Adding Security to EnOcean Receivers
NOTES FOR SECURITY IN DIFFERENT RECEIVER APPLICATION

Table of contents

1. INTRODUCTION

This document describes how to include security into line powered applications, primarily applications such as Gateways and Actuators.

Before reading this document you should be familiar with the "Security EnOcean for Radio Networks" specification [1]. You can also find a good summary of this in the App Note 509.

1.1. Definitions

Term / Abbr.	Description
µC	Microcontroller (external)
AES	Advanced Encryption Standard
API	Application Programming Interface
APP	Application
ASK	Amplitude Shift Keying
CBC	Cipher Block Chaining
CMAC	Cipher Based Message Authentication Code
CRC	Cyclic Redundancy Codes
DATA	Payload of a radio telegram
Device	Customer end-device with an integrated EnOcean radio module
EEP	EnOcean Equipment Profile
EHW	Energy Harvested Wireless protocol
ERP	EnOcean Radio Protocol (ERP1 = Version 1, ERP2 = Version 2)
ESP3	EnOcean Serial Protocol V3
FSK	Frequency Shift Keying
Gateway	Module with a bidirectional serial communication connected to a HOST
GP	Generic Profiles
ID	Unique module identification number
KEY	Specific parameter used to encrypt / decrypt / transform DATA
MAC	Message Authentication Code
MSB	Most Significant Byte
PSK	Pre-shared Key
PTM	Pushbutton Transmitter Module
RLC	Rolling Code
R-ORG	Message parameter identifying the message type
SLF	Security Level Format specifying which security parameters are used
TXID	ID of a transmitter
VAES	Variable AES

1.2. References

[1] Security of EnOcean radio networks (System Specification) - http://www.enocean.com/en/security-specification/

[2] http://www.kotfu.net/2011/08/what-does-it-take-to-hack-aes/

[3] EEP Specification - http://www.enocean-alliance.org/eep/

[4] GP Specification - http://www.enocean-alliance.org/

[5] EnOcean Radio Protocol 1 - http://www.enocean.com/fileadmin/redaktion/pdf/tec_docs/EnOceanRadioProtocol.pdf

[6] Smart Acknowledge -
http://www.enocean.com/fileadmin/redaktion/pdf/tec_docs/SmartAcknowledgement.pdf

[7] Remote Management -
http://www.enocean.com/fileadmin/redaktion/pdf/tec_docs/RemoteManagement.pdf

[8] Gateway Controller -
http://www.enocean.com/en/enocean-software/gateway-controller/

[9] Dolphin V4 Gateway Controller
http://www.enocean.com/en/enocean-software/

[10] EnOcean Link
http://www.enocean.com/en/enocean-software/enocean-link/

[11] EnOcean Link Gateway example:
http://www.enocean.com/fileadmin/redaktion/support/enocean-link/_gateway_example_8cpp-example.html

[12] Decoding Gateway
http://www.enocean.com/en/enocean-software/decoding-gateway-controller/

[13] DolphinAPI
http://www.enocean.com/en/download/

[14] http://www.enocean.com/en/enocean-software/

1.3. Revision History

No	Major Changes
1.0.	First version

2. SECURITY IN LINE-POWERED ENDDEVICES

When we consider line powered applications we traditionally refer to them as receivers. This convention is based on the very basic use case: sensor sending data to an actuator. But over time the features of EnOcean networks and devices have expanded so that today most line powered devices are actually transceivers. In this document we will focus on security features in line powered device taking the bidirectional aspect also into consideration.

A receiver application can consist of one or more processing units. The most common receiving applications are:

- Stand alone - the EnOcean module handles all application tasks. For example an HVAC actuator.
- Dual processing units – the EnOcean Module is used as transparent gateway and there is an external CPU which implements all application relevant functions. For example a Smart Home box.
- Distributed - the EnOcean Module is also used as transparent gateway but the external CPU only forwards the telegram further. It can possibly translate it or tunnel it by a different protocol. For example a protocol gateway.

Figure 1 illustrates this classification. Here we defined typical possible use cases using security:

- TCM 310 (C/U) or TCM 410J with external CPU, running EnOcean Link
- TCM 310 (C/U) or TCM 410J with external CPU with security functions implemented by the customer
- TCM 315 with external CPU
- TCM 415J with external CPU
- A stand alone EnOcean Module with security implementation using DolphinAPI by the customer

We will focus on each use case and how security features are implemented in these applications.

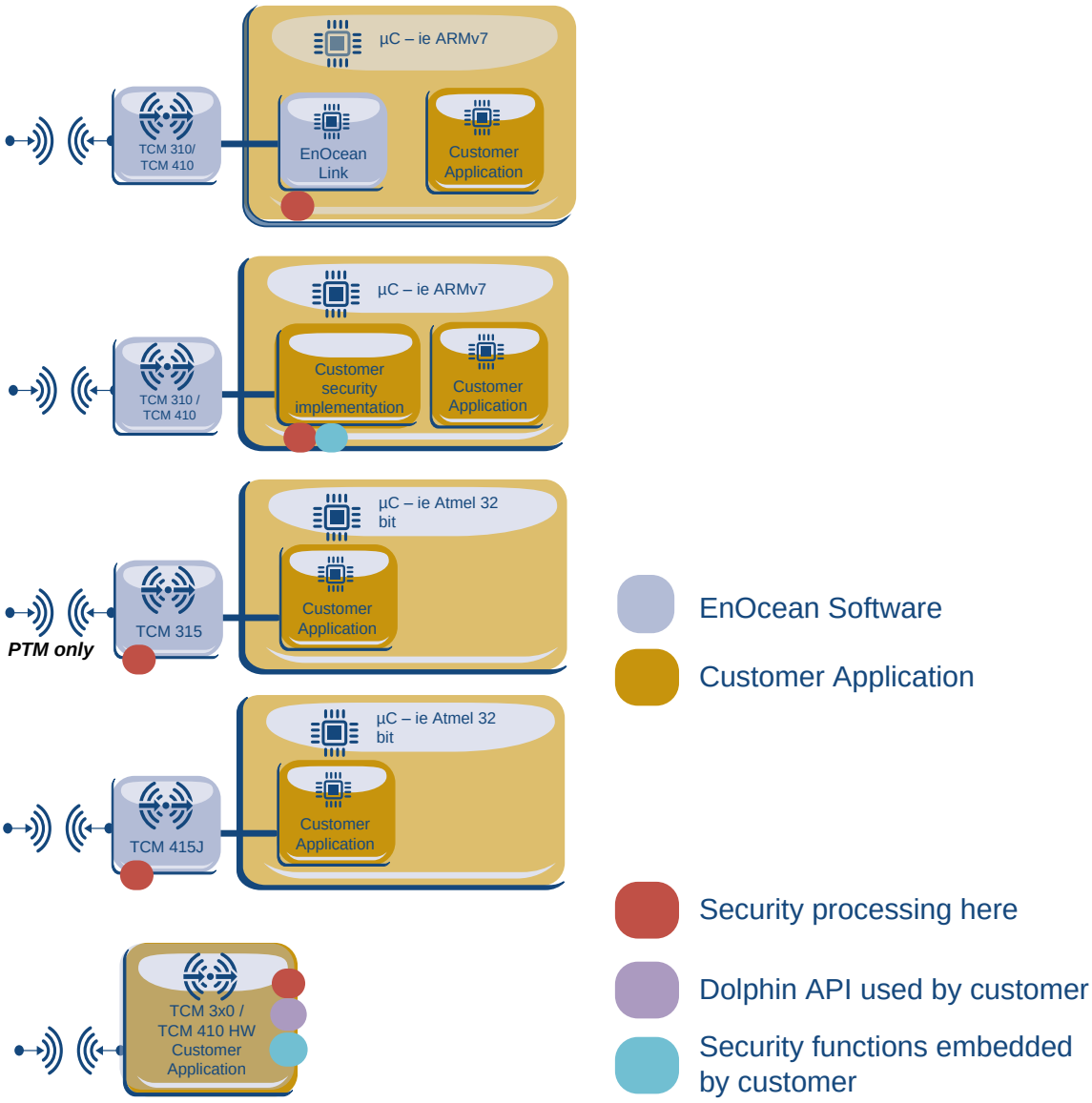

Figure 1 Security in receiver applications

Line powered devices are considered to have unlimited energy source and large computing capacities. In the most common field scenario there are many autarkic devices – sensors, but only one or a few line powered devices – actuators. Therefore the line-powered devices are predesigned to be able to receive and process telegrams from more devices at once.

For every processed device the receiver has to know its security parameters. SLF, KEY and RLC. This information together builds a – *security profile*. The security profile must be stored for every incoming device.

2.1. Common task of a secure line powered device

In this chapter we will focus on the additional tasks compared to a non-secure device. The common tasks of a line-powered device are primary:

- Store and maintain security profiles.
  This means that the device has to have a non-volatile memory which can be written with high periodicity – as the RLC will change. The alternative is not to store every change of the RLC but only every 30th or 50th for example. This will get the write cycles down and enabling memories with lower endurance. On the other hand it can result in an application instability, because if the device, for example, encounters an unplanned shut down, it can lose the synchronisation with the RLCs.
- Process Security Teach-In.
  Most actuators or Smart Home boxes have a Teach-In mode. This mode is used for the profile teach-in. For every receiver with security capabilities this teach-in must be extended to accept and process security teach-in telegrams. Devices which have no teach-in mode must incorporate security teach-in processing in the backbone or somewhere in the processing architecture.
- Process the Telegrams with security features.
  The essential task of a receiver is to be able to apply the defined security features. The amount of security features and specific attributes can vary. The definition of what specific features a transmitting device is using is defined in the SLF format at secure teach-in [1]. A receiver therefore should impermanent all defined features and its parameters so it can be interoperable. Exceptions are specific and dedicated receivers which have predefined communications partners.
- Send out teach-in telegrams and communication protected by security features.
  This task is required for secure bidirectional communication. To establish a bidirectional "secure link" both devices need to send a secure teach-in and thus manage each others security profile. A line-powered device needs to have a separate security profile for every outgoing secure communication.
- Resynchronisation possibility.
  A receiver must have the possibility to resynchronize transmitters RLC. For example if a receiver is powered down for longer time, but the transmitter keep on transmitting and incrementing the RLC the devices may become automatically desynchronised. The receiver will recall at start-up the last stored RLC, but the transmitter can be meanwhile much further outside the possible RLC window. In this case a resynchronisation of the devices will be required. Please see details on resynchronisation in the specification [1]. A resynchronisation might become also necessary also when the transmitter is outside of the receivers range or the channel is permanently blocked.

In the next chapter we will describe the security applications and how they can cope with these tasks.

3. IMPLEMENTATION OF SECURITY IN LINE-POWERED DEVICES

In this chapter we will describe each scenario shown in Figure 1 also considering the tasks mentioned in chapter 2.1. In general security functionality implementation can be:

- In EnOcean Modules - For "small" applications with dedicated long-term use case, e.g. HVAC control
- In external µC – for application with brought use and dynamic environment, e.g. smart home

When deciding where to put security implementation, one has to also consider also the computing effort required to perform the security features. For example, on the EnOcean Dolphin Platform with DolphinAPI one ASE128 computing cycle takes 0.25 ms – 1 ms. In this case having a rolling code window of 100 then validation a message can take in worst case 25 – 100 ms. Other external CPUs may have better performance, so please also consider what implementation you use for this process.

3.1. Secure application with transparent gateway

This application scenario consists of an EnOcean Gateway and an external controller. The typical use cases are for example set-up boxes, TVs and Smart Home boxes. The EnOcean Gateway is a transparent modem which forwards all telegrams received without any change to the serial interface and sends out all serial requests to the air. This use case is shown in Figure 2.

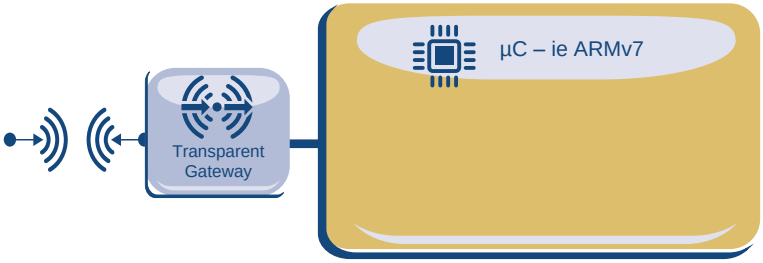

Figure 2 Security with transparent gateway

For the EnOcean transparent gateway you can use for example:

- Gateway Controller Firmware [8] – by default on: TCM 310 (C/U), USB 300 (C/U) or flash it into TCM 300 (C/U)
- Dolphin V4 Gateway Controller – by default on: TCM410J and USB 400J

In this case the security features are implemented on the external µC. The EnOcean transparent gateway forwards all messages unchanged to the µC. Security features only affect only the message payload which is not affected by the translation between serial and radio protocol, therefore a seamless bidirectional translation between radio and serial protocol also with security features is possible. So if a message is protected by security features this message will keep these features also on the UART interface. For illustration of this please see Figure 3.

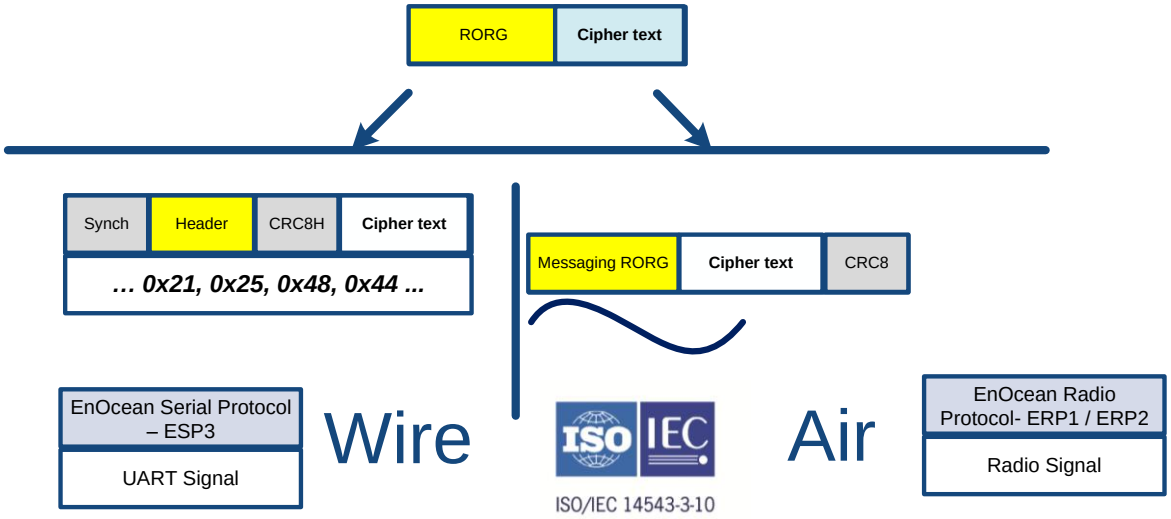

Figure 3 Seamless translation between serial and radio protocol

Processing of security features is possible also in external controller. In the next two chapters we will focus on how to implement them.

3.1.1. Using EnOcean Link

EnOcean Link is the middleware for the EnOcean Protocol stack. It is a library delivered also as source code. Its features provide and handle all tasks required to embed EnOcean into an application. By embedding EnOcean Link on a µC you gain a very powerful tool which handles besides the tasks of security processing also other EnOcean Protocol stack implementations. For further EnOcean Link information please see reference [10]. For the use case visualization with an EnOcean transparent gateway please see Figure 4.

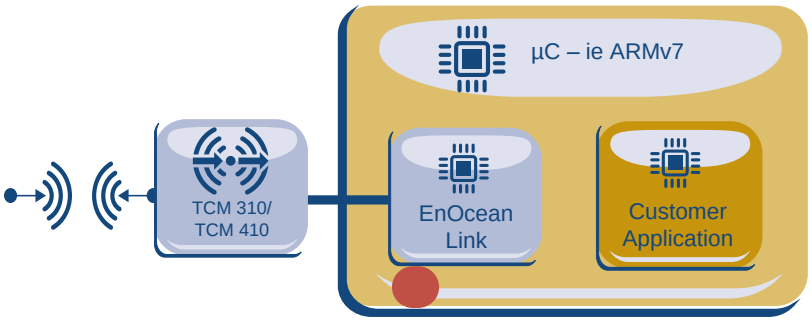

Figure 4 Security implementation with EnOcean Link and gateway

Security is an essential part of EnOcean Link so just by embedding EnOcean Link you will automatically receive all security features functionality required, also bidirectional. There are no additional tasks required. EnOcean link will also handle all tasks regarding storage and RLC update (see Class `eoDevice [11]`). An example can be reviewed in the EnOcean Link user manual, look for Gateway example [11].

Here is important to notice, that Security Teach-in and profile teach will be two separate actions (separate telegrams) and the function `gateway.Receive()` will return at security teach the flag `RECV_SECTEACHIN` and at profile teach `RECV_PROFILE`. The application does

not have to perform any explicit action at security teach-in. For details on the returned flag information please see the link [11] or [10].

For the application use case of a transparent gateway and µC we recommend to use EnOcean Link as the preferred solution.

3.1.2. Implementing own security solution and functions

If in an application the use of EnOcean Link is not applicable you can use the fall back solution and implement the security features by yourself. This means you have to implement all features described in the specification [1] in your µC environment. In general it means:

- Implement AES 128 encryption and decryption algorithm
- Implement VAES encryption and decryption
- Implement AES-CBC encryption and decryption
- Implement CMAC validation
- Implement Secure teach-in parsing
- Implement storage of security profiles- SLC RLC and KEYs

The VEAS, CMAC and AES-CBC tasks require a common AES 128 implementation. Depending on your platform and development environment you can maybe find a reference implementation or source code / library in the Internet.

3.2. Secure application with decoding controller

This application also consists of an EnOcean Gateway and an external µC, but compared to the use case in 0, the EnOcean Gateway handles the security tasks. The typical use cases are dedicated actuators and controllers, such as light actuators or HVAC controllers. Here is assumed that the external µC has comparable computing capacities and features as the EnOcean module. Otherwise it is to consider placing the security functions into the external µC. Please see visualization of this case in Figure 5.

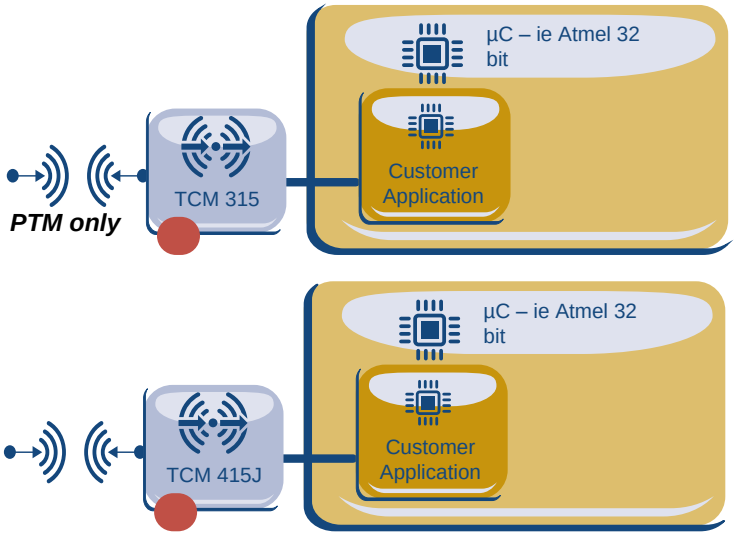

Figure 5 Security with decoding gateway

In this application the µC only controls the LRN mode of the decoding gateway. Storage of the security profiles and all security tasks are handled in the decoding gateway. It is rec-

ommended to connect an external EEPROM through I^2C to the decoding gateway so it stores the security profiles outside of the EnOcean module.

You can find exact usage description and details in the User Manuals of the decoding Gateway[12]. The TCM 315 is currently capable to decode only PTM 210 telegrams, for feature requests please contact us (support@enocean.com). TCM 415J is capable of bidirectional communication and processing all possible security features.

3.3. Custom security solution on EnOcean Modules with Dolphin API

This application is constrained in resources and features compared to applications in chapter 3.2 and 3.3 but it can still offer a valid baseline for many smaller or specific dedicated applications.

In general here all aspects of security implementation as described in chapter 3.1.2 must be considered, but without the actual security functionality implementation. These tasks are handled by the DolphinAPI. The DolphinAPI offers these security implementation functions:

```
sec_convertToNonsecure
sec_convertToSecure
sec_createTeachIn
sec_parseTeachIn
```

For details and usage please refer to the DolphinAPI user manual. There you can find examples and implementation references [13].

These implementation tasks remain:

- Implement storage of security profiles- SLC RLC and KEYs
- Implement application logic.

For the storage of security profiles we recommend to use an external memory (e.g. EEPROM). A receiver must be able to update and store RLC of all receiving devices. Storage of every change of every RLC into non-volatile memory will result in high amount of write cycles required, which can only be handled by an external memory module. Not storing every transmission but for example only every 30^{th} or 50^{th} will ease the situation and allow the implementation of storage memories with lower endurance e.g. also in the Dolphin Module.

Please consider this example of storing RLCs in Dolphin Module flash memory with a pure receiver application (no secure transmission):

Datasheet:

- Dolphin Flash Page – 20 000 Erase cycles per page (minimum specified endurance)
- Flash page size – 256 b
- Erasing current – typ. 20 mA
- Erasing time – typ. 20 ms

Storing keys:

- 2 – Flash Pages for keys (one key 16 bytes) and device IDs (one ID 4 bytes). 16 + 4 = **20**
- 2*256 / 20 = **24 possible keys / devices**

Storing Rolling Code:

- 2 – Flash Pages for Rolling Code
- 128 storing of Rolling Codes before Erasing one page (one byte RLC, one byte index)
  Note: A page can have up 256 write operations before an erase is required. But always a different byte cell must be written. Rewriting a byte cell which was already written will result in a failure.
  Here a suitable strategy of storing the RLC must be chosen. The RLC is typically 2 or 3 bytes long, but only one byte is changing, therefore we do only a pre-calculation based on the assumption, that only one byte is written per write cycle. With this byte an index of the device is written too. Together two bytes are written.
- Storing Rolling Code every 30th received Telegram
- 128 * 2 * 30 * 20 000 / 24 = 6 400 000 Telegrams per Device (minimum, the endurance can be much higher)
- STM330 – sends 96 Telegrams per day
- 6 400 000 / 96 = **187 years of operation**
  Note: By choosing more effective RLC storing strategy you can increase this number or optimize for an specific use case.
  Please consider that erase operations require 20 ms and 20 mA additional current. It is crucial that the device does not have a brown-out during flash erase operation. It was reported that also other pages can be damaged in this case. Please ensure that your device does not lose power supply in case of erase operations (e.g. connect an capacitator as buffer). Consider to turn of radio during erase operation to lower overall energy consumption.

Having a storing cycle higher than the rolling code window is not recommended. If an unplanned power-down of the receiver will occur then at power-up the receiver will recover the last stored RLC. The recovered RLC can be far less than the actual RLC of the transmitter. If the storing cycle is bigger than the RLC window itself, then the receiver and transmitter may become desynchronised, although the device powered-up immediately after power-down. This will result in repeated resynchronisation which require in most cases users involvement. Therefore choose the RLC window and storage window precisely.