Software Design Specification

Z-Wave Application Security Layer

Document No.:	SDS10865
Version:	8
Description:	This document describes the security layer that provides confidentiality and message integrity for the Z Wave Network.
Written By:	JFR;JRM;NTJ
Date:	2009-07-28
Reviewed By:	JRM;PSH;JFR;JSI
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2009-07-28	14:00:20	NTJ	Niels Thybo Johansen	

CONFIDENTIAL

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20070213	NTJ; JFR	ALL	First draft
2	20071112	JFR	Section 3 Section 4.1 Section 4.2.1 Section 7	Feature list added How to obtain a secure solution extended Security levels with respect to exchange of the network wide key added Interoperability description added.
2	20071115	JRM	Section 4.4	Security Command Class Identifier added to encapsulated package, table 1. Encrypted payload maximum set to 29 bytes.
3	20071130	JRM	ALL	Minor changes.
4	20080311	JRM	Section 7.2 Section 7.2 Section 3, 4.1.3 & 4.2.1	Reference to Security Level Configuration Command Class removed. Rules with respect to Basic command class listing in node info frame. Temporary key changed from octal to decimal
5	20080408	JFR	Section 4.2.1	Updated to only support security 0 using low power to improve usability, but security scheme is extendable to offer stronger key exchange at a later stage.
6	20080425	JFR	Section 4.2.1	Updated to only support security 0 using normal power to improve usability,
7	20080922	JFR	Section 4.1.3	Security 20 remark removed.
8	20090709	JRM	Section 6.2	Message Authentication Code, clarification on contents and order

Table of Contents

Table of Figures

Table of Tables

1 ABBREVIATIONS

Abbreviation	Explanation
AES	The Advanced Encryption Standard is a symmetric block cipher algorithm. The AES is a NIST-standard cryptographic cipher that uses a block length of 128 bits and key lengths of 128, 192 or 256 bits. Officially replacing the Triple DES method in 2001, AES uses the Rijndael algorithm developed by Joan Daemen and Vincent Rijmen of Belgium.
ENC	Encryption; here: AES in Output Feedback mode
IV	Initialization Vector
MAC	Message Authentication Code; here: AES in Cipher Block Chaining mode.
AT	Authentication Tag, identical to MAC.
Nonce	Number used once
OPP	Original Payload Package
PRNG	Pseudo-Random Number Generator
RI	Receiver's nonce Identifier
SPP	Security Payload Package

2 INTRODUCTION

This document describes the security solution that provides confidentiality and message integrity for the Z-Wave network.

2.1 Purpose

Describe the general security mechanism in Z-Wave device classes.

2.2 Audience and prerequisites

Zensys and Z-Wave alliance partners

3 Z-WAVE SECURITY FRAMEWORK

The Z-Wave security solution comprises of the following features:

- End-to-end security on application level (communication using command classes).
- In-band network key exchange at normal power using a 0 decimal temporary key.
- Single network wide key.
- AES symmetric block cipher algorithm using a 128 bit key length.
- Secure and non-secure nodes can co-exist in the same network.
- No security solution on MAC layer and routing layer.
- Non-secure nodes can act as repeaters for secure nodes.
- Only single cast is supported.
- Only 40+ kbps nodes can act as secure nodes.
- Modification attack prevention. The attacker modifies a message in transit, trying to make the recipient believe that the modified message was sent by a legitimate sender.
- Fabrication attack prevention. The attacker creates a completely new message, trying to make the recipient believe that it was sent by a legitimate sender.
- Replay attack prevention using a challenge/response mechanism. The attacker intercepts a message in transit and stores it. At some later point in time, he sends this message to the intended receiver, making him believe that the message was sent just recently.
- Preplay attack prevention. The attacker requests a response from a node before it is requested by a network node. Once the request from the legitimate node comes, he sends the old, recorded response.
- Reordering attack prevention. The attacker delays a message until another message that was sent later has arrived at the destination.
- Interception attack prevention. The attacker intercepts a message in transit and reads the contents.

4 THE SECURITY LAYER

4.1 Security goals

The security layer provides message integrity, confidentiality and data freshness:

- *Message integrity* means that the receiver can be sure that the message received was sent by a secure node in the network, that this message has not been altered. Messages sent by radio transmitters outside the network will be recognized as fakes.
- *Data Freshness* means that the message has been sent recently.
- *Confidentiality* means that only secure nodes in the network can read the actual contents of the message. For all other radio receivers, the messages sent appear as random white noise.

These goals are achieved by transforming outgoing messages using encryption and a message authentication code (MAC). The protocols used for sending and receiving secure messages are described in Section 6.

Note that there are a number of attacks that the security layer does not protect against, as e.g. denial of service attacks, hardware side-channel attacks, traffic analysis, protocol side-channel attacks, or attacks against application-layer security.

The overall Z-Wave security solution comprises therefore of three parts:

- Security elements implemented on protocol level
- Security elements implemented on application level provided by Zensys
- Security elements implemented on application level provided by OEM

Each part or combinations provides protection, detection or reaction against a number of security attacks. The complete security solution must encompass all three security components of prevention, detection, and reaction.

- Prevention - facilities and systems to prevent people getting in and taking information.
- Detection - to find out if anybody has gotten in, and compromised important information or processes.
- Reaction - to allow the "bad guys" to be identified and their activity stopped.

4.1.1 Interruption

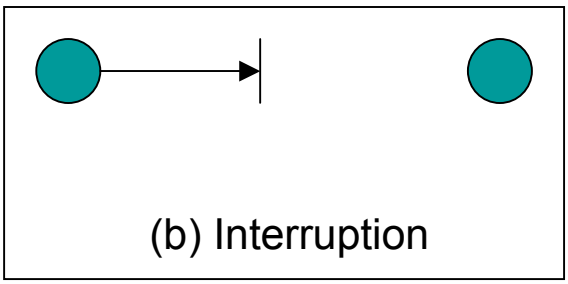

Figure 1, Interruption of information flow

It is impossible to protect 100% against all kind of interruption attacks. A very simple attack could be jamming the radio frequency used, tampering of the device in question or denial-of-service attack by saturating the target node with external communication requests.

Detection of interruption attacks is especially important in the destination nodes because they receive typically crucial alarm information. This can be achieved by sending keep-alive messages to the destination at a given rate. This enables detection of interruption of the keep-alive messages in the destination. This functionality is not provided by Zensys and must therefore be implemented by the OEM on application level.

4.1.2 Interception

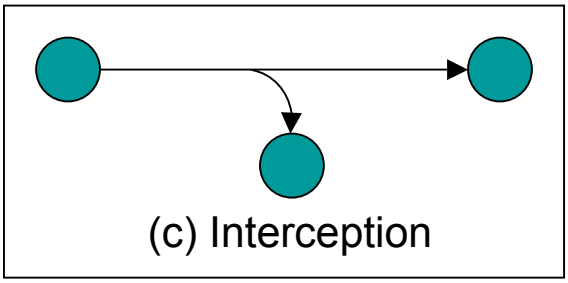

Figure 2, Interception of information flow

Zensys provides a solution which prevents the release of any user data transmitted (interception).

A passive attack is traffic analysis which may reveal important knowledge about the state of the system. Traffic analysis can be performed even when the messages are encrypted and cannot be decrypted. In order to make traffic analysis difficult the OEM's application should have the same traffic profile independent of the systems state (armed, unarmed etc.).

4.1.3 Modification

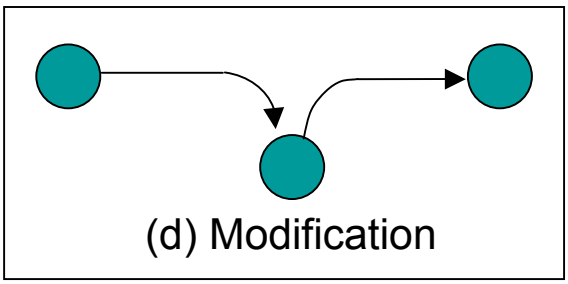

Figure 3, Modification of information flow

Zensys provides a solution, which makes it extremely difficult to change or corrupt the content in the messages as long as the intruder does not know the 128bit key used by the network.

The solution does not provide individual link keys due to the limited memory resources.

The only realistic way to eavesdrop the key is during inclusion of nodes, because the temporary initial encryption key is weaker than the network key. The temporary initial key used for exchanging the network key is currently only Security 0/N where null key is used. The primary/inclusion controller determined security level used.

To make it difficult eavesdrop the key during inclusion low output power can (typically range is below 1 meter) be applied instead of normal output power.

4.1.4 Fabrication

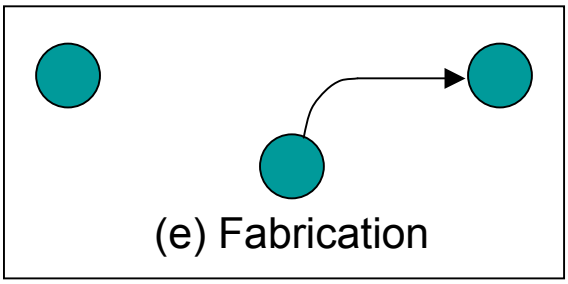

Figure 4, Fabrication of information

Zensys provides a solution which makes it difficult to fabricate messages as long as the intruder doesn't know the 128 bit key used by the network. The same apply as mentioned under the modification attacks. Another type of attack in this category is replay attacks in which a valid data transmission is maliciously or fraudulently repeated or delayed. The Zensys solution protect also against replay attacks using a challenge response mechanism.

4.1.5 Network Management

Network management functionalities such as inclusion, exclusion etc. can be on application level protected by a password to avoid unauthorized operation. The OEM must implement this kind of access control.

4.2 Key handling

A node that is part of the network is characterized by being in possession of the network key. This network key is 128 bit long and is shared by all nodes of the network.

The initial key exchange is performed in band at reduced output power. The key is generated by use of a PRNG.

The network key K_N is stored in NVRAM. It is never used directly for encryption and authentication purposes. Instead, authentications key K_A and an encryption key K_E are derived from it, as follows:

$$K_A = AES(K_N; V_1) \qquad K_E = AES(K_N; V_2)$$

This means that constants V_1 and V_2 are encrypted under the network key to obtain the authentication and encryption keys, using the core AES cipher as described in section 4.1. The constants V_1 and V_2 are the bytes 0x55 and 0xAA, respectively, repeated 16 times. The derived keys K_A and K_E are stored in SRAM.

4.2.1 Security Schemes

Distribution of network keys uses a temporary key to protect the key exchange. Exchange of network key happens immediately after successful inclusion of the node. It requires a secure primary/inclusion controller to include a secure node into the secure network as secure. Currently one Security 0/N scheme using normal power for key exchange exist which is extendable at a later stage.

For further details, refer to [6].

4.3 Position in the protocol stack

The protocol stack of the Z-Wave solution without security consists of the following layers (from bottom to top): *physical*, *media access control*, *network*, *routing*, and *application*. If the security layer is used, it is placed between the routing and the application layer.

Two types of payload are relevant in the context of this document:

- An ***original payload package (OPP)*** is generated by the application layer. It does not contain cryptographic protection and uses the application layer frame format described in [5].
- A ***security payload package (SPP)*** is generated by the security layer. Its contents are usually cryptographically protected, and it uses the frame format described in Section 4.4.

Figure 5 describes how these payload types are used by the layers of the protocol stack.

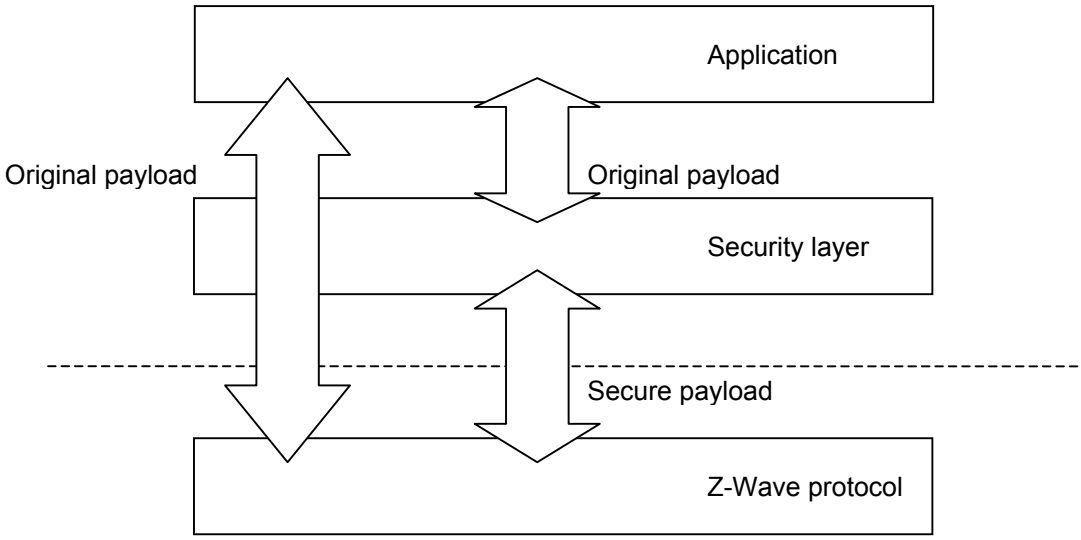

Figure 5, Application stack and message flow

When sending a payload package, the application layer calls either the "send" or the "send_secure" function. In the latter case, the OPP is transformed into an SPP by the security layer. Only after this transformation is it passed on to the lower layers of the protocol stack.

When receiving a payload package, the Z-Wave protocol checks whether the Command class is equal to the security command class [6]. If not equal, the package is considered as original payload and is passed directly on to the Z-Wave protocol. If equal, then the package is considered as security payload, and it is transformed back into an OPP by the security layer before being passed on to the Z-Wave protocol.

4.4 Format of security payload package

There exist different types of security payload packages, which have different formats. However, they all have a security header, which may or may not be followed by security data and the original (i.e., application) payload data in encrypted form.

- The security header specifies the type of security package and defines if the package contains security data and/or original payload data in encrypted form. It also contains flags that control the protocol flow between the sender's and the receiver's security layers.

- The security data may include a new nonce value, an identifier for an old nonce value, and/or an authentication tag.

- If the packet contains original payload data, it is in encrypted form.

If the OPP received from the application layer is large, adding the security header can increase the total size of the SPP beyond the acceptable limit. Thus, the security layer verifies whether the original payload is at most 29 bytes long. If this threshold is exceeded, processing must be aborted, and an error message is returned.

The general format of SPPs is described in Figure 6, and the fields used are listed in Table 1. Note that not all fields are used with all types of SPPs; the shortest possible packages consist only of a security header (see Table 2).

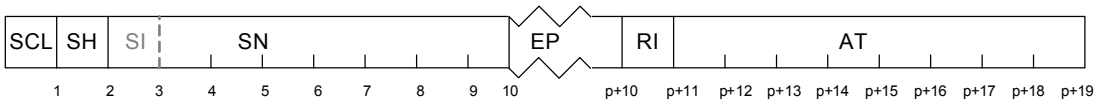

Figure 6, Structure of a security payload package

Table 1, Fields of a security payload package

Name	Size	Content
SCL	1 byte	Security Command Class Identifier
SH	1 byte	Security Header
SI	1 byte	Sender's nonce Identifier (part of SN)
SN	8 bytes	Sender's Nonce
EP	0-29 bytes	Encrypted Payload
RI	1 byte	Receiver's nonce Identifier
AT	8 bytes	Authentication Tag

The structure of a security header is described in Figure 7. The following fields are present:

- SN (Sender's Nonce embedded)
- NR (Nonce Request; a packet containing only a sender's nonce is returned immediately)
- C (Content)
  - o E (0000; Empty)
  - o P (0001; Payload)

Note that bits 3 through 7 are set to zero in the current version. They are reserved for use in later versions of the protocol. Table 2 describes the types of SPPs (header flag combinations and fields contained) currently supported.

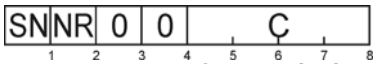

Figure 7, Structure of a security header

Table 2. Possible types of SPPs

Type	SH	Flags	Has SN	Has EP	Has RI+AT
Nonce request	0100-0000	NR, C = E	-	-	-
Nonce	1000-0000	SN, C = E	√	-	-
Message	1000-0001	C = P	√	√	√
Message w. Nonce Req.	1100-0001	NR, C = P	√	√	√

5 ELEMENTARY BUILDING BLOCKS

5.1 Core AES (AES)

The security layer uses an AES-128 software implementation. This algorithm encrypts a single 128-bit block of data using the Advanced Encryption Standard, AES. It receives a 128-bit key and a 128-bit data block as input and produces a 128-bit output block. Note that this method can not be used for decryption. The algorithm is defined in [2].

5.2 Message Encryption (ENC)

The security layer offers an invertible method of encryption, which is primarily used for payload data. This algorithm encrypts and decrypts messages of arbitrary length, using AES in Output Feedback Mode. It receives a 128-bit key, a v-bit initialization vector (IV), and an n-bit data block as input and produces an n-bit output block. The mode of operation is defined in [4].

5.3 Message Authentication Code (MAC)

The security layer offers a method of message authentication, using AES in cipher block chaining mode. The input consists of a 128-bit key, a v-bit initialization vector (IV), and an n-bit data block[1]. An 8-byte authentication tag is produced as output. Again, the mode of operation is defined in [4].

5.4 Pseudo-random Number Generator (PRNG)

The Z-Wave hardware can produce random numbers. Since, however, a node running this hardware generator can not participate in normal communication and since the generator is slow, using the hardware generator is often not practical. Instead, a pseudo-random number generator (PRNG) is used. This generator has a 128-bit inner state (stored in SRAM) and three functions: state generation, state update, and output generation.

The solution is based on the Barak-Halevi PRNG proposal [1] and the Davies-Meyer hash function [3], both of which are provably secure. The three functions are:

5.4.1 State initialization

The inner state is initialized upon including the node into the network and after SRAM loss (e.g., due to a power-down):

1. Set the inner state to zero.
2. Run the state update function.

5.4.2 State update

The inner state of the generator is sometimes updated with new random bits from the extractor. This is done as follows:

1. Get 128 bits of fresh input H_2 as follows:
    a. Collect 256 bit (32 byte) of data from the hardware generator.
    b. Split these bits into two 128-bit keys denoted K_1 and K_2, and set H_0 to be the value 0xA5 (repeated 16 times).
    c. Compute $H_1 = AES(K_1; H_0) \oplus H_0$.
    d. Compute $H_2 = AES(K_2; H_1) \oplus H_1$.
2. Compute S = Inner State $\oplus$ H_2.
3. Use S as AES key and encrypt the value 0x36 (repeated 16 times).
4. Store the result as the new inner state in SRAM, and make sure to delete the old inner state and all intermediate values.

In literature, it is often recommended to call the PRNG update function on a regular basis, in order to keep the entropy in the system sufficiently high. However, in the current version of the system, the update function is called only as a part of the initialization process. This does not endanger the security of the system, since the attacker can neither learn about the PRNG inner state without breaking the underlying AES, nor read the PRNG from memory without getting access to the keys (thus breaking the system completely).

5.4.3 Output generation

Every time that k bits ($k \leq 128$) of output are requested from the PRNG, the following steps are followed:

1. Use the current inner state as AES key.
2. Encrypt the value 0x5C (repeated 16 times) and use the least significant k bits of the result as PRNG output.
3. Encrypt the value 0x36 (repeated 16 times) and store the result as the new inner state in SRAM. Make sure to delete the old inner state and all intermediate values.

If more than 128 bits are required, the output generation function has to be called several times.

6 COMMUNICATION PROTOCOL

6.1 The role of nonces

A nonce (i.e., a ***n***umber used ***once***) is a publicly known value. In the security layer, nonces have two properties:

- They are fresh, i.e., the attacker can not predict them before they were generated, and they remain valid only a short time. Thus, they can be used to check whether a message associated with the nonce is also fresh.
- They are used only once, i.e., the attacker does not gain anything from storing a tuple (message, nonce) in the hope that the same nonce is used again.

In practice, nonce's are used as part of the initialization vector (IV) that is used for authentication and encryption. Both the sender and the receiver of the SPP contribute an 8-byte nonce value to this IV, which is simply constructed by concatenation:

IV = (sender's nonce || receiver's nonce)

Obviously, the sender has to be in possession of the receiver's nonce before he can construct the IV and send the SPP. Thus, we need a special communication protocol for nonce exchange, which will be described in the following sections.

6.2 Sending individual packages

The communication protocol is best illustrated using the simplest possible scenario: The security layer of node **A** has received one single OPP from the application layer, which is to be sent in a secure way to node **B**'s application layer. This is done by going through the following steps:

1. **A** sends a "nonce request" SPP to **B**.
2. **B** uses its PRNG to generate a random 8-byte nonce and sends it in a "nonce" SPP to **A**. He also stores the nonce in his internal nonce table (see Section 6.3).
3. **A** considers this value as *receiver's nonce* for node **B**. He then transforms the OPP into an SPP, as follows:
    a. **A** uses its PRNG to generate an 8-byte *sender's nonce*. It is concatenated with the receiver's nonce to form the 16-byte IV, i.e.: *IV* = (*sender's nonce* || *receiver's nonce*).
    b. **A** encrypts the original payload under the encryption key K_E and the IV.
    c. **A** computes the authentication tag under the authentication key K_A and the IV. The tag is based on the following data, in order:
        - Full Initialisation Vector (IV)
        - Security Header
        - Sender Node ID.

- Receiver Node ID.
- Payload length
- The encrypted payload data (and thus implicitly command class, command, and parameters[2]).

d. **A** appends the security header, the sender's nonce, the first byte of the receiver's nonce (as Receiver's nonce Identifier (RI)) and the authentication tag to the encrypted payload (for details, see Figure 6).

After that, the resulting SPP is passed on to the routing layer for sending.

4. Upon receiving the SPP, **B** uses the Receiver's nonce Identifier (RI) in the package to identify the correct receiver's nonce in his "internal nonce" table. He reconstructs the IV and then verifies the correctness of the authentication tag. If it is incorrect, the SPP is discarded. Otherwise, the original payload is decrypted and passed on to the application layer.

A graphical illustration of the protocol run is given in Figure 8.

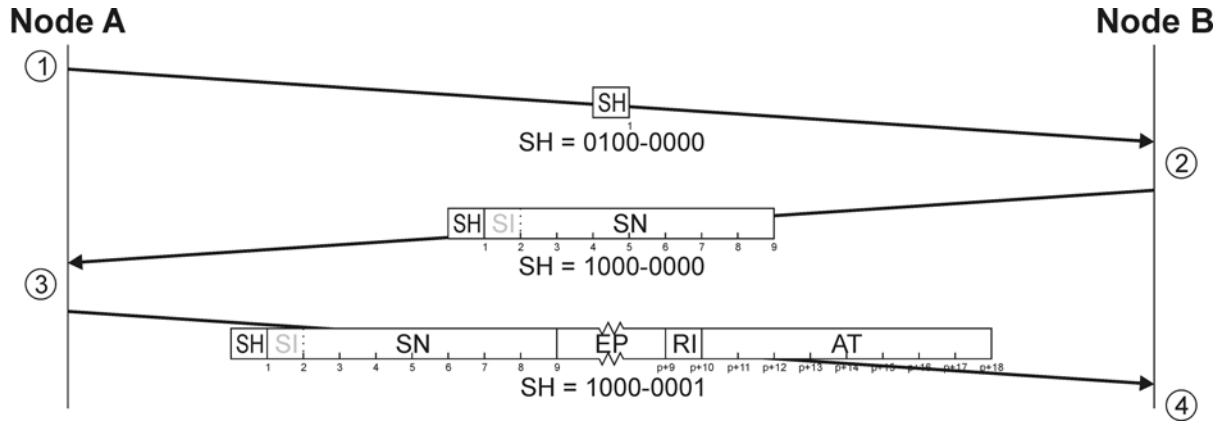

Figure 8, Example for a single-message scenario

[2] This is important, since the command class is used by the application to interpret the payload data. If the attacker would be able to modify the command class without being noticed, the results at application level could be unpredictable.

6.3 Handling internal nonces

A node has to keep a copy of all the nonces that it has send out and that are currently active. This is handled by the table of internal nonces, containing the following data:

- Nonce value (8 bytes)
- Nonce receiver ID (1 byte)
- Validity (1 byte)

Nonces are identified in the table by their first byte. When generating a new nonce, it is ensured that the new nonce does not have a first byte value that is already contained in one of the table entries[3]. Since at each point in time, this first byte is unique in the table, it can be used to identify nonces. Thus, when receiving an SPP containing an authentication tag and Receiver's nonce Identifier (RI), the receiver can use this value to identify the nonce corresponding to this tag.

The handling of such nonces is described in Figure 9. The process is as follows:

1. A nonce is generated by node **B**. This can be because **A** has requested a nonce or because **B** wants to send an SPP on behalf of his own application layer. In both cases, the nonce is stored in the internal nonce table.

2. The nonce is sent as part of an SPP to **A**.

3. If **A** wants to reply to **B**'s package, he uses the nonce as receiver's nonce to compute the IV (and thus to encrypt and authenticate the OPP). He also includes its Receiver's nonce Identifier (RI) in the package.

4. Node **B** uses the Receiver's nonce Identifier (RI) to find the correct nonce in his internal nonce table and uses it to process **A**'s package.

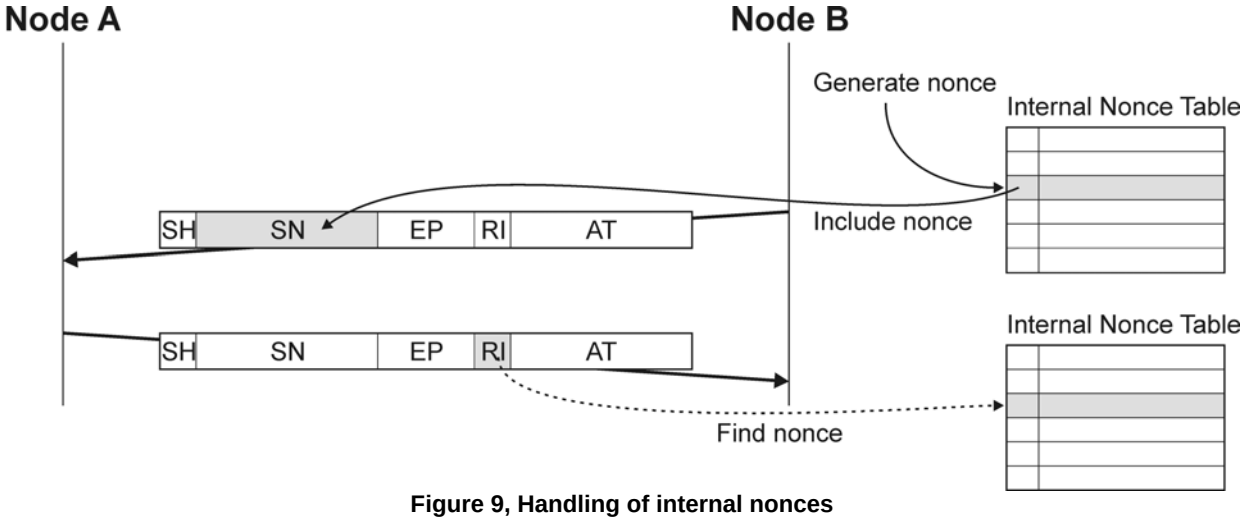

Figure 9, Handling of internal nonces

[3] If the PRNG returns a nonce whose first byte is already contained in the table, then the process is repeated.

The table is not sorted. It has a fixed size that is chosen at compilation time, but is limited to at most 128 entries[4]. The larger the table gets, the more connections can be serviced. At the same time, the memory consumption (and to some extend, also the running time) increases with the table size.

When a new nonce has to be stored in the table, the first free space is used. If no free space is available, the new nonce must be thrown away allowing for active communication to finish first. This is safe to do, due to the second rule below. Old entries in the table must be deleted, if one of the following is true:

1. A reply from the nonce receiver was obtained, regardless of whether this reply had a valid authentication tag or not. Note that this means that upon receiving one reply from a certain node, **all** nonces associated with this node are deleted (not just the one contained in the reply or nonces that are older than the one contained in the reply). This serves to prevent re-ordering attacks.
2. The validity of the nonce has expired. A nonce is valid up to 3 seconds after its generation.

6.4 Handling external nonces

The sending node can send only one SPP at any one time. When an SPP shall be send, there are two possible scenarios:

1. An external nonce (to be used as receiver's nonce now) is already present, since it was contained as sender's nonce in an SPP received earlier. In this case, the SPP can be send right away.

2. No external nonce is present. In this case, a nonce request has to be sent and an answer has to be received before the OPP can be transformed to an SPP and sent.

In both cases, a buffer is required.

In **scenario 1**, the external nonce has to be stored until it is used. Figure 10 describes how such an external node is processed:

1. **A** receives an SPP from **B** which contains a nonce. Since **A** does not have any immediate use for this nonce, he stores it in his external nonce buffer.

2. Later (but before the external nonce expires), **A** wants to send a full-size SPP to **B**. He fetches the nonce from the external nonce buffer. He then uses it as receiver's nonce to compute the IV (and thus to encrypt and authenticate the OPP). He also includes its Receiver's nonce Identifier (RI) in the package.

[4] Since a nonce is valid only 3 seconds, since the network can only handle 25 packets per second, and since only half of these packets can be used to send nonces, the maximum table size should be 38. If this value is chosen, no nonces that did not expire yet have to be pushed out of the table.

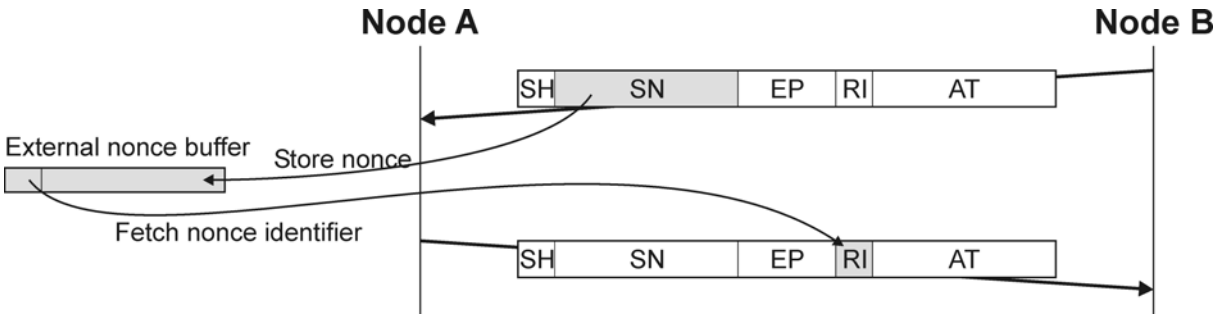

Figure 10, Handling of external nonces

A node is only able to store one such nonce at any one time, and it is stored as a triple *(node ID, nonce, validity)*. The validity of an external nonce is 1 second after it was received. If no SPP needs to be sent to the nonce's originator before it expires, then the nonce is discarded. The same happens if an external nonce from another node was received before the old nonce could be used - in this case, the older nonce is discarded.

In **scenario 2**, the OPP itself has to be stored until an external nonce is received. Again, only one such OPP can be stored at any one time, and it is stored as a triple *(node ID, OPP, validity)*. If no nonce is received before the validity expires, the OPP is discarded. No new OPPs are accepted for processing while a valid OPP is still stored in the buffer.

The validity of an OPP should be as short as possible. Since it depends on the application scenario, it is set at compile time. Note that increasing the validity of the OPP also does increase the risk of attacks where the attacker delays the sending of the SPP in order to send it at a later time than intended by the user. Thus, a default validity of 3 seconds is recommended. However, for applications where delaying of messages does not pose a threat, this default validity may be increased.

7 INTEROPERABILITY

This chapter describes various aspects of interoperability.

7.1 Application versus Stack implementation Interoperability

Z-Wave provides secure application communication between nodes through a flexible security layer implementation:

- Option 1: Z-Wave Application security layer implementation. Security layer is implemented on application level (including AES-128) on the ZW0201/ZW0301 Z-Wave single chip platforms.
- Option 2: Z-Wave Protocol security layer implementation. Security layer is implemented in the Z-Wave protocol and with AES.128 implemented in hardware on the ZW0401 Z-Wave single chip platform.

  Option 1 and option 2 is interoperable.

7.2 Secure and non-secure Z-Wave nodes Interoperability

- Security enabled Z-Wave nodes can be included by a non-secure inclusion controller as non-secure nodes ensuring backwards compatibility with existing applications.
- Non-secure Z-Wave nodes can be included by a secure inclusion controller as non-secure nodes.
- Security enabled nodes configured into flex security mode can communicate secure to some nodes and non-secure to others
- Both secure and non-secure nodes participate in the Z-Wave routing protocol and Network management functionalities
- Security enabled Z-Wave nodes can NOT be included by a non-secure inclusion controller as secure nodes.

The Z-Wave security solution supports networks with mixed secure and non-secure communication in order to leverage on the existing non-secure products. Both secure and non-secure nodes can participate in the routing algorithm.

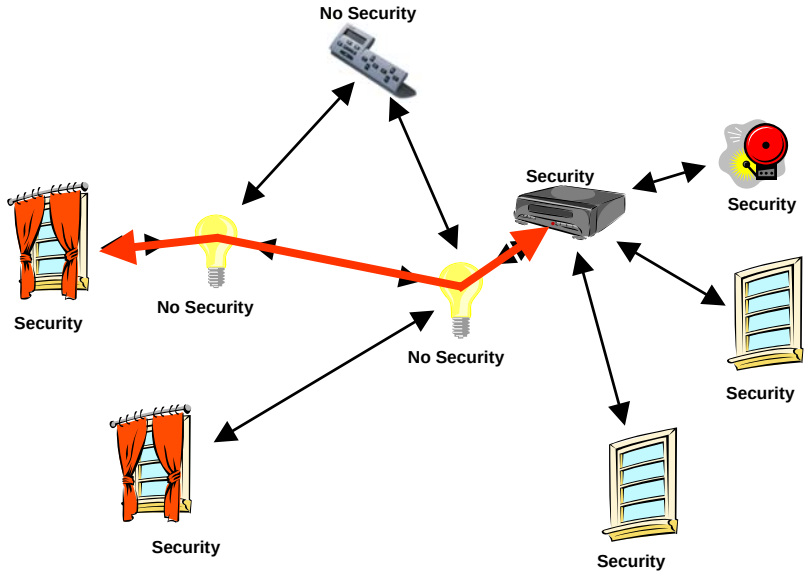

Figure 11, Mixed secure and non-secure network

It is up to the implementation of each application to decide which commands should be supported using security encapsulation. For example, a device may choose to support all its command classes non-secure if it is being included to a non-secure network, but no command classes non-secure if it is included secure.

If a command class is only supported using security encapsulation it must not be listed in the node info frame, but must instead be listed in the security commands supported report frame. Additionally, the node information frame must contain the security command class.

Basic command class must be listed in the Node Info Frame if it is supported non-secure. If not listed in the Node Info Frame it is implicitly supported Secure.

8 ERROR HANDLING

Some potential error situations have already been discussed in the relevant sections of this document. In addition, a number of additional error situations can occur:

8.1 Problems with incoming OPP

- If an OPP with length greater then 29 bytes is attempted sent using the security layer, an error should be returned.

8.2 Problems with incoming SPP

- If the Security Header (SH) contains an unknown bit pattern, then the SPP is discarded. This does not only protect against misinterpretation of SPP that have been modified (accidentally or on purpose), but also against misinterpretation of SPP that are formatted according to a future version of the security protocol.
- If the Receiver's nonce Identifier (RI) in an SPP does not correspond to one of the nonces in the receiver's internal nonce table, the SPP is discarded. The sender's nonce (SI) contained in the SPP, however, is stored in the buffer for future use.
- If the Authentication Tag (AT) in an SPP is not correct, the SPP is discarded (as described in Section 6.1). The Sender's nonce Identifier (SI) contained in the SPP, however, is stored in the buffer for future use.

8.3 Problems with outgoing SPP

- If an error occurs at network level when attempting to send a secure packet with payload, this error should be forwarded to the application.
- If an error occurs at network level when attempting to send a secure packet without payload, the following steps are taken:
    - If the package was a nonce request, the OPP is discarded and the application is notified.
    - If the package was a nonce, then the nonce is deleted from the internal nonce table and no further action is taken.
- If an outgoing OPP waits for a nonce and times out, the OPP is discarded, as discussed in Section 6.4. In addition, the application is notified.
- If too many internal nonces are generated in a short spell of time (i.e., before they expire or get used by the communication partner), the table of challenges will overflow. New nonces will then push the oldest ones out. In an extreme situation (e.g., a massive alarm by all sensors in a sensor network), this can lead to a deadlock situation where due to a constant demand for new nonces, internal nonces get overwritten by new ones before the responses corresponding to those nonces are received. The table should thus be dimensioned in such a way that it can handle the worst-case communication situation for the application at hand.

8.4 Network problems

- The application layer has to be aware that SPPs can get lost in transmission just as OPPs can.
- The application layer has to be aware of that an Ack or a Routed Ack for an SPP does not mean that the package reached the receiver's application layer. It could have been accepted by the network or routing layer (which send the Ack or Routed Ack, respectively) and then have been discarded by the security layer (e.g. due to wrong Receiver's nonce Identifier (RI) or wrong Authentication Tag (AT)).

8.5 Failure recovery

- If the battery is depleted or an A/C powered device is disconnected, the SRAM will be lost. This affects a number of security-relevant data which have to be reset properly:
    - The PRNG has to be re-initialized.
    - The encryption and authentication keys K_E and K_A have to be re-computed.
    - The internal nonce table has to be reset.
    - The buffers for external nonce's and waiting OPPs have to be reset.
- The same steps have to be taken if the application crashes and has to be restarted.

REFERENCES

[1] B. Barak, S. Halevi: A model and architecture for pseudo-random generation with applications to /dev/random. IACR eprint 2005/029.
[2] Federal Information Processing Standards Publication 197, November 26, 2001: Advanced Encryption Standard (AES).
[3] S. Matyas, C. Meyer, J. Oseas: Generating strong one-way functions with cryptographic algorithm. IBM Technical Disclosure Bulletin, 27(1985), pp. 5658-5659.
[4] NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation - Methods and Techniques, 2001.
[5] Zensys, SDS10243, System Design Specification: Z-Wave Protocol Overview.
[6] Zensys, SDS10868, System Design Specification: Security Command Class.