Software Design Specification

Z-Wave Application Security Layer (S0)

Document No.:	SDS10865
Version:	11
Description:	This document describes the S0 security layer for Z-Wave applications. Note that S0 has been superseded by S2.
Written By:	JRM;JFR;ABR;BBR
Date:	2018-03-02
Reviewed By:	JRM;CST;HKRONER;AES;JBU;ABR;JFR
Restrictions:	Public

Approved by:

Date	CET	Initials	Name	Justification
2018-03-02	15:48:38	JFR	Jørgen Franck	on behalf of NTJ

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 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 7.1 Section 7.1 Section 3, 6, 8	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	ALL	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	ALL	Updated to only support security 0 using normal power to improve usability,
7	20080922	JFR	-	Security 20 remark removed.
8	20090709	JRM	Section 6.2	Message Authentication Code, clarification on contents and order
9	20130618	JRM	ALL	Major rework, to reflect recent specification. The document is now supplementary to the command class, instead of duplicate.
9	20131001	JRM	ALL	Fixed typos.
10	20160301	JFR	Section 2	Introduced both Security S0 and S2
11	20180302	BBR	All	Added Silicon Labs template

Table of Contents

Table of Figures

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.
AT	Authentication Tag, identical to MAC.
ENC	Encryption. Represents AES in Output Feedback mode in this document
IV	Initialization Vector
MAC	Message Authentication Code. Represents AES Cipher Block Chaining mode in this document
Nonce	Number used once
PRNG	Pseudo-Random Number Generator
RI	Receiver’s nonce Identifier
S0	First-generation security based on the Security Command Class
S2	Second-generation security based on the Security 2 Command Class

2 INTRODUCTION

This document describes the first generation security solution based on Security Command Class (S0) that provides confidentiality and message integrity for the Z-Wave network. The term S0 is used to distinguish the Security Command Class from the Security 2 Command Class (S2). S2 supersedes S0. Compared to S0, S2 features stronger key exchange, stronger authentication, lower communication overhead and multicast support.

This document presents S0. For further details regarding S2, refer to [8], [9], [10], [11] and [12].

2.1 Purpose

Describe the general security mechanism in Z-Wave Device Classes [5].

2.2 Audience and prerequisites

The intended audience is Silicon Labs and Z-Wave alliance partners.

Familiarity with the Security (S0) Command Class described in [6] and [7] is a prerequisite.

3 Z-WAVE S0 SECURITY FRAMEWORK

The Z-Wave security solution S0 comprises of the following features:

- End-to-end security on application level (communication using command classes).
- In-band network key exchange well-known 128 bit zero 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.

4 S0 SECURITY LAYER

4.1 Security goals

The S0 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 and 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 payload of encrypted messages sent appear as random data.

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 more details in the Security Command Class in [7].

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 two parts:

- Security elements implemented on application level provided by Silicon Labs
- Security elements implemented on application level provided by OEM

Each part or combinations provide 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 obtaining information.
- Detection - to find out if anybody has gained access, and compromised important information or processes.
- Reaction - to allow the "bad guys" to be identified and their activity stopped.

5 S0 BUILDING BLOCKS

While the frame flow and frame format of the Security Command Class is described in [7], it requires a number of building blocks to facilitate the different encryption modes and other operations necessary.

For encapsulating messages, Z-Wave requires four commands. Before sending an encrypted frame, the sender MUST acquire a nonce (number used once) from the recipient. The sender then uses this number along with a locally generated nonce to encrypt the message using an encryption key and finally signs with an authentication key to generate the Security Message Encapsulation Command as illustrated below.

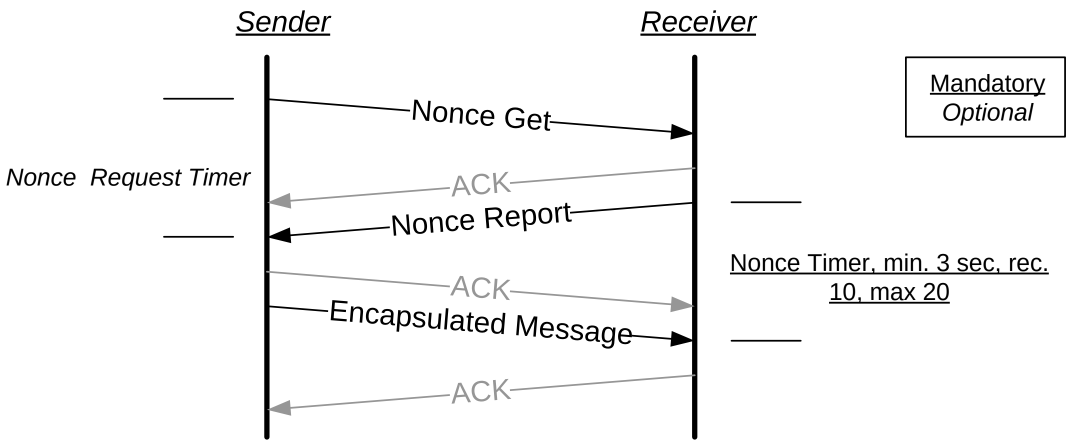


Figure 1, Sending secure messages

To perform these operations the Security Command Class relies on the AES engine and the accompanying modes of operation:

- AES-128 "RAW". This is also known as AES-Electronic Code Book, or ECB, and defines the basic operation of encrypting a 16 Byte Plaintext into a 16 Byte Ciphertext using an encryption key. This is used as the basis of the following modes.
- AES-128 OFB. Output Feedback, or OFB, is the mode of operation used to encrypt and decrypt the payload.
- AES-128 CBC-MAC. Cipher Block Chaining – Message Authentication Code, or CBC-MAC, is the mode of operation used to create the Message Authentication Code that allows the authenticity of the frame to be verified.

In addition to the AES modes, the following building blocks or terms are also used:

- Pseudo Random Number Generator, or PRNG. The PRNG is hardware seeded by the Z-Wave radio and is utilized to create Initialization Vectors and Network Keys.
- Initialization Vectors, or IV's. Also called Number Used Once, or Nonce. Are used ensure freshness of a Secure frame to prevent replay attacks.

The communication protocol can be easily summed up, **A** wishes to send a secure message to **B**:

1. **A** sends a Nonce Get to **B**.

2. **B** uses its PRNG to generate a random 8-byte nonce and sends it in a Nonce Report to **A**. He also stores the nonce in his internal nonce table.

3. **A** considers this value as *receiver's nonce* for node **B**.

    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 MAC under the authentication key K_A and the IV.

    d. **A** composes a Security Message Encapsulation frame and sends it to B

4. Upon receiving the Security Message Encapsulation frame, **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 MAC. If it is incorrect, the frame MUST be discarded. Otherwise, the original payload is decrypted and passed on to the application layer.

The following sections will describe the above building blocks in more detail.

5.1 Core AES (AES)

The security layer uses AES-128, implemented in either hardware or software depending on setup. This algorithm encrypts a single 128-bit block of plaintext using the Advanced Encryption Standard, AES. It receives a 128-bit key (in the following referred to as Network Key, Encryption Key or Authentication Key) and a 128-bit plaintext block as input and produces a 128-bit ciphertext block. Note that this method cannot be used for decryption.

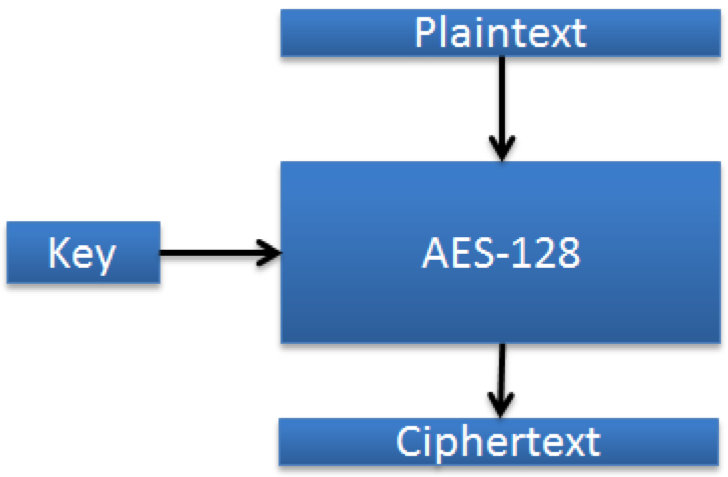


Figure 2, AES-ECB (Electronic Code Book)

5.1.1 Key Handling

A node that has been securely included into the network is characterized by being in possession of the AES Shared Network Key. This Network Key is 128 bit long and is shared by all secure nodes in the network.

The Network Key is generated by the initial including controller by taking 16 random bytes from the PRNG which MUST be hardware seeded as described in the PRNG section.

The Network Key K_N is stored in NVRAM. It is never used directly for encryption or authentication purposes. Instead Authentication Key K_A and Encryption Key K_E are derived from it, as follows:

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

Where V_1 and V_2 are defined as 0x55 and 0xAA, respectively, repeated 16 times. This means to create K_A , V_1 is encrypted using the Network Key K_N. And similarly using V_2 to create K_E. The derived keys K_A , and K_N are stored in SRAM.

5.2 Payload Encryption – AES OFB

The command class payload in the Security Message Encapsulation frame, is encrypted using AES OFB. An initialization vector is exchanged between the two parties as per the Security Command Class specification. The IV is then used as input for the AES-128 function using the Encryption Key K_E . The resulting encrypted IV is XOR'ed with the first 16 Bytes of the payload plaintext. If the plaintext is less that 16 bytes it is padded with 0x00 to the LSB). The resulting XOR'ed value is the first ciphertext block. For each 16 bytes of payload plaintext this process is continued, with the exception that the encrypted IV from the previous operation is used as IV in the following operation.

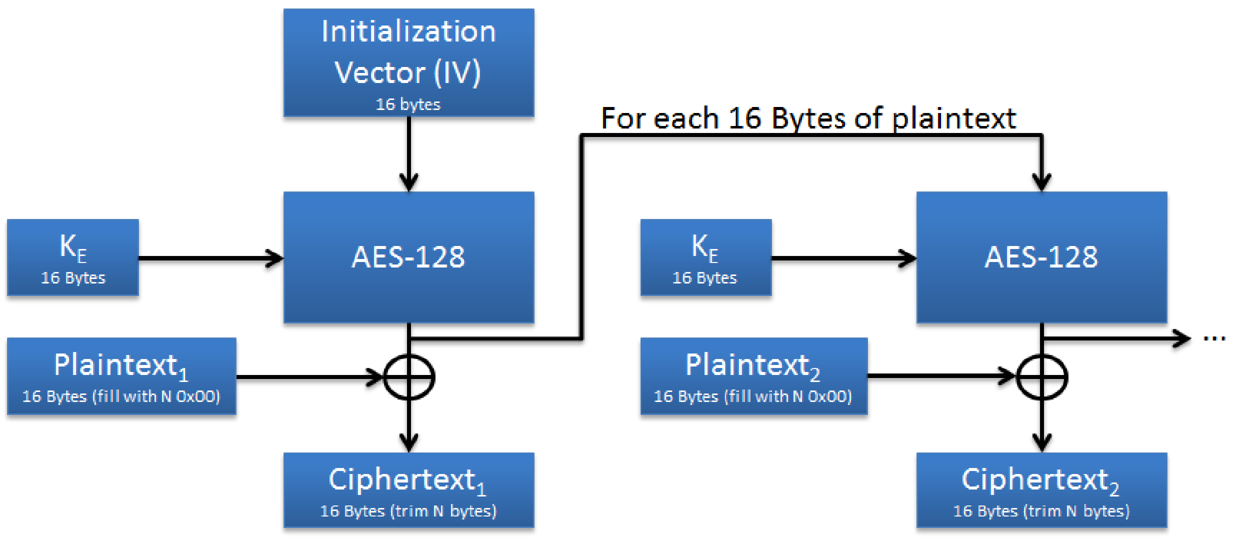


Figure 3, AES OFB

5.2.1 Initialization Vector - Nonce

The Initialization Vector, or 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.

The nonce MUST be generated by the PRNG.

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 Security Message Encapsulation frame contribute an 8-byte nonce value to this IV, which is simply constructed by concatenation:

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

5.2.1.1 Handling internal nonces

A node has to keep a copy of all the nonces that it has sent 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[1]. 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 Security Message Encapsulation frame containing a MAC and Receiver's nonce Identifier (RI), the receiver can use this value to identify the nonce corresponding to this tag.

The process is as follows:

1. A nonce is generated by node **B**. In response to a request by **A.**
2. The nonce is sent as part of an Nonce Report to **A**.
3. If **A** wants to reply to **B**'s package, he uses the nonce as receiver's nonce to compute the IV. 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.

The table is not sorted. It has a fixed size that is chosen at compilation time, but is limited to at most 128 entries. 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] If the PRNG returns a nonce whose first byte is already contained in the table, then the process is repeated.

1. A reply from the nonce receiver was obtained, regardless of whether this reply had a valid MAC 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 as per the timer definitions in the Security Command Class specification [7].

5.3 Handling external nonces

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

1. An external nonce (to be used as receiver's nonce now) is already present. In this case, the Security Message Encapsulation frame can be send right away.

2. No external nonce is present. In this case, a Nonce Get has to be sent and an answer has to be received before the Security Message Encapsulation frame can be constructed and sent

5.4 Message Authentication Code – AES CBC-MAC

Having encrypted the payload a Message Authentication Code, MAC, must be created, allowing the receiving node to verify the authenticity of the frame. The MAC is created by encrypting the first 16 bytes of authentication data, as specified in the next section, by using AES with Authentication Key, K_A. The output of this operation is XOR'ed with the next 16 bytes of authentication data and passing this through the AES function with K_A . This scheme is continued until all authentication data has been passed through. If the last block is less than 16 bytes it MUST be padded with 0x00's on LSB.

The resulting output of the last iteration is trimmed down to contain the first 8 bytes. This 8 byte value is now considered as the MAC.

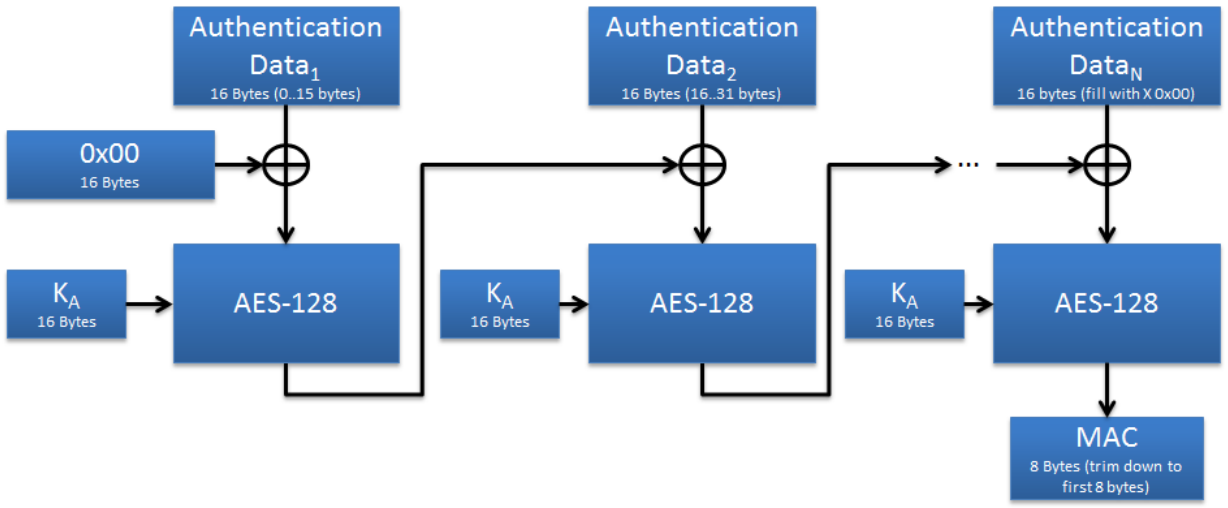


Figure 4, AES CBC-MAC

5.4.1 Authentication Data

The Authentication Data structure consist of the following, in order:

- 16 Bytes IV (senders nonce || receivers nonce)
- 1 Byte Security Header (command identifier)
- 1 Byte Sender Node ID
- 1 Byte Receiver Node ID
- 1 Byte Payload Length
- N Bytes Encrypted payload data (all inclusive, also sequencing)

5.5 Pseudo Random Number Generator – PRNG

The Z-Wave hardware can produce random numbers. Since, however, a node running this hardware generator cannot 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:

5.5.1 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.

5.5.2 State update

The inner state of the generator is sometimes updated with new random bits from the hardware random number generator. 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 $S = \text{Inner State} \oplus H_2$.
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).

5.5.3 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.

5.6 Timers

The Security Command Class is governed by a number of timers that MUST adhered to. Mainly they can be categorised into two groups:

- *Message Timers*: When sending and receiving secure messages there are two timers in effect, if any of the two timers are exceeded the message MUST be discarded.
    - *Nonce Request Timer* that is optional for the application to implement. This timer governs the duration of time permitted from sending a nonce get until receiving a nonce report and in effect how long it may take from an action on the application is resulted in the encrypted Security Message Encapsulation frame to be sent.
    - *Nonce Timer* that is mandatory and MUST be implemented by the application. This timer state the duration of timer permitted for the receiving node to wait for the Security Message Encapsulation frame to be received after it has sent the nonce report.
- *Inclusion Timers:* During inclusion each individual step of the inclusion process has a specific timer that must not be exceeded. If the timer is exceeded that secure inclusion MUST be aborted and the newly included node MUST become node secure.

6 ATTACKS AND ERROR HANDLING

When working with security extra caution must be taken to ensure the operation cannot be compromised.

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:

6.1 Problems with incoming Security Message Encapsulation frame

- If the Receiver's nonce Identifier (RI) in a Security Message Encapsulation frame does not correspond to one of the nonces in the receiver's internal nonce table, the an Security Message Encapsulation frame is discarded
- If the MAC in a Security Message Encapsulation frame is not correct, the Security Message Encapsulation frame MUST be discarded

6.2 Problems with outgoing Security Message Encapsulation frame

- 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 plaintext that was attempted transmitted 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 frame waits for a nonce and times out, it MUST be discarded. 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.

6.3 Network problems

- The application layer has to be aware that a Security Message Encapsulation frame can get lost in transmission just as non-secure frames can.
- The application layer has to be aware of that an Ack or a Routed Ack for an Security Message Encapsulation frame 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 MAC.

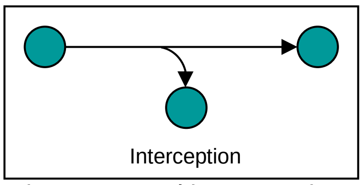
6.4 Failure recovery

- 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 payloads have to be reset.

The same steps have to be taken if the application crashes and has to be restarted

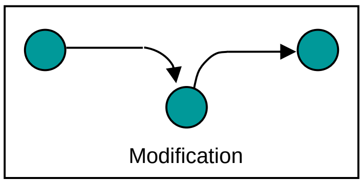
Besides the previously mentioned error conditions, which may or may not occur as a result of malicious intentions by an attacker, there are other concerns that must be handled.

- *Interception attack prevention*.



A passive attack using traffic analysis 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.).

- *Modification attack prevention*. It is extremely difficult to change or corrupt the content in the messages if the intruder does not know the 128bit shared key used by the network.

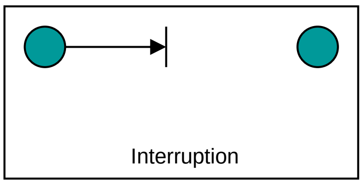


The solution does not provide individual link keys due to the limited memory resources.

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 temporary key is used. The primary/inclusion controller determined security level used.

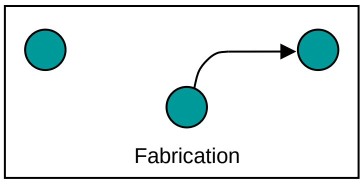
To make it difficult to eavesdrop the key during inclusion, low output power (typically range is below 1 meter) can be applied.

- *Interruption attack prevention*. 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 in case they are designed to receive 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 as part of the protocol library and must therefore be implemented by the OEM on application level.

- *Fabrication attack prevention*. It is extremely difficult to fabricate messages if the intruder does not know the 128 bit key used by the network. Another type of attack in this category is replay attacks in which a valid data transmission is maliciously or fraudulently repeated or delayed. A challenge response mechanism is used to protect against replay attacks.



  1. 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.

  2. 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. The Silicon Labs solution limits the interval in which reordering can occur by imposing a number of timers that dictate how long a message is valid.

- 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.

7 INTEROPERABILITY

This chapter describes various aspects of interoperability.

7.1 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.

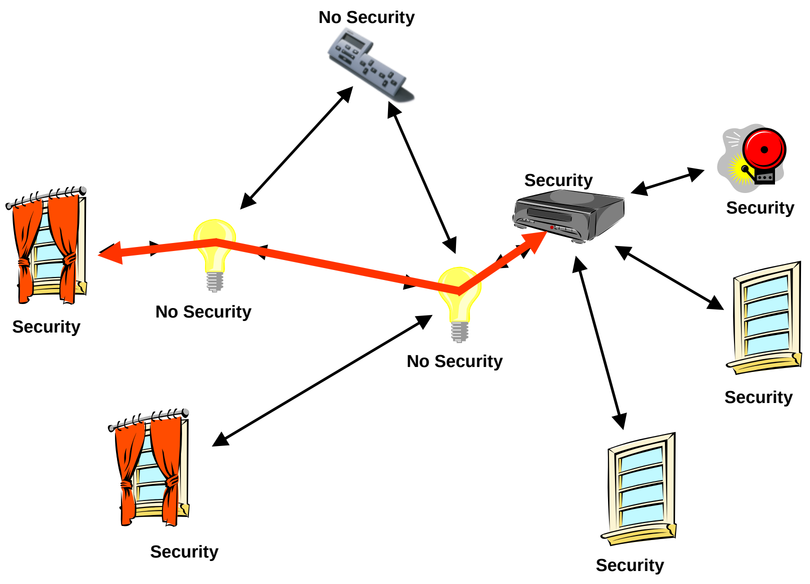


Figure 5, Mixed secure and non-secure network

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 securely.

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.

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] SD, SDS10242, Software Design Specification, Z-Wave Device Class Specification

[6] SD, SDS12657, Software Design Specification, Z-Wave Command Class Specification A-M

[7] SD, SDS12652, Software Design Specification, Z-Wave Command Class Specification N-Z

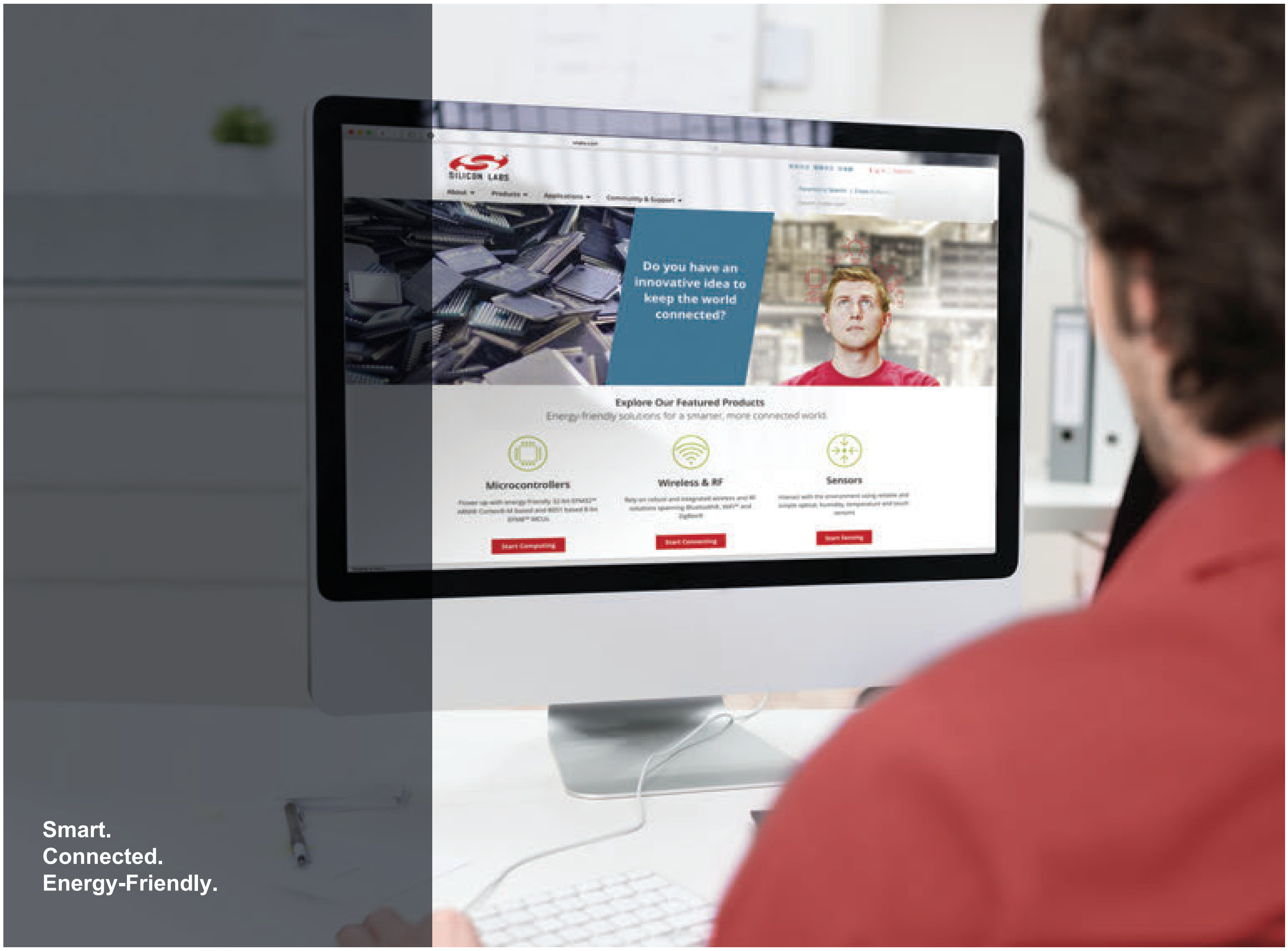
[8] SD, INS13474, Instruction, Z-Wave Security Whitepaper

[9] SD, SDS11274, Software Design Specification, Security 2 Command Class, version 1 (S0, S2, Security Command Class)

[10] SD, SDS13321, Software Design Specification, Supervision Command Class

[11] SD, SDS13349, Software Design Specification, Security considerations in Home Control installations

[12] SD, APL13434, Application Note, FAQ: On the use of S0, S2 and Supervision CC in product design and deployments


SILICON LABS
Do you have an innovative idea to keep the world connected?
Explore Our Featured Products
Energy-friendly solutions for a smarter, more connected world.
Microcontrollers
Wireless & RF
Sensors
Smart.
Connected.
Energy-Friendly.

Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com