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

1. INTRODUCTION

This document describes how to add security to line powered applications, primarily gateways and actuators.

Before reading this document you should be familiar with the specification "Security of EnOcean Radio Networks" [1]. You can also find a good summary of this in application 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
1.1.	Language Corrections

2. SECURE LINE-POWERED END DEVICES

When we consider line powered applications we traditionally refer to them as receivers. This convention is based on the very basic use case of a 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. This document focuses on security features in line-powered devices, also taking bidirectional communication in to consideration.

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

- Stand-alone units - the EnOcean module handles all application tasks. For example an HVAC actuator.
- Dual processing units – the EnOcean Module is used as a transparent gateway and there is an external CPU which implements all application relevant functions. For example a Smart-Home box.
- Distributed Systems- the EnOcean Module is also used as a transparent gateway but the external CPU only forwards the telegram further. The telegrams may be translated or tunnelled by a different protocol. For example a protocol gateway.

Figure 1 illustrates the different application types with security:

- TCM 310 (C/U) or TCM 410J with an external CPU running EnOcean Link
- TCM 310 (C/U) or TCM 410J with and external CPU and security functions implemented by the customer
- TCM 315 with external CPU
- TCM 415J with external CPU
- A stand-alone EnOcean Module with security features implemented using the Dolphin API by a customer

In this application note, it will be demonstrated how security features are implemented in each of these use cases.

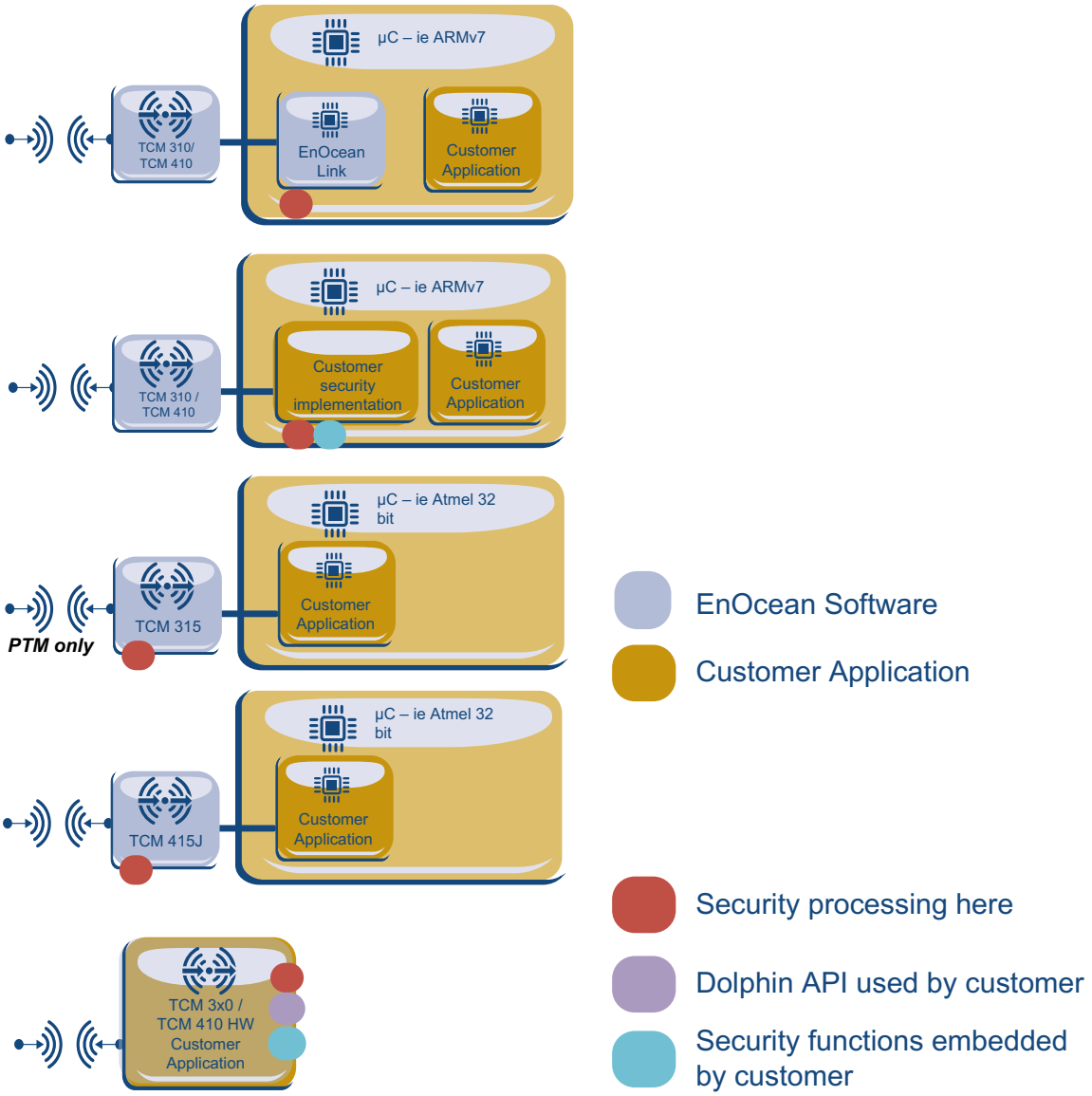

Figure 1 Security in receiver applications

Line powered devices are considered to have unlimited energy and large computing capabilities. In the most common scenario, there are many energy autarkic devices (sensors), but only one or two line powered devices (actuators). Therefore, line-powered devices are designed to be able to receive and process telegrams from more than one device at once.

The receiver has to know the security parameters of all the devices that send telegrams to it. These are the security level format (SLF), security key (KEY) and rolling code (RLC). This information combined, builds a *security profile*. The security profile of every incoming device must be stored.

2.1. Common tasks of a secure line powered device

In this chapter, the additional tasks of a secure device, compared to a non-secure device, are discussed. These are as follows:

- **Store and maintain security profiles.**
  This means that the device needs non-volatile memory with a large number of write cycles, as the RLC will change frequently. The alternative is not to store every change of the RLC but only every 30th or 50th for example. This will reduce the number of write cycles, enabling memories with lower endurance to be used. On the other hand it can result in application instability. If the device loses power unexpectedly, it can lose synchronisation with the RLCs of the incoming devices.
- **Process Security teach-In messages.**
  Most actuators or Smart Home boxes have a Teach-In mode. This mode is used to logically link sensors and actuators. If a receiver supports security, this teach-in process must be extended to accept and process secure teach-in telegrams. Devices which do not have a teach-in mode must incorporate security teach-in processing in the backbone or somewhere else in the system architecture.
- **Process telegrams with security features.**
  The essential task of a receiver is to be able to apply the defined security features. The security features supported by a device can vary. The definition of what specific features a transmitting device uses is defined in the SLF format during the secure teach-in process [1]. A receiver should support all security features to be interoperable. An exception to this can be made for receivers with predefined communication partners which do not use all security features.
- **Transmit secure teach-in and data telegrams.**
  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 other's security profile. A line-powered device needs a separate security profile for every outgoing secure communication.
- **Resynchronisation.**
  A receiver must have the possibility to resynchronize itself with the rolling code of incoming devices. For example, if a receiver is powered down for a long time, but the transmitter continues transmitting and incrementing its RLC, the devices may become unsynchronised. At start-up the receiver will load the last stored RLC, but the RLC transmitted may have been incremented outside of the RLC window. In this case the devices need to be resynchronized. Please see details on resynchronisation in the specification [1]. A loss of synchronization may also be caused if the transmitter is out of range of the receiver or if the channel becomes blocked.

3. IMPLEMENTATION OF SECURITY IN LINE-POWERED DEVICES

In this chapter, it is described how the tasks discussed in chapter 2.1 can be implemented for each scenario, shown in Figure 1. In general, security functionality can be implemented:

- In EnOcean Modules - For "small" applications with dedicated long-term use cases, e.g. HVAC control
- In external µCs – for applications in dynamic environments, e.g. smart home

When deciding where to implement the security functionality, one has to also consider the computing effort required to perform the security features. For example, on the EnOcean Dolphin Platform with Dolphin API, one ASE128 computing cycle takes 0.25–1ms. In this case with a rolling code window of 100, the validation of a message can take in worst case 25–100ms. Other external CPUs may have better performance, so please also consider how this process is implemented.

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 radio interface. This use case is shown in Figure 2.

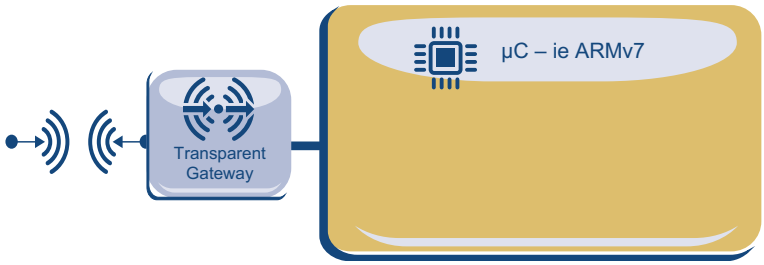

Figure 2 Security with transparent gateway

For the EnOcean transparent gateway you can use for example:

- Gateway Controller Firmware [8] – The default firmware for TCM 310 (C/U) and USB 300 (C/U). It can also be flashed to TCM 300 (C/U)
- Dolphin V4 Gateway Controller – The default firmware for 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 the message payload which is not changed by the translation from the enocean radio to the enocean serial protocol. Therefore a seamless bidirectional translation between radio and serial protocol including security features is also possible. If a message is encrypted, the encryption will remain when it has been translated from a radio to a serial message. For illustration of this please see Figure 3.

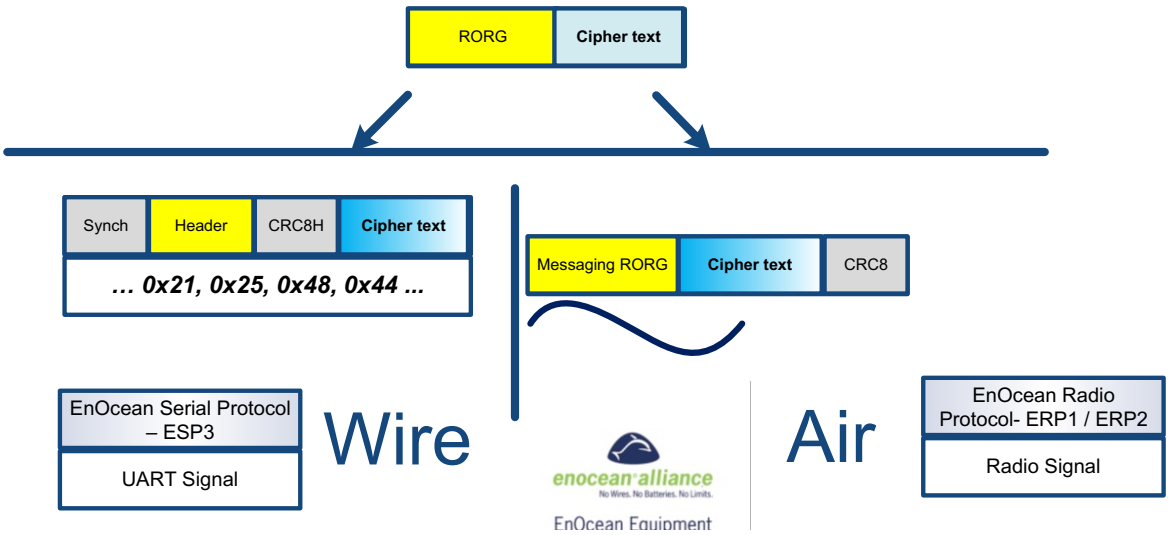

Figure 3 Seamless translation between serial and radio protocol

The pprocessing of security features is also possible in an external controller. The next two chapters focus on their implementation.

3.1.1. Using EnOcean Link

EnOcean Link is middleware for the EnOcean Protocol stack. It is a library delivered along with 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 the processing of the entire EnOcean protocol stack, including the security features. For further EnOcean Link information please see reference [10]. For the use case visualization with an EnOcean transparent gateway please see Figure 4.

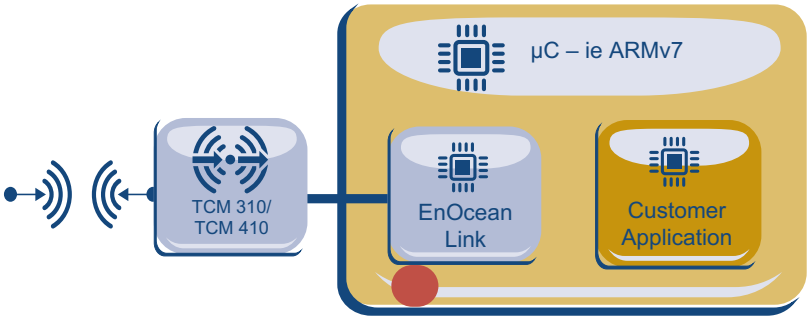

Figure 4. A Security implementation with EnOcean Link and an EnOcean gateway

Security is an essential part of EnOcean Link. By embedding EnOcean Link you will automatically receive all security functionality for both unidirectional and bidirectional communication. There are no additional tasks required. EnOcean link will also handle all tasks regarding storage and updating of RLCs (see Class `eoDevice` [11]). An example can be reviewed in the EnOcean Link user manual, for this, see Gateway example [11].

The security teach-in process and profile teach-in process are separate actions (separate telegrams). The EoLink function `gateway.Receive()` will return the flag RECV_SECTEACHIN when a security teach-in message is received and the flag RECV_PROFILE when a profile teach-in is received. The application does not have to perform any explicit action at security teach-in. For details on the returned flag information please see links [11] and [10].

For the use case of a transparent gateway and external µC, we recommend using EnOcean Link as the best solution.

3.1.2. Implementing own security solution and functions

If the use of EnOcean Link is not applicable, you can implement the security features yourself. This means that you have to implement all features described in the specification [1] on your µC. The following must be implemented:

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

The VEAS, CMAC and AES-CBC tasks require a common AES 128 implementation. Depending on microcontroller platform and development environment it may be possible find a reference implementation or source code on 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 it is assumed that the external µC has comparable computing capacities and features as the EnOcean module. Otherwise it may make more sense to implement the security functions into the external µC. Please see visualization of this case in Figure 5.

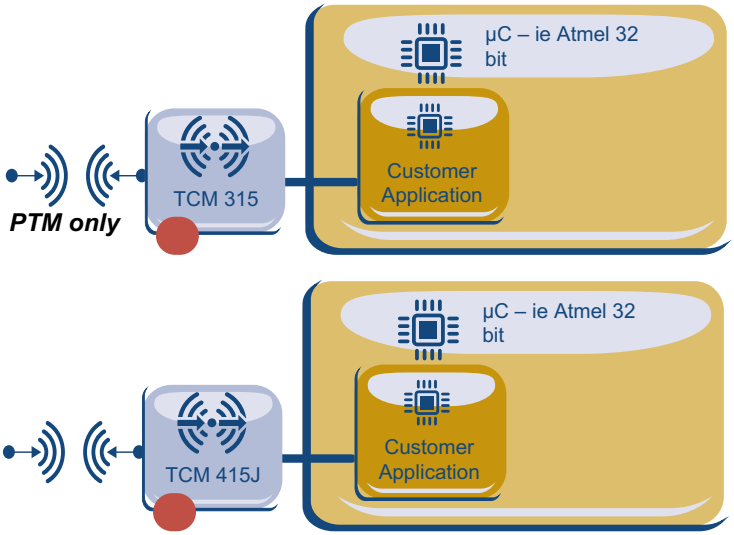

Figure 5 Security with decoding gateway

In this application, the µC only controls the learn mode of the decoding gateway. Storage of the security profiles and all security tasks are handled in the decoding gateway. It is recommended to connect an external EEPROM through I^2C to the decoding gateway to store the security profiles outside of the EnOcean module.

You can find as more detailed description of this in the User Manual 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). The TCM 415J is capable of bidirectional communication and processing of 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 small or specific 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 Dolphin API user manual [13]. There you can find examples and implementation references.

The following tasks remain:

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

For the storage of security profiles, we recommend the use of external memory (e.g. EEPROM). A receiver must be able to update and store the RLC of all incoming devices. Storage of every change of every RLC requires memory which can endure a high number of

write cycles. The number of write cycles can be reduced by only storing every 30th or 50th change of the RLC and allows memories with lower endurance to be used. E.g. The RLCs may be stored in the flash memory of the Dolphin chip.

Please consider the example application using a receiver only (no secure transmission): and storing RLCs in Dolphin Module flash memory

Datasheet:

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

Storing keys:

- 2 flash pages to store keys and Device IDs: A key is 16 bytes and a device ID is 4 bytes long. Total = 20 bytes each.
- 2*256 / 20 = **24 possible keys / devices**

Storing Rolling Code:

- 2 flash pages for storage of the rolling code
- 128 writes of a rolling code and index can be written before erasing one page (one byte of RLC stored, one byte index).
  Note: A page can have up 256 write operations before an erase is required. But the same byte of a flash page cannot be written twice without first erasing the whole page. Rewriting a byte 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 a more effective RLC storage strategy this number can be increased or optimized for a specific use case.
  Please consider that erase operations require 20ms and 20 mA of current. It is crucial that the device does not lose power during a flash erase operation. It has been reported that other pages can also be damaged in this case. Please ensure that your device does not lose power during erase operations (e.g. connect a capacitator as an energy buffer). Consider turning off the radio during erase operations to lower the 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, even if the device powered-up immediately after losing power. This will result in repeated resynchronisation which requires, in most cases, some action from the user. Choose the RLC window and storage window carefully.