which allows a roll over and uses a window algorithm at the receiver side to ensure that both devices are in sync.

0b011: 16 Bit explicit RLC which allows a roll over and uses a window.

0b100: 24 Bit implicit RLC which allows a roll over and uses a window algorithm at the receiver side to ensure that both devices are in sync.

---

[1] In older specification the RLC_ALGO field was split into RLC_ALGO + RLC_TX. The same functionality of the old specification is still possible, case 0,2,3,4,5. Only case 6 & 7 are actually new and defined a new behaviour

0b101: 24 bit explicit RLC, the RLC will be transmitted. Roll over is not allowed and no window algorithm is used

0b110: 32 bit explicit RLC only 24 bit of it will be transmitted. Roll over is not allowed and no window algorithm is used

0b111: 32 bit explicit RLC, the full RLC will be transmitted. Roll over is not allowed and no window algorithm is used

Field CMAC_TYPE

This field defines the type of algorithm used for the CMAC calculation and can have the following values:

0b00: CMAC is not used

0b01: CMAC is a 3-byte-long code. MAC is calculated using AES128.

0b10: CMAC is a 4-byte-long code. MAC is calculated using AES128.

0b11: Reserved for future use

Field ENCRYPTION_TYPE

Defines the type of algorithm used for the encryption of data telegrams calculation and can have the following values:

0b000: DATA is not encrypted

0b001, 0b010: Reserved for future use

0b011: DATA will be encrypted/decrypted XORing with a string obtained from an AES128. See Chapter 6.1

0b100 … 0b111: Reserved for future use