Software Design Specification

Security 2 Command Class, version 0.9

<table>
<thead>
<tr>
<th>Document No.:</th>
<th>SDS11274</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td></td>
</tr>
<tr>
<td>Description:</td>
<td>Definition of S2 commands, security algorithms and frame flows for secure singlecast and secure multicast with the optional used of pre-agreed Nonce (&quot;rolling code&quot;).</td>
</tr>
<tr>
<td>Written By:</td>
<td>JRM;JBU;ABR;TRASMUSSEN;JFR;AES;NOBRIOT</td>
</tr>
<tr>
<td>Date:</td>
<td></td>
</tr>
<tr>
<td>Reviewed By:</td>
<td>JRM;JBU;ABR;TRASMUSSEN;AES;JFR;NOBRIOT;NTJ</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>Public</td>
</tr>
</tbody>
</table>

Approved by:

<table>
<thead>
<tr>
<th>Date</th>
<th>CET</th>
<th>Initials</th>
<th>Name</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016-08-26</td>
<td>15:14:24</td>
<td>NTJ</td>
<td>Niels Thybo Johansen</td>
<td></td>
</tr>
</tbody>
</table>

Documentation disclaimer on next page regarding copyright notice, trademark notice, license restrictions warranty/consequential damages disclaimer, warranty disclaimer, restricted rights notice and hazardous applications notice.
DOCUMENTATION DISCLAIMER

Copyright Notice

Copyright © August 23, 2016, Sigma Designs, Inc. and/or its affiliates. All rights reserved.

Trademark Notice

Sigma Designs, Inc. and Z-Wave are the registered trademarks of Sigma Designs, Inc. and/or its affiliates. Other names may be trademarks of their respective owners.

License Restrictions Warranty/Consequential Damages Disclaimer

This documentation is provided under certain restrictions on use and disclosure and is protected by intellectual property laws. You may not license, any part, in any form, or by any means. You may use, copy and re-distribute this documentation, in whole or in part. This permission does not grant the recipient's right to modify information contained in this documentation and redistribute this modified information, in whole or in part. Notwithstanding anything contained to the contrary herein, the creation of any derivative works which affects Z-Wave interoperability, based on this documentation shall be strictly prohibited, unless such derivative works are first submitted to the Z-Wave Alliance for review and approval.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. Sigma Designs and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

Restricted Rights Notice

If this is documentation that is delivered or accessed by the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Any Sigma Designs software, hardware and/or documentation delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs and/or software or documentation, including any integrated software, any programs installed on hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This documentation is developed for general use. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this documentation to create or facilitate the creation of dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Sigma Designs and its affiliates disclaim any liability for any damages caused by use of this documentation in dangerous applications.
## REVISION RECORD

<table>
<thead>
<tr>
<th>Doc. Rev</th>
<th>Date</th>
<th>By</th>
<th>Pages affected</th>
<th>Brief description of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>20160823</td>
<td>NOBRIOT</td>
<td>ALL</td>
<td>Prepared for Public Z-Wave initiative</td>
</tr>
</tbody>
</table>
# Table of Contents

1 ABBREVIATIONS ..............................................................................................................................2

2 INTRODUCTION ...........................................................................................................................3

2.1 Purpose .........................................................................................................................................3

2.2 Terms used in this document ....................................................................................................3

3 COMMAND CLASS DEFINITION ..................................................................................................4

3.1 Security 2 Command Class .......................................................................................................4

3.1.1 Compatibility Considerations ...............................................................................................5

3.1.1.1 Command Class dependencies .........................................................................................5

3.1.1.2 Mixed Security Classes ..................................................................................................5

3.1.1.3 Migration of existing devices to the Security 2 Command Class .......................................5

3.1.2 Security Considerations .........................................................................................................6

3.1.2.1 Application enabled delivery confirmation .......................................................................6

3.1.2.2 Potential Singlecast Delay Attack via interception and jamming ....................................6

3.1.2.3 Potential Multicast Delay Attack .....................................................................................6

3.1.2.4 Circumventing DSK authentication ..................................................................................6

3.1.2.4.1 Controller-side authentication .....................................................................................6

3.1.2.4.2 Client-side authentication ............................................................................................7

3.1.2.5 Protecting keys from physical extraction ..........................................................................7

3.1.3 Interoperability considerations ...............................................................................................7

3.1.3.1 Pragmatic Decryption calculations in constrained environments ......................................7

3.1.4 Building Blocks .......................................................................................................................8

3.1.4.1 Core AES (AES) ..............................................................................................................9

3.1.4.2 AES-128 CCM Encryption and Authentication ..................................................................10

3.1.4.2.1 CCM profile ..................................................................................................................10

3.1.4.2.2 Additional Authenticated Data (AAD) .........................................................................10

3.1.4.3 Message Authentication Code – AES-128 CMAC ............................................................12

3.1.4.4 Pseudo Random Number Generator (PRNG) ..................................................................12

3.1.4.5 Key extraction and derivation ............................................................................................12

3.1.4.5.1 CKDF-TempExtract .....................................................................................................13

3.1.4.5.2 CKDF-TempExpand ......................................................................................................13

3.1.4.5.3 Permanent Key Exchange ............................................................................................14

3.1.4.6 Nonces for CCM ..................................................................................................................15

3.1.4.7 SPAN NextNonce Generator ............................................................................................15

3.1.4.7.1 SPAN Instantiation .........................................................................................................15

3.1.4.7.2 Generation .....................................................................................................................16

3.1.4.8 MPAN NextNonce Generator ...........................................................................................17

3.1.4.8.1 MPAN Generation .........................................................................................................17

3.1.5 Message Encapsulation ..........................................................................................................17

3.1.5.1 Singlecast messages and SPAN Management ...................................................................18

3.1.5.1.1 SPAN Table ....................................................................................................................18

3.1.5.2 Multicast messages and MPAN Management ...................................................................21

3.1.5.2.1 MPAN table ....................................................................................................................21

3.1.5.2.2 Adding an S2 Multicast group member .........................................................................25

3.1.5.2.3 Removing an S2 Multicast group member .....................................................................26

3.1.5.2.4 Handling a missing S2 Multicast frame ........................................................................26

3.1.5.3 Message encapsulation commands ....................................................................................27

3.1.5.3.1 Security 2 Nonce Get Command ..................................................................................27

3.1.5.3.2 Security 2 Nonce Report Command .............................................................................28

3.1.5.3.3 Security 2 Message Encapsulation Command ...............................................................29
3.1.5.4 Duplicate Message Detection ................................................................. 37
3.1.6 Key Management ....................................................................................... 38
  3.1.6.1 Key Exchange ....................................................................................... 39
  3.1.6.2 Device Specific Key and User Verification ............................................ 39
  3.1.6.3 Client-Side Authentication .................................................................... 41
  3.1.6.4 Initial Key Exchange ............................................................................ 42
    3.1.6.4.1 Security 2 bootstrapping Interrupt Points ......................................... 47
    3.1.6.4.2 Security 2 bootstrapping Timeouts .................................................. 48
3.1.7 Security 2 Key Exchange commands ....................................................... 49
    3.1.7.1 Security 2 KEX Get Command ............................................................ 49
    3.1.7.2 Security 2 KEX Report Command ..................................................... 49
    3.1.7.3 Security 2 KEX Set Command ........................................................... 52
    3.1.7.4 Security 2 KEX Fail Command ......................................................... 53
    3.1.7.5 Security 2 Public Key Report Command ............................................. 54
    3.1.7.6 Security 2 Network Key Get Command .............................................. 55
    3.1.7.7 Security 2 Network Key Report Command .......................................... 56
    3.1.7.8 Security 2 Network Key Verify Command .......................................... 56
    3.1.7.9 Security 2 Transfer End Command .................................................... 57
3.1.8 Discovery of Security capabilities commands .......................................... 57
    3.1.8.1 Security 2 Commands Supported Get Command .............................. 58
    3.1.8.2 Security 2 Commands Supported Report Command ........................... 58
    3.1.8.3 Security 2 Capabilities Get Command ............................................... 60
    3.1.8.4 Security 2 Capabilities Report Command ........................................... 60

REFERENCES ........................................................................................................ 61

Table of Figures

Figure 1 AES-ECB (Electronic Code Book) ........................................................... 9
Figure 2, AES-128 CMAC building blocks ............................................................ 12
Figure 3, CKDF-TempExtract function block diagram ........................................ 13
Figure 4, CKDF-TempExpand function block diagram ........................................ 14
Figure 5, CKDF-NetworkKeyExpand function block diagram ................................ 15
Figure 6, MPAN generation ............................................................................... 17
Figure 7 Singlecast communication frame flow: SPAN establishment ..................... 20
Figure 8, Next MPAN calculation example .......................................................... 24
Figure 9, Multicast communication example with receivers out of sync and Supervision 25
Figure 10, DSK text format (example) ................................................................. 40
Figure 11, DSK label and QR code (example) ...................................................... 41
Figure 12, S2 bootstrapping frame flow ............................................................... 43
## ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Symbol used to denote the concatenation of information fields</td>
</tr>
<tr>
<td>AAD</td>
<td>Additional Authenticated Data</td>
</tr>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>CCM</td>
<td>Counter with CBC-MAC</td>
</tr>
<tr>
<td>CKDF</td>
<td>Cipher-based Key Derivation Function</td>
</tr>
<tr>
<td>CMAC</td>
<td>Cipher-based Message Authentication Code</td>
</tr>
<tr>
<td>CSA</td>
<td>Client-side Authentication</td>
</tr>
<tr>
<td>CTR_DRBG</td>
<td>Counter mode Deterministic Random Byte Generator</td>
</tr>
<tr>
<td>DSK</td>
<td>Device Specific Key</td>
</tr>
<tr>
<td>DSK-KDF</td>
<td>DSK-Key Derivation Function</td>
</tr>
<tr>
<td>EI</td>
<td>Entropy Input (used for seeding a new CTR_DRBG)</td>
</tr>
<tr>
<td>ECDH</td>
<td>Elliptic Curve Diffie-Hellman</td>
</tr>
<tr>
<td>IV</td>
<td>Initialization vector</td>
</tr>
<tr>
<td>KEX</td>
<td>Key exchange</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Byte</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
</tr>
<tr>
<td>MGRP</td>
<td>Multicast Group</td>
</tr>
<tr>
<td>MITM</td>
<td>Man-In-The-Middle (security attack)</td>
</tr>
<tr>
<td>MOS</td>
<td>Multicast Out Of Sync</td>
</tr>
<tr>
<td>MPAN</td>
<td>Multicast Pre-Agreed Nonce</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Byte</td>
</tr>
<tr>
<td>Nonce</td>
<td>Number used once</td>
</tr>
<tr>
<td>NWI</td>
<td>Network-Wide Inclusion</td>
</tr>
<tr>
<td>OOB</td>
<td>Out-Of-Band (security authentication method)</td>
</tr>
<tr>
<td>PRK</td>
<td>Pseudo Random Key</td>
</tr>
<tr>
<td>PRNG</td>
<td>Pseudo-Random Number Generator</td>
</tr>
<tr>
<td>QR</td>
<td>Quick Response code</td>
</tr>
<tr>
<td>REI</td>
<td>Receiver’s Entropy Input</td>
</tr>
<tr>
<td>S0</td>
<td>Security 0 Command Class</td>
</tr>
<tr>
<td>S2</td>
<td>Security 2 Command Class</td>
</tr>
<tr>
<td>S2 MC</td>
<td>Security 2 Multicast</td>
</tr>
<tr>
<td>S2 SC</td>
<td>Security 2 Singlecast</td>
</tr>
<tr>
<td>S2 SC-F</td>
<td>Security 2 Singlecast Follow-up for multicast</td>
</tr>
<tr>
<td>SCSG</td>
<td>Security Commands Supported Get</td>
</tr>
<tr>
<td>SCSR</td>
<td>Security Commands Supported Report</td>
</tr>
<tr>
<td>SEI</td>
<td>Sender’s Entropy Input</td>
</tr>
<tr>
<td>SIS</td>
<td>SUC (Node) ID Server</td>
</tr>
<tr>
<td>SOS</td>
<td>Singlecast Out Of Sync</td>
</tr>
<tr>
<td>SPAN</td>
<td>Singlecast Pre-Agreed Nonce</td>
</tr>
<tr>
<td>SRP</td>
<td>Secure Remote Password</td>
</tr>
<tr>
<td>SUC</td>
<td>Static Update Controller</td>
</tr>
</tbody>
</table>
2 INTRODUCTION

This document describes the second generation of Z-Wave transport security, the Security 2 Command Class. The following text uses the short term S2 to represent the Security 2 Command Class.

The Security 2 Command Class builds on publicly accessible security principles. The references [1], [2], [3], [4], [8] provide an introduction to the domain while the references [9], [10], [11], [12], [13], [14] define actual mechanisms that are incorporated in the Security 2 Command Class.

2.1 Purpose

The purpose of this document is to describe the Security 2 Command Class (version 1) in sufficient detail to allow an independent and interoperable implementation.

2.2 Terms used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document MUST be interpreted as described in IETF RFC 2119 [14].
3 COMMAND CLASS DEFINITION

3.1 Security 2 Command Class

The Security 2 Command Class is a framework for allowing nodes to communicate securely in a Z-Wave network.

The Security 2 Command Class provides backwards compatibility to nodes implementing the Security 0 Command Class. Security 2 Command Class also defines a new encapsulation format, new Security Classes and a new KEX Scheme 1, which together offers a number of advantages over the Security 0 Command Class. Security 2 Command Class is scalable and allows more KEX Schemes, Security Classes and encapsulation formats to be introduced in the future if necessary.

Communication may be protected for a number of purposes. Known as CIA, the three main areas addressed by communications security are Confidentiality, Integrity and Authenticity. Confidentiality ensures that only the recipient can decode the communication. Integrity ensures that the recipient can determine if the communication has been modified or replayed. Authentication ensures that the communication really comes from the advertised sender.

Security 2 provides Confidentiality, Integrity and Authentication through:

- Key Exchange, that allows distribution of Network Keys in a Secure Network while preventing interception through:
  - Out-of-Band verification
  - Narrow time windows
  - Physical Activation
- Secure Message Encapsulation between nodes that have a shared network key.
  - A Pre-Agreed Nonce (PAN) is used to prevent Replay Attacks where communication is recorded and played back later, e.g. to unlock a door. In S2 a Nonce is exchanged once and then used to compute subsequent Singlecast PANs (SPAN) and Multicast PANs (MPAN), respectively.
  - Secure Singlecast communication, supports one-frame secure messages; allowing for lower latency and faster response times.
  - Secure Multicast communication, allows a sending node to simultaneously send secure messages to multiple nodes.
- Cryptographic algorithms are intimately dependent on a perfect Pseudo Random Number Generator (PRNG). The PRNG is seeded with a unique noise pattern to ensure a unique starting point in the long sequence of random numbers generated by the PRNG. Many radio systems feature a special mode where the radio is capable of generating white noise which is not affected by the RF signal received from the antenna.
3.1.1 Compatibility Considerations

3.1.1.1 Command Class dependencies

Nodes supporting the Security 2 Command Class MUST also support the following command classes:

- Supervision Command Class [6]

In addition, controllers MUST support the following Command Class:

- Inclusion Controller Command Class [16]

3.1.1.2 Mixed Security Classes

With the advent of the Security 2 Command Class, three new security classes have been added to the existing non-secure and Security 0. This makes for a total of five different security levels of which neither can communicate directly with each other.

In Security 0, this could be alleviated to some extent by allowing a device to choose to support certain of its command classes as non-secure. However, the impact of this is that a device is not entirely secure. For this reason, command classes SHOULD NOT be supported non-securely by S0 enabled nodes if they leak information about the state of the device.

In Security 2, a device MUST NOT support any command classes using anything but its highest Security Class granted during security bootstrapping. This also means S0 and Non-Secure communication MUST NOT be supported by an S2 bootstrapped device that was granted at least one S2 Security Class.

To allow inter-device communication for different security classes, the SIS (or Primary Controller) MAY perform Security Class elevation, by working as a middle-man and thus elevating the Security Class of a sending device to reach a different Security Class on a receiving device.

In case a forwarding rule is created from a lower Class to a higher Class, the UI MUST issue a warning to the user.

A node supporting S2 MAY control Command Classes at any of the granted Security Classes.

A controlling node attempting to communicate with a supporting node SHOULD try using its highest Security Class. If the communication is not successful and the controlling node has been granted several Security Classes, the controlling node MAY try using any lower Security Classes.

3.1.1.3 Migration of existing devices to the Security 2 Command Class

An included node may be upgraded to support S2 via an OTA firmware update. Since non-secure operation as well as the Security 0 Command Class provide a lower level of trust, it is not possible to automatically switch to S2 protection. Instead the Device Specific Key (DSK) MUST be generated by the running firmware on the device after being updated. Having the DSK generated internally, means it is not possible to input it directly on the S2 bootstrapping controller (for Access Control and Authenticated Security Classes), instead it MUST be possible to input the DSK of the S2 bootstrapping controller on the joining node.

This process is described in further details in 3.1.6.3.
3.1.2 Security Considerations

3.1.2.1 Application enabled delivery confirmation

The use of SPAN and MPAN enables secure communication without preparations for each message. It is powerful, yet it requires application awareness. Safety and security related applications like door locks may require immediate command confirmations via the Supervision Command Class. Further, the risk of a delay attack can be mitigated through the use of the Supervision Command Class. This applies to S2 Singlecast as well as S2 Multicast transmissions.

A firmware update process may prefer to transfer firmware fragments as fast as possible while accepting the minor risk that the process stops for a moment in the unlikely event that the SPAN needs to be updated by the transmitter.

3.1.2.2 Potential Singlecast Delay Attack via interception and jamming

Since the SPAN is not limited by a timeout or synchronized clock, it is possible to perform a delay attack by intercepting an encrypted message while at the same time jamming the intended receiver. This way, the receiver SPAN is not incremented and the receiver will accept the encrypted message when an attacker decides to transmit the delayed message. This attack further requires that the attacker returns a MAC layer acknowledgement to the sender to avoid that the user gets error messages.

The Supervision Command Class SHOULD be used for S2 delivery acknowledgement. Only the intended receiver can respond correctly to a Supervision Get command.

3.1.2.3 Potential Multicast Delay Attack

Since the MPAN is not limited by a timeout or synchronized clock, it is possible to perform a delay attack by intercepting an encrypted message while at the same time jamming the intended receivers. This way the receiver MPAN is not incremented and the receivers will accept the encrypted message when an attacker decides to transmit the delayed message.

There is no way to distinguish this attack from simple radio interference phenomena as S2 Multicast messages are not acknowledged on the Z-Wave MAC layer.

While the singlecast follow-up message is primarily intended to ensure command execution in all multicast group members, the message also makes the receiver increment its MPAN inner state to invalidate previous multicast messages. This effectively eliminates an attacker’s options for mounting a multicast delay attack.

S2 Multicast and singlecast follow-up messages are described in 3.1.5.2.

3.1.2.4 Circumventing DSK authentication

3.1.2.4.1 Controller-side authentication

The first 2 bytes of the public key of the joining node are obfuscated when requesting access to the “S2 Access Control” or “S2 Authenticated” class. The purpose of the DSK validation procedure is to force the user to enter information that can only be gathered from the joining device as PIN code or QR code.

In other words, a user entering the PIN code or QR code proves to the including controller that the joining node is indeed the node that the user wants to join. With this information, the including controller can construct the full public key of the joining node.

It is theoretically possible for the including controller to guess the complete public key of the joining node in less than 65536 calculations without user interaction. The including controller needs to try decrypting the received frame until a valid KEX Set (Echo) command is found.
The DSK verification is a device authentication step that ensures that a joining node can be trusted. The security of the system does not depend on the public key being secret but an including controller that skips authentication runs the risk of handing out network keys to joining nodes that cannot be trusted. A man-in-the-middle attacker may establish a shared secret with the joining node by using the joining node’s public key but the user’s including controller will detect a mismatch between the DSK of the joining node and the first 2 or 16 bytes of the attacker’s public key. Therefore the including controller rejects to complete the security bootstrapping before the network key is exposed to the attacker.

This applies to the S2 Access Control Class as well as the S2 Authenticated Class.

3.1.2.4.2 Client-side authentication

When upgrading existing devices to support Security 2 through an over-the-air (OTA) firmware update, there is no DSK printed on the node, which is required for S2 bootstrapping of the S2 Authenticated and S2 Access Control Classes.

In this case, a reverse verification procedure can be carried out, where the controller obfuscates the first 4 bytes of its public key and the joining node is input the controller’s DSK in order to perform the authentication.

3.1.2.5 Protecting keys from physical extraction

A device may be mounted in an outdoor or public location. In such locations, there is a risk that the device is removed physically. The hardware of the device SHOULD be designed to prevent the read-back of ECDH keys and network keys via debug connectors.

The hardware of the device SHOULD be designed to prevent the read-back of ECDH keys and network keys via a malicious firmware image.

The non-volatile memory used for storing security keys SHOULD be automatically cleared in case a firmware image is programmed in the NVM via the debug interface; only allowing keys to survive if firmware update is handled entirely via internal software APIs.

3.1.3 Interoperability considerations

3.1.3.1 Pragmatic Decryption calculations in constrained environments

This specification recommends that a receiving node accepts incoming S2 Multicast frames which are up to 4 iterations into the future relative to the current MPAN inner state for the actual Group ID.

This specification also recommends that a sending node issues S2 Singlecast Follow-up frames after sending S2 Multicast frames.

Decryption calculations may put a significant load on constrained processors. Thus, the receiving node may still be trying to decrypt a received S2 Multicast frame when an S2 Singlecast Follow-up is received.

A receiving node MUST respond correctly to an S2 Singlecast frame received immediately after an S2 Multicast frame; even if that means that the receiving node has to cancel a running decryption attempt.

A receiving node SHOULD monitor the arrival of S2 Multicast frames in order to avoid repeated MPAN synchronization in conditions where the S2 Multicast frame is only rarely received correctly.
3.1.4 Building Blocks

S2 functionality is relying on a number of building blocks for different parts of the protocol:

In the following, the term Node A refers to the including node (typically a primary controller or SIS) while the term Node B refers to the joining node.

- **Inclusion Step 1: Create a shared secret between Node A and Node B**
  Both nodes calculate a shared secret based on an Authenticated Elliptic Curve Diffie Hellman key exchange (AuthECDH). Node A takes as input the Public Key of B, KeyPub_B and its own Private Key, KeyPriv_A. Node B takes as input the Public Key of A, KeyPub_A and its own Private Key, KeyPriv_B. Both returning the same ECDH Shared Secret.
    - **AuthECDH** is based on ECDH using Curve25519 [14]. Authentication is achieved through user verification as specified in Section 3.1.6.2.

- **Inclusion Step 2: Derive shared symmetric key for key exchange**
  To establish a temporary Network Key for AES128-CCM and CTR_DRBG, two steps are needed:
    - To convert the ECDH Shared Secret into a 16-byte Pseudo Random Key (PRK), **CKDF-TempExtract** takes as input the ECDH Shared Secret along with KeyPub_A and KeyPub_B.
    - Temporary symmetric keys are derived based on CKDF-TempExpand, by giving the PRK, KeyPub_A and KeyPub_B as input. This returns the following keys:
      - Temporary CCM Key, combined Encryption and Authentication Key, denoted TempKeyCCM
      - Temporary Personalization String, denoted TempPersonalizationString.

- **Inclusion Step 3: Exchange permanent Network Keys**
  To exchange one or several Permanent Network Key (PNK), Singlecast Message Encapsulation is used with temporary symmetric derived keys (TempKeyCCM and TempPersonalizationString).
    - All Permanent Network Key Exchanges are carried out using the temporary symmetric key.
    - All Permanent CCM Keys, KeyCCM, KeyMPAN and PersonalizationString, are derived from the corresponding PNK using **CKDF-NetworkKeyExpand**
    - All CKDF functions are based on AES128-CMAC.

- **Singlecast Message Encapsulation:**
  The algorithm uses an authenticated encryption scheme conforming to AES128-CCM [11]. It is used to encrypt and authenticate secure payloads. AES128-CCM takes the KeyCCM and a Nonce as input. The algorithm returns a CCM Authenticated Ciphertext.
    - A Nonce is a “Number used Once”. Nonces are initially exchanged between the two nodes and then mixed into a Mixed Entropy Input, MEI, using **CKDF-MEI-Extract** and **CKDF-MEI-Expand**. With both nodes holding the same MEI, they use the MEI and PersonalizationString as input into CTR_DRBG to generate a new Nonce.

- **Multicast Message Encapsulation:**
  Both singlecast and multicast frames use AES-128 CCM for encryption and authentication but
multicast frames use a different algorithm to generate the IV. For multicast, the CCM IV is called the Multicast Pre-Agreed Nonce (MPAN).

MPANs are pushed from Node A, to multiple receiving nodes, B1, B2, etc. MPANs MUST be generated by Node A (see section 3.1.4.8) and MUST be distributed securely to each node B1, B2 etc. using singlecast messages.

Implementations of the Security 2 Command Class MUST provide AES-128 cryptographic services in the following modes of operation:

- **AES-128 “RAW”**
  This is also known as the AES-Electronic Code Book (ECB) and defines the basic operation of encrypting a 16 Byte Plaintext into a 16 Byte Ciphertext using an encryption key. AES-128 is used as a foundation for the following modes.

- **AES-128 CCM**
  The Counter with CBC-MAC (CCM) [11] is an authenticated encryption scheme.

- **AES-128 CMAC**
  The Cipher based Message Authentication Code (CMAC) is an essential part of the Key Derivation functions and used to mix Nonce contributions for SPAN synchronization.

- **AES-128 CTR_DRBG**
  The Counter mode Deterministic Random Byte Generator (CTR_DRBG) [12] is a block cipher based Pseudo Random Number Generator (PRNG) that is used to create Initialization Vectors and Network Keys.

In addition to the AES modes, the following building blocks MUST also be provided:

- **AuthECDH**
  Authenticated Elliptic Curve Diffie Hellman key exchange. Provides the temporary shared key material for protecting the network key exchange during inclusion.

### 3.1.4.1 Core AES (AES)

The security layer MUST implement the AES-128 encryption algorithm. This algorithm encrypts a single 128-bit block of plaintext using the Advanced Encryption Standard, AES. It receives a 128-bit key and a 128-bit plaintext block as input and produces a 128-bit cipher text block.

![Figure 1 AES-ECB (Electronic Code Book)](image-url)
3.1.4.2 AES-128 CCM Encryption and Authentication

The payload in the S2 Message Encapsulation Command MUST be encrypted and authenticated using AES-128 CCM. For details about AES-128 CCM and message decryption and validation, refer to [11] and [13].

CCM takes the following input:

- A combined encryption and authentication key denoted KeyCCM
- A Nonce
- A variable-length additional authenticated data structure (AAD)
- A variable-length payload to encrypt and authenticate

CCM yields as output:

- An encrypted version of the payload combined with an authentication tag covering the payload and the AAD

3.1.4.2.1 CCM profile

[13] defines a number of parameters that together determine the CCM profile used. S2 nodes MUST use the following parameter values for the CCM profile:

- Additional Authenticated Data (AAD) MUST be used (Adata = 1)
  The actual AAD is defined in section 3.1.4.2.2.
- The Length field MUST be 2 bytes long (L = 2 bytes)
- The Authentication tag length MUST be 8 bytes (M = 8 bytes)
- The Nonce length MUST be 13 bytes (N = 13 bytes)

The following length requirements MUST be observed:

- AAD structures of up to 30 bytes MUST be supported. Larger AAD structures MAY be supported.

The 13 byte Nonce MUST be generated by taking the 13 most significant bytes of the NextNonce or Nonce0 as described in section 3.1.4.6.

3.1.4.2.2 Additional Authenticated Data (AAD)

The following data structure MUST be used as AAD input for each CCM operation.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Sender NodeID</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Destination Tag</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>HomeID Byte 1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>HomeID Byte 4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Message Length Byte 1 (MSB)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Message Length Byte 2 (LSB)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>Enc Ext</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ext</td>
</tr>
<tr>
<td></td>
<td>Extension Data 1</td>
<td></td>
</tr>
</tbody>
</table>
Sender NodeID (1 byte)
NodeID of the sending node.

Destination Tag (1 byte)
The use of this field depends on the actual frame.
If the field is used for a Singlecast frame, this field MUST carry the Receiver NodeID.
If the field is used for an S2 Multicast frame, this field MUST carry the S2 Multicast Group ID.

HomeID (4 bytes)
HomeID of the sending node.

Message length (2 bytes)
This field indicates the total length in bytes of the Security 2 Message Encapsulation Command.

Sequence Number (1 byte)
Refer to the Security 2 Message Encapsulation Command (3.1.5.3.3).

Ext (1 bit)
Refer to the Security 2 Message Encapsulation Command (3.1.5.3.3).

Enc Ext (1 bit)
Refer to the Security 2 Message Encapsulation Command (3.1.5.3.3).

Reserved
Refer to the Security 2 Message Encapsulation Command (3.1.5.3.3).

Extension Data (M bytes)
This field MUST contain all non-encrypted extension objects.
This field MUST include the Length and Type fields prepending the actual data of each extension.
3.1.4.3 Message Authentication Code – AES-128 CMAC

AES-128 CMAC is used for key derivation and Nonce mixing. The CMAC operation is specified in [10].

3.1.4.4 Pseudo Random Number Generator (PRNG)

The PRNG MUST be used for:

- Generating new network keys when provisioning a new network.
- Generating Nonce contributions for synchronizing the SPAN with peer nodes.

The PRNG MUST be implemented as an AES-128 CTR_DRBG as specified in [12]. The following profile MUST be used:

- No derivation function
- No reseeding counter
- Personalization string of 0x00 repeated 32 times
- Output length = 16 bytes
- security_strength is not used

The entropy_input [12] for instantiating the PRNG MUST be generated by a truly random source, e.g. white radio noise. The PRNG MUST be hardware seeded.

The inner state of the PRNG MUST be separated from the SPAN table.

3.1.4.5 Key extraction and derivation

The functions described in this section are used during key exchange.
3.1.4.5.1 CKDF-TempExtract

The **CKDF-TempExtract** function is used to extract the key entropy from the non-uniformly distributed ECDH Shared Secret.

\[
\text{CKDF-TempExtract} (\text{Constant}_{PRK}, \text{ECDH Shared Secret}, KeyPub_A, KeyPub_B) \rightarrow PRK
\]

- The function’s input is defined by:
  - Constant\textsubscript{PRK} = 0x33 repeated 16 times
  - ECDH Shared Secret is the output of the ECDH key exchange
  - Public Keys of Nodes A and B
  - \( PRK = \text{CMAC}(\text{Constant}_{PRK}, \text{ECDH Shared Secret} | KeyPub_A | KeyPub_B) \)

![Figure 3, CKDF-TempExtract function block diagram](image)

3.1.4.5.2 CKDF-TempExpand

Once the PRK has been computed, the temporary Authentication, Encryption and Nonce Keys MUST be derived using the **CKDF-TempExpand** function [9].

\[
\text{CKDF-TempExpand} (PRK, Constant_{TE}, KeyPub_A, KeyPub_B) \rightarrow \{\text{TempKeyCCM}, \text{TempPersonalizationString}\}
\]

- The function’s input is defined by:
  - PRK is calculated in the previous section 3.1.4.5.1
  - Constant\textsubscript{TE} = 0x88 repeated 15 times
  - KeyPub\textsubscript{A} = The ECDH Public Key of the including node
  - KeyPub\textsubscript{B} = The ECDH Public Key of the joining node

- Calculations are performed as follows:
  - \( T1 = \text{CMAC}(PRK, KeyPub_A | KeyPub_B | Constant_{TE} | 0x01) \)
  - \( T2 = \text{CMAC}(PRK, T1 | Constant_{TE} | 0x02) \)
  - \( T3 = \text{CMAC}(PRK, T2 | Constant_{TE} | 0x03) \)

- Output is defined as follows:
  - TempKeyCCM = T1. Temporary CCM Key, combined Encryption and Authentication Key.
  - TempPersonalizationString = T2 | T3
A node that has been security bootstrapped into the network is characterized by being in possession of one or more Network Key(s). This key(s) is used throughout the lifetime of the network, typically measured in years to decades. Different nodes MAY have different Class Keys. A central controller node MUST manage all the keys and select at inclusion time which class key(s) to share with a given node.

Network Keys for all Security 2 Classes MUST be 16 random bytes, generated by using the PRNG function described in section 3.1.4.4.

### 3.1.4.5.3.1 Key Derivation

Once the Network Key(s) has been exchanged, the KeyCCM, PersonalizationString and KeyMPAN values MUST be derived using the `CKDF-NetworkKeyExpand` function [9].

CKDF-NetworkKeyExpand (PNK, ConstantNK) \(\rightarrow\) \{KeyCCM, PersonalizationString, KeyMPAN\}

- **The function’s input is defined by:**
  - PNK is the permanent network key.
  - ConstantNK = 0x55 repeated 15 times
- **Calculations are performed as follow:**
  - T1 = CMAC(PNK, ConstantNK | 0x01)
  - T2 = CMAC(PNK, T1 | ConstantNK | 0x02)
  - T3 = CMAC(PNK, T2 | ConstantNK | 0x03)
  - T4 = CMAC(PNK, T3 | ConstantNK | 0x04)
- **Output is obtained by:**
  - KeyCCM = T1; CCM Key, combined Encryption and Authentication
  - PersonalizationString = T2 | T3
  - KeyMPAN = T4
### 3.1.4.6 Nonces for CCM

The Nonce is used for the CCM encryption in a Security 2 Message Encapsulation Command. Nonces provide these security benefits:

- **Replay protection.** Replaying an old message will result in the replay being discarded because the Nonce has already been used.

- **The security proofs for CCM rely on the assumption that the same plaintext is never encrypted under the same key twice.** Having unique Nonces upholds this assumption.

Singlecast and Multicast transport use different mechanisms for generating Nonces as described in the following sections.

**Singlecast Pre-Agreed Nonces** are referred to as **SPAN**. **Multicast Pre-Agreed Nonces** are referred to as **MPAN**.

### 3.1.4.7 SPAN NextNonce Generator

**Singlecast Nonces** MUST be generated in the following way:

1. **Skip steps 1 through 3 if a SPAN is already established.**
2. Exchange 16 bytes of Entropy between the Sender and the Receiver
3. Mix the entropy contributions
4. Instantiate CTR_DRBG and store the working state in the SPAN table
5. Generate 13 bytes of Nonce from the established SPAN using the `NextNonce` function.
6. Save the updated working state in the SPAN table

#### 3.1.4.7.1 SPAN Instantiation

The Sender and Receiver MUST instantiate the CTR_DRBG as follows:

1. The Sender and Receiver MUST exchange 16 bytes of Entropy Input (EI), resulting in 32 bytes of shared entropy
   a. Both contributions MUST be generated using the PRNG (see Section 3.1.4.4)
2. Mix the 32 bytes Ei into MEI, using CKDF-MEI-Extract and CKDF-MEI-Expand functions

   The CTR_DRBG MUST be instantiated using the following profile:
   a. Entropy Input = MEI (obtained with CKDF-MEI_Expand)
   b. Personalization_String = PersonalizationString
   c. Output length = 16
   d. No derivation function
   e. No reseeding counter
   f. No Security_strength

3. The CTR_DRBG now has its inner state set, referred to as the InnerSPAN

3.1.4.7.1.1 Mixing

Mixing of the two EIs is done by first calling CKDF-MEI-Extract with the EIs as input and using the result as input for CKDF-MEI-Expand

3.1.4.7.1.1.1 CKDF-MEI-Extract

CKDF-MEI-Extract(ConstNonce, SenderEI | ReceiverEI) -> NoncePRK

   - The Input is defined by:
     o ConstNonce = 0x26 repeated 16 times
   - The Output is obtained by:
     o NoncePRK = CMAC(ConstNonce, SenderEI | ReceiverEI)

3.1.4.7.1.1.2 CKDF-MEI-Expand

CKDF-MEI-Expand(NoncePRK, ConstEntropyInput) -> MEI

   - The Input is defined by:
     o NoncePRK is the pseudo random value obtained in the Extract step.
     o ConstEntropyInput = 0x88 repeated 15 times
   - The Output is obtained by:
     o T0 = ConstEntropyInput | 0x00
     o T1 = CMAC(NoncePRK, T0 | ConstEntropyInput | 0x01)
     o T2 = CMAC(NoncePRK, T1 | ConstEntropyInput | 0x02)
     o MEI = T1 | T2

3.1.4.7.2 Generation

   With the CTR_DRBG having its inner state set, new SPANs MUST now be generated by subsequent calls to the NextNonce function.

3.1.4.7.2.1 NextNonce

   The NextNonce function is defined as the CTR_DRBG_Generate_algorithm [12] with the following parameters:
   - working_state = InnerSPAN
   - requested_number_of_bits = 128
   - additional_input = "" (empty string, i.e. zero bytes)

   The NextNonce function MUST return a new SPAN for each call. The InnerSPAN MUST be stored in the SPAN table as described in Section 3.1.5.1.1.
3.1.4.8  MPAN NextNonce Generator

Multicast Pre-Agreed Nonces (MPAN) are generated by encrypting successive values of a counter. The initial value MUST be a 16 byte random number generated by the PRNG (3.1.4.4). Whenever an MPAN is generated and consumed by the CCM module, the inner MPAN state is incremented.

MPAN instantiation and synchronization is described in section 3.1.5.2.

3.1.4.8.1  MPAN Generation

MPAN generation is needed when a node sends or receives a secure multicast message. The generation procedure, called NextMPAN, MUST comply with the following description:

1. The node reads the 16-byte inner MPAN state N from the MPAN table.
2. The node performs an AES-128-ECB encryption of the inner MPAN state N using KeyMPAN (obtained during key derivation 3.1.4.5.3.1) as key. The result is MPAN N.
3. The node increments the inner MPAN state N. The result is the inner MPAN state N+1 which is stored in the MPAN table. The increment operation MUST treat the inner MPAN state as an unsigned 16-byte integer. Thus, incrementing all ones MUST yield all zeros.

![Figure 6, MPAN generation](image)

3.1.5  Message Encapsulation

The Security 2 Command Class supports Singlecast as well as Multicast communication.

The S2 Transport Layer MUST NOT provide retransmission if the security layer discards a message due to SPAN synchronization failure or failed authentication.

An application SHOULD use the Supervision Command Class for delivery acknowledgement of Security 2 Encapsulated commands.

The Supervision Report command returns high-level status information on the execution status of the transmitted command which SHOULD be used by a controlling application instead of polling the destination node repeatedly.
3.1.5.1 Singlecast messages and SPAN Management

Compared to the Security 0 Command Class, the Security 2 Command Class provides a more efficient way to communicate securely. The Security 2 Command Class eliminates the frame overhead of the Security 0 Command Class challenge-response handshake.

Singlecast transport MUST comply with the steps described below:

1. Before sending an encrypted frame, the **Sender** MUST establish a Singlecast Pre-Agreed Nonce (SPAN) with the **Receiver** if a SPAN is not already established.
   a. If a SPAN exists between the two nodes:
      i. Skip to step 3
   b. If the **Sender** is in possession of a matching **Receiver’s Entropy Input**:
      i. The **Sender** uses its PRNG to generate a random 16-byte Nonce (SEI).
      ii. The **Sender** uses the combined 32-byte Sender’s and Receiver’s Entropy Input to instantiate a **NextNonce** Generator, hence creating the inner SPAN state. (described in 3.1.4.7)
      iii. Skip to step 3
   c. If no SPAN or matching **Receiver’s Entropy Input** exists between the two parties:
      i. The **Sender** holds back the frame for subsequent transmission.
      ii. The **Sender** sends a Nonce Get to the **Receiver**

2. The **Receiver** uses its PRNG to generate a random 16-byte Nonce, store it in the SPAN table and send it in a Nonce Report to the **Sender**, with the Singlecast Out of Sync (SOS) flag set, to indicate that the frame contains a Receiver’s Entropy Input (REI).
   a. The **Sender** stores the Receiver’s Entropy Input in the SPAN table in the entry matching the receiver’s NodeID.
   b. If the **Sender** has a pending frame for transmission then proceed to step 3. Otherwise no further action is taken.

3. The **Sender** constructs a Security 2 Message Encapsulation Command.
   a. If the inner SPAN state was just created in step 2.a, the SPAN extension containing the SEI is added to the S2 Message encapsulation Command to allow the **Receiver** to compute the same inner SPAN state.
   b. The **Sender** creates the next SPAN with the **NextNonce** function.
   c. The **Sender** uses the SPAN as IV for the AES-128 CCM encryption and authentication (3.1.4.2) of the plaintext payload using **KeyCCM**.
   d. The Security 2 Message Encapsulation Command is transmitted to the **Receiver**.

4. The **Receiver** inspects the received Security 2 Message Encapsulation Command.
   a. If the **Receiver** is unable to authenticate the singlecast message with the current SPAN, the **Receiver** SHOULD try decrypting the message with one or more of the following SPAN values, stopping when decryption is successful or the maximum number of iterations is reached. The maximum number of iterations performed by a receiving node MUST be in the range 1..5.
      If the maximum number of iterations is reached without successful decryption, a Nonce Report MUST be sent to the **Sender** with the SOS flag set and containing a new REI. At the same time, the **Receiver** MUST invalidate the SPAN table entry for the Sender NodeID.
b. If a SPAN Extension is present, the Receiver MUST:
   
i. Instantiate a new SPAN Generator using the Receiver’s Entropy Input stored locally and the Sender’s Entropy Input just received.
   
   ii. Store the inner SPAN state in a SPAN table entry with the Sender as Peer NodeID.
   
   iii. Generate a SPAN by running the \textit{NextNonce} function on the newly instantiated inner SPAN state.
   
   iv. Attempt authentication with the SPAN.
   
   v. If the authentication succeeds, skip to step 5.
   
   vi. If the authentication fails, the Receiver MUST go to step 2.

   c. If the SPAN is already established and a SPAN Extension is not present, the Receiver MUST calculate a new inner SPAN state by passing the old inner SPAN state through the \textit{NextNonce} function.

5. After the Receiver has successfully authenticated the encapsulation command, the decrypted payload can be presented to the application layer.

The process is also illustrated in Figure 7.
Figure 7 Singlecast communication frame flow: SPAN establishment
3.1.5.1.1 SPAN Table

A receiving S2 node MUST accept a SPAN table update even if the SPAN table is full. It is RECOMMENDED that the node discards the least recently used entry from the SPAN table to make room for the new entry.

The SPAN table MUST support at least 5 entries. The following format is RECOMMENDED:

Table 1, SPAN table format

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Security key</td>
<td>Nonce Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAN State Byte 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPAN State Byte 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Peer NodeID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Security key (4 bits)

This field is used to remember which to Security key the SPAN is associated to. An example is given in Table 2.

Table 2, SPAN table::Security key

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ECDH Temporary Key</td>
</tr>
<tr>
<td>0x01</td>
<td>S2.2: Access Control</td>
</tr>
<tr>
<td>0x02</td>
<td>S2.1: Authenticated</td>
</tr>
<tr>
<td>0x03</td>
<td>S2.0: Unauthenticated</td>
</tr>
<tr>
<td>0x04</td>
<td>S0</td>
</tr>
</tbody>
</table>

Nonce Type (2 bits)

This field is used to advertise the format of the SPAN State field.

The field SHOULD be encoded according to Table 3.
Table 3, SPAN table:: Nonce Type

<table>
<thead>
<tr>
<th>Nonce Type</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Free</td>
<td>Table entry is not in use.</td>
</tr>
<tr>
<td>0x01</td>
<td>Receiver's Entropy Input</td>
<td>Bytes 1-16 are set to zero Bytes 17-32 contain the Receiver’s Entropy Input</td>
</tr>
<tr>
<td></td>
<td>(Locally generated)</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>SPAN State</td>
<td>Bytes 1-32 contain the SPAN state</td>
</tr>
</tbody>
</table>

Sequence Number (1 byte)

This field is used to store the sequence number. Refer to description in 3.1.5.3.3 Security 2 Message Encapsulation Command.

A receiving node MUST validate the Sequence Number and the actual sender’s NodeID before accepting a SPAN table update.

SPAN State (32 bytes)

When the Nonce Type is “SPAN state”, this contains the internal state of the NextNonce Generator. When the Nonce Type is Receiver’s Entropy Input this field contains a locally generated Nonce awaiting the matching Sender’s Entropy Input.

Peer NodeID (1 byte)

This field is used to store the NodeID of the corresponding peer.

3.1.5.2 Multicast messages and MPAN Management

The S2 Multicast protocol allows a sending node to reach a group of nodes by using pre-agreed information for authentication and encryption.

The following terminology is used in this section:

- S2 SC = S2 Singlecast (carried in Z-Wave singlecast frame)
- S2 MC = S2 Multicast (carried in Z-Wave broadcast frame, with the MGRP extension)
- S2 SC-F = S2 Singlecast Follow-up (carried in Z-Wave singlecast frame, with the MGRP extension)

An S2 Multicast group MUST be represented by a unique (sender NodeID, Group ID) combination. A multicast sender or group owner MAY maintain multiple S2 Multicast groups. Sending and receiving nodes MUST maintain a unique Multicast Pre-Agreed Nonce (MPAN) value for each Group ID. The MPAN is sender specific, and MUST NOT be used by the receiver to send S2 Multicast frames. S2 Multicast can only be performed within a group of nodes in the same S2 Security Class.

All nodes in the network will receive the S2 Multicast frame and will not try to decrypt it if the MPAN of the corresponding (sender NodeID, Group ID) is not known. The S2 Multicast protocol uses S2 SC-F frames for MPAN synchronization.

A sending node SHOULD send an S2 SC-F frame to each S2 Multicast group member after sending an S2 MC frame. This ensures that all group members receive the frame and at the same time eliminates the risk of the S2 MC frame being replayed in a delay attack.

S2 SC-F SHOULD be sent frequently in order to ensure that MPAN is synchronized at every group member.
S2 Multicast employs a self-healing algorithm for MPAN synchronization. The reception of an S2 SC-F allows the receiving node to return an MPAN Out of Sync (MOS) indication (either using a Nonce Report with the MOS flag or a S2 Message Encapsulation with the MOS extension, refer to 3.1.5.3.2, 3.1.5.3.3.1.4) if necessary. Thus, a sending node SHOULD NOT push an up-to-date MPAN before sending an S2 Multicast frame to a new S2 Multicast receiver.

A sending node MUST maintain the MPAN inner state according to Table 4.

**Table 4, Sending node MPAN maintenance**

<table>
<thead>
<tr>
<th>Frame type</th>
<th>Description</th>
<th>MPAN increase after transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 Singlecast</td>
<td>S2 SC frame</td>
<td>0</td>
</tr>
<tr>
<td>S2 Unacknowledged Multicast</td>
<td>S2 MC frame</td>
<td>1</td>
</tr>
<tr>
<td>S2 Acknowledged Multicast</td>
<td>S2 MC frame followed by an S2 SC-F frame to each S2 Multicast group member.</td>
<td>2</td>
</tr>
</tbody>
</table>

A receiving node MUST maintain the MPAN inner state according to Table 5.

**Table 5, Receiving node MPAN maintenance**

<table>
<thead>
<tr>
<th>Frame type</th>
<th>Description</th>
<th>MPAN increase after reception</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 Singlecast</td>
<td>S2 SC frame</td>
<td>0</td>
</tr>
<tr>
<td>S2 Multicast</td>
<td>S2 MC frame</td>
<td>1</td>
</tr>
<tr>
<td>S2 Singlecast Follow-up</td>
<td>S2 SC-F frame</td>
<td>1</td>
</tr>
</tbody>
</table>

If the Receiver is unable to decrypt the S2 MC frame with the current MPAN, the Receiver MAY try decrypting the frame with one or more of the subsequent MPAN values, stopping when decryption is successful or the maximum number of iterations is reached.

The maximum number of iterations performed by a receiving node MUST be 5.

If the maximum number of iterations is reached without successful decryption, the MOS flag MUST be set for the actual Group ID. The MOS state is reported back to the sending node only if the node receives a SC-F for the actual group.

A node MUST support being member of 5 multicast groups at the same time.

MPAN synchronization and state maintenance examples are given in Figure 8 and Figure 9.
Figure 8, Next MPAN calculation example
Figure 9, Multicast communication example with receivers out of sync and Supervision

3.1.5.2.1 MPAN table
A node SHOULD maintain the information indicated in Table 6 for each multicast group. The actual implementation format is out of scope of this specification.

Table 6, MPAN table entry, example

<table>
<thead>
<tr>
<th>Information</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner NodeID</td>
<td>The NodeID of the node distributing this MPAN (rx only)</td>
</tr>
<tr>
<td>Group ID</td>
<td>Unique ID chosen by the owner node</td>
</tr>
<tr>
<td>MPAN Inner state</td>
<td>MPAN inner state used for encryption and decryption</td>
</tr>
<tr>
<td>MPAN entry state</td>
<td>MPAN_FREE: This entry is currently not in use</td>
</tr>
<tr>
<td></td>
<td>MPAN_USED: This entry is currently in use</td>
</tr>
<tr>
<td></td>
<td>MPAN_MOS: This entry is out of sync</td>
</tr>
<tr>
<td></td>
<td>Waiting to send Nonce Report in response to singlecast follow-up</td>
</tr>
</tbody>
</table>
3.1.5.2.2 Adding an S2 Multicast group member
A group member can be added to a S2 Multicast group by sending a Singlecast follow-up message to the new NodeID after a S2 Multicast message. The newly added NodeID will notify the sender that it is not synchronized for the given Group ID and it will be synchronized when receiving the next singlecast frame.

3.1.5.2.3 Removing an S2 Multicast group member
A node can be removed from an S2 Multicast group by shuffling the MPAN state. At the next S2 Multicast frame, the newly removed node will set in MOS state and wait for re-synchronization in the next singlecast messages. When the newly removed node receives subsequent singlecast messages without MPAN extension, the newly removed node MUST forget about the Multicast group ID.

3.1.5.2.4 Handling a missing S2 Multicast frame
An S2 MC frame may be missing due to simple radio interference or a deliberate delay attack.

In either case, the MPAN needs to be re-synchronized. Further, the MPAN needs to be advanced immediately to close the time window where an attacker can replay the stolen S2 MC frame.

The transmission of a singlecast follow-up frame ensures re-synchronization and prevents a potential delay attack. Figure 8 outlines an example of the MPAN state changes when an S2 MC frame is missing.

In the case both multicast and singlecast follow-up frames are intercepted for a delay attack, the attacker has the possibility to replay the multicast frame at a later time. Using the Supervision Command Class in singlecast follow-up frames prevent this attack as the receiving node needs to report the application level status in a new singlecast frame.
3.1.5.3 Message encapsulation commands

This section presents the commands of the Security 2 Command Class used for message encapsulation.

3.1.5.3.1 Security 2 Nonce Get Command

This command is used to request a fresh Nonce.

The Security 2 Nonce Report Command MUST be returned in response to this command.

A node sending this command MUST accept a delay up to \(<\text{Previous Round-trip-time to peer node}> + 250 \text{ ms}\) before receiving the Security 2 Nonce Report Command.

This command MUST NOT be issued via multicast addressing. A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

A sending node MUST specify a unique sequence number starting from a random value. Each message MUST carry an increment of the value carried in the previous outgoing message.

A receiving node MUST validate the Sequence Number and the actual sender’s NodeID before accepting this command.

A receiving node MUST use this field for duplicate detection. Refer to section 3.1.5.4.
3.1.5.3.2 Security 2 Nonce Report Command

This command is used to advertise a fresh Nonce in preparation for secure communication. This command MUST NOT be issued unless it is done in response to the S2 Nonce Get or the S2 Message Encapsulation Command.

A sending node MUST set at least one of the MOS and SOS flags to the value ‘1’ in this command. The command MUST NOT be sent if both the MOS and SOS flags are cleared.

A receiving node MUST update the MPAN and/or SPAN of the node sending this command by adding a SPAN and/or MPAN extension to the next Security 2 Message Encapsulation Command.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2_NONCE_REPORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>MOS</td>
<td>SOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver's Entropy Input Byte 1 (Optional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver's Entropy Input Byte 16 (Optional)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sequence Number (1 byte)

A sending node MUST specify a unique sequence number starting from a random value. Each message MUST carry an increment of the value carried in the previous outgoing message.

A receiving node MUST use this field for duplicate detection. Refer to section 3.1.5.4.

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

SOS – SPAN out of Sync (1 Bit)

When set by a sending node, the value ‘1’ MUST indicate that the sending node does not have a SPAN established for the receiving node or was unable to decrypt the most recently received singlecast Security 2 Message Encapsulation Command from the destination of this command.

The value ‘0’ MUST indicate that there was no problem decrypting the most recently received singlecast Security 2 Message Encapsulation Command.

If the SOS flag is set to ‘1’, the REI field MUST be included in the command.

If the SOS flag is set to ‘0’, the REI field MUST NOT be included in the command.

A receiving node MUST establish a new Singlecast Pre-Agreed Nonce (SPAN) and return a Security 2 Message Encapsulation Command with the SPAN extension, if this flag is set to ‘1’ by a sending node.
MOS – MPAN Out of Sync (1 Bit)

When set by a sending node, the value ‘1’ MUST indicate that the sending node does not have a MPAN state for Multicast group used in the most recently received singlecast follow-up Security 2 Message Encapsulation Command from the destination of this command.

The value ‘0’ MUST indicate that there was no problem decrypting the most recently received multicast Security 2 Message Encapsulation Command that was followed by a singlecast follow-up.

A sending node MUST NOT advertise the MOS flag for a given (source NodeID, Group ID) tuple more than one time.

If this flag is set to ‘1’ by a sending node, a receiving node MUST transfer the Multicast Pre-Agreed Nonce (MPAN) of the most recent singlecast follow-up transmission in a subsequent singlecast message if the sending node belongs to the multicast group. The MPAN MUST be transferred by sending a singlecast Security 2 Message Encapsulation Command with the MPAN extension.

Receiver’s Entropy Input (REI) (16 Bytes)

This field is optional. When present, this field is used to carry the Receiver’s Entropy Input in preparation for new S2 transmissions based on the SPAN. Refer to 3.1.5.1.

If the SOS flag is set to ‘1’, the REI field MUST be included in the command.
If the SOS flag is set to ‘0’, the REI field MUST NOT be included in the command.

3.1.5.3.3 Security 2 Message Encapsulation Command

The Security 2 Nonce Report Command MUST be returned in response to this command if the receiving node fails decrypting the command or if the message contains an MGRP extension with a Group ID that is unknown or out of sync (MOS).

This command MAY be issued via Z-Wave Multicast addressing. A receiving node MUST NOT return a response if this command is received via Z-Wave Multicast addressing. The Z-Wave Multicast frame and the broadcast NodeID are both considered Z-Wave Multicast addressing methods.
<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Security Header = SECURITY_2_MESSAGE_ENCAPSULATION

<table>
<thead>
<tr>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extension 1 Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>More to follow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extension 1 Type</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Extension 1 Byte 1</th>
</tr>
</thead>
</table>

...|

<table>
<thead>
<tr>
<th>Extension 1 Byte K</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Extension 2 Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>More to follow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extension 2 Type</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Extension 2 Byte 1</th>
</tr>
</thead>
</table>

...|

<table>
<thead>
<tr>
<th>Extension 2 Byte L</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Encrypted Extension 1 Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>More to follow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encrypted Extension 1 Type</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Encrypted Extension 1 Byte 1</th>
</tr>
</thead>
</table>

...|

<table>
<thead>
<tr>
<th>Encrypted Extension 1 Byte M</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Encrypted Extension 2 Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>More to follow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encrypted Extension 2 Type</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Encrypted Extension 2 Byte 1</th>
</tr>
</thead>
</table>

...|

<table>
<thead>
<tr>
<th>Encrypted Extension 2 Byte N</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CCM Ciphertext object Byte 1</th>
</tr>
</thead>
</table>

...|

| CCM Ciphertext object byte P |

**Sequence Number (1 byte)**

A sending node MUST specify a unique sequence number starting from a random value. Each new message MUST carry an increment of the value carried in the previous message.

A receiving node MUST use this field for duplicate detection. Refer to section 3.1.5.4.
Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node. However, the value MUST be used as part of the AAD input (3.1.4.2.2) used for authentication of the frame by the sending node as well as the receiving node.

Extension (1 bit)

This field is used to indicate if one or more non-encrypted extensions are included.

The value ‘1’ MUST indicate that non-encrypted extensions are included. The value ‘0’ MUST indicate that non-encrypted extensions are not included.

Refer to the Extension object field description.

Encrypted Extension (1 bit)

This field is used to indicate if one or more encrypted extensions are included.

The value ‘1’ MUST indicate that encrypted extensions are included. The value ‘0’ MUST indicate that encrypted extensions are not included.

Refer to the Encrypted Extension object field description.

[Extension Object] (N instances)

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Extension Length

This field specifies the length of this extension, in bytes, including the “Extension Length” field.

Type (6 bit)

This field defines the type of this extension. Valid extension types are listed in 3.1.5.3.3.1.

Critical (1 bit)

A receiving node MUST discard the entire command if this flag is set to ‘1’ and the Type field advertises a value that the receiving node does not support.

If this flag is set to ‘0’ and the Type field advertises a value that the receiving node does not support, the actual extension MUST be ignored.

A receiving node SHOULD continue processing of the encapsulation command after the discarded extension.
More to Follow (1 bit)

If the More to Follow flag is set to ‘1’, another Extension Object MUST follow this Extension Object.

Extension (Variable length)

This field carries the actual extension. Refer to 3.1.5.3.1. The length of this field MUST comply with the length specified in the Extension Length field of this Extension Object.

[Encrypted Extension Object] (N instances)

The format of this object is identical to the unencrypted Extension Object. However, this object MUST be part of the encrypted payload.

CCM Ciphertext Object (P bytes)

This field contains an encrypted message. It comprises:

- (Optional) Complete Z-Wave command (comprising Command Class Identifier, Command Identifier and optional command payload)
- CCM control data
- CCM authentication tag.

The preceding Encrypted Extension fields are technically also a part of the CCM ciphertext object. But conceptually they are separate from the message and have been described as such.

3.1.5.3.1 Valid Extensions and Encrypted Extensions

The defined extension types are listed in Table 7.

<table>
<thead>
<tr>
<th>Extension Type Identifier</th>
<th>Format</th>
<th>Content protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>SPAN Extension</td>
<td>Not Encrypted</td>
</tr>
<tr>
<td>0x02</td>
<td>MPAN Extension</td>
<td>Encrypted</td>
</tr>
<tr>
<td>0x03</td>
<td>MGRP Extension</td>
<td>Not Encrypted</td>
</tr>
<tr>
<td>0x04</td>
<td>MOS Extension</td>
<td>Not Encrypted</td>
</tr>
</tbody>
</table>

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.
3.1.5.3.3.1.1 SPAN Extension

The SPAN Extension is used by the sender to establish a SPAN by sending a Sender’s Entropy Input to the Receiver. The combined Sender’s and Receiver’s Entropy Input may then be passed through the NextNonce function to generate the SPAN.

This extension MUST be sent unencrypted.

<table>
<thead>
<tr>
<th>Length (1 byte)</th>
<th>Critical (1 bit)</th>
<th>Type = SPAN Extension (6 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

| Sender’s Entropy Input Byte 1 |
| ...                          |
| Sender’s Entropy Input Byte 16 |

Length (1 byte)
This field MUST be set to 18.

Type (6 bits)
This field MUST be set to the SPAN Extension type (1).

Critical (1 bit)
This flag MUST be set to ‘1’.

More to Follow (1 bit)
If this flag is set to ‘1’, another unencrypted Extension Object MUST follow this unencrypted Extension Object.
If this flag is set to ‘0’, this MUST be the last unencrypted Extension Object.

Sender’s Entropy Input (16 bytes)
This field MUST carry the entropy input contribution used to instantiate a NextNonce Generator.
3.1.5.3.3.1.2 MPAN Extension

The MPAN Extension is used by the sender to establish or update an MPAN by sending the full 16 bytes MPAN state to the receiver to enable the reception of encrypted multicast transmissions.

This extension MUST be sent encrypted.

<table>
<thead>
<tr>
<th>Length (1 byte)</th>
<th>More to follow</th>
<th>Critical</th>
<th>Type = MPAN Extension = 2</th>
<th>Group ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

### Inner MPAN state (16 bytes)

This field carries the value to be used in the encryption and decryption of the next S2 Multicast frame.
3.1.5.3.3.1.3 MGRP Extension

The Multicast Group (MGRP) Extension MUST be included in all S2 Multicast and S2 Singlecast follow-up frames.

A receiving node MUST use the Group ID to select the MPAN for decrypting this message. When receiving this extension, the matching MPAN (if found) MUST be incremented by one after decryption.

If the Group ID is not found in the receiver MPAN table and the received frame is a singlecast (follow-up), the node MUST return a Nonce Report with the MOS flag set, or alternatively, return another Security 2 Message Encapsulation Command with the MOS extension included.

This extension MUST NOT be sent together with the MPAN extension.

This extension MUST be sent unencrypted.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>More to follow</td>
<td>Critical = 1</td>
<td>Type = MGRP Extension = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Length (1 byte)

This field MUST be set to 3.

Type (6 bits)

This field MUST be set to the MGRP Extension type (3).

Critical (1 bit)

This flag MUST be set to ‘1’.

More to Follow (1 bit)

If this flag is set to ‘1’, another unencrypted Extension Object MUST follow this unencrypted Extension Object. If this flag is set to ‘0’, this MUST be the last unencrypted Extension Object.

Group ID (1 byte)

This field is used by the sender to uniquely identify which MPAN was used to encrypt this multicast frame.
3.1.5.3.3.1.4  MOS Extension

The MOS extension is used to indicate that the sending node does not have a MPAN state for the Multicast group used in the most recently received singlecast follow-up Security 2 Message Encapsulation Command from the destination of this command. The receiver MAY choose to re-synchronize the sending node by returning a new Security 2 Message Encapsulation Command with a MPAN extension included.

This extension MUST be sent unencrypted.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Length = 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More to follow  Critical  Type = MOS Extension = 4

**Length (1 byte)**

This field MUST be set to 2.

**Type (6 bits)**

This field MUST be set to the MOS Extension type (4).

**Critical (1 bit)**

This flag MUST be set to ‘0’.

**More to Follow (1 bit)**

If this flag is set to ‘1’, another unencrypted Extension Object MUST follow this unencrypted Extension Object. If this flag is set to ‘0’, this MUST be the last unencrypted Extension Object.
3.1.5.4 Duplicate Message Detection

A sending node MUST set the Sequence Number to a random value on startup. The Sequence Number MUST be incremented for the transmission of each new unique frame; whether it is a singlecast or multicast transmission.

A receiving node MUST use the Sequence Number field in the Nonce Get, Nonce Report and Message Encapsulation commands for duplicate detection. Duplicates of recently received commands MUST be discarded.

The following algorithm SHOULD be used for duplicate detection of singlecast messages:

1. If an incoming sequence number and Peer NodeID match an entry in the SPAN table, the frame MUST be discarded.
2. If it is a new sequence number, the received sequence number and Peer NodeID MUST be stored in the SPAN table.

The following algorithm SHOULD be used for duplicate detection of multicast messages:

1. If an incoming sequence number and Peer NodeID match an entry in the MPAN table, the frame MUST be discarded.
2. If it is a new sequence number, the received sequence number and Peer NodeID MUST be stored in the MPAN table.
3.1.6 Key Management

S2 communication relies on two separate derived keys that MUST be shared between any nodes communicating with each other; the combined Encryption and Authentication Key denoted KeyCCM and the CTR_DRBG Key denoted PersonalizationString. Both keys MUST be derived from one Network Key that is exchanged between the nodes using the CKDF-NetworkKeyExpand function detailed in 3.1.4.5.3.1.

In S2, not all nodes share the same network key. S2 separates devices into different Security Classes, so that if one Security Class is compromised, it does not affect other Security Classes. The joining node MUST advertise the supported Security Classes during the S2 bootstrapping process.

The including node MAY grant membership of one or more Security Classes. The joining node MUST then request the granted keys, one at a time.

The Security Classes are listed in Table 8.

S2 Access Control and S2 Authenticated Security Classes are equivalent from a technical perspective, but do not share the same network key. This is simply to prevent compromising the Access Control key if the Authenticated key is compromised.

Client-side authentication is an alternative authentication mechanism intended for the authenticated security bootstrapping of devices which do not have a DSK. Refer to section 3.1.6.3.

Table 8, S2 Security Class Overview

<table>
<thead>
<tr>
<th>Value</th>
<th>Class Name</th>
<th>Example devices</th>
<th>Controller authentication (refer to 3.1.6.2)</th>
<th>Client-side authentication (refer to 3.1.6.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Display DSK</td>
<td>Input DSK</td>
<td>Display DSK</td>
</tr>
<tr>
<td>2</td>
<td>S2 Access Control</td>
<td>Door locks, Garage doors</td>
<td>14 bytes</td>
<td>12 bytes (Optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>None (QR code)</td>
<td>QR code (16 bytes)</td>
</tr>
<tr>
<td>1</td>
<td>S2 Authenticated</td>
<td>Lighting and Sensors (Managed systems and security systems)</td>
<td>14 bytes</td>
<td>12 bytes (Optional)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>None (QR code)</td>
<td>QR code (16 bytes)</td>
</tr>
<tr>
<td>0</td>
<td>S2 Unauthenticated</td>
<td>Lighting and Sensors (Unmanaged systems)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>S0</td>
<td>Legacy door locks</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

All other values are reserved and MUST NOT be used by a sending node. Reserved values MUST be ignored by a receiving node.

The manufacturer decides which classes are relevant to advertise for the intended use of the product. For instance:
- A light dimmer device may advertise the S2 Unauthenticated Class and the S2 Authenticated Class as the intended classes. If a constrained key fob without display is used to create a small system, the wall controller may grant only the S2 Unauthenticated Class to the light dimmer, which then requests only the S2 Unauthenticated Class key.

- A light bulb may advertise the S2 Unauthenticated Class as its only intended class because it does not provide a DSK. Depending on the configured security level, an authentication capable gateway may accept including S2 Unauthenticated Class devices or it may reject including S2 Unauthenticated Class-only devices.

- A door lock device may advertise the S2 Access Control Class as its only intended class because it requires the highest protection level. A constrained key fob, which can grant only the S2 Unauthenticated Class key, rejects including the door lock device and returns an error indication to the user because it cannot authenticate the door lock.

### 3.1.6.1 Key Exchange

The S2 key exchange order MUST comply with Table 9.

<table>
<thead>
<tr>
<th>Order</th>
<th>Key to be exchanged</th>
<th>Key to use for the Key Exchange</th>
<th>Key to use for the Key Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S2.2: Access Control</td>
<td>ECDH Temporary Key</td>
<td>S2.2</td>
</tr>
<tr>
<td>2</td>
<td>S2.1: Authenticated</td>
<td>ECDH Temporary Key</td>
<td>S2.1</td>
</tr>
<tr>
<td>3</td>
<td>S2.0: Unauthenticated</td>
<td>ECDH Temporary Key</td>
<td>S2.0</td>
</tr>
<tr>
<td>4</td>
<td>S0</td>
<td>ECDH Temporary Key</td>
<td>S0</td>
</tr>
</tbody>
</table>

If the S0 Security Class was granted to the joining node, the joining node MUST request the S0 Security Class key after the S2 Security Class keys.

A number of Security Class keys may be granted to the joining node. A temporary key and SPAN MUST be used for the exchange of the Security Class keys. The temporary SPAN MUST be initialized with the Shared Nonce that is established after the S2 temporary key is established. The SPAN value MUST be updated after each Security Class key exchange.

Verification of an assigned key (including S0) MUST always use the newly exchanged key to encrypt the S2 verification message.

If a joining node is unsuccessful requesting all the Security Class keys that it was granted, the node MUST abort the S2 bootstrapping entirely.

### 3.1.6.2 Device Specific Key and User Verification

A Device Specific Key (DSK) is used to protect against man-in-the-middle attacks (MITM) where a malicious attacker tries to intercept and manipulate the key exchange.

The DSK is defined as the first 16 bytes of the Public Key of a node. An S2 node MUST respect the DSK requirements listed in Table 10.
Table 10, DSK requirements

<table>
<thead>
<tr>
<th>Role type</th>
<th>Highest supported Security Class</th>
<th>DSK Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>S2 Access Control</td>
<td>The node MUST make its DSK visible via its UI at any time when Learn mode is enabled. The node MUST create a new ECDH key pair after every S2 bootstrapping attempt (regardless whether it was successful or not).</td>
</tr>
<tr>
<td></td>
<td>S2 Authenticated</td>
<td>The node SHOULD create a new ECDH key pair every after every S2 bootstrapping attempt (regardless whether it was successful or not).</td>
</tr>
<tr>
<td></td>
<td>S2 Unauthenticated</td>
<td></td>
</tr>
<tr>
<td>Slave</td>
<td>S2 Access Control</td>
<td>A unique DSK MUST be assigned to each individual node. The node MUST have a printed label with its DSK. The node SHOULD have a printed QR code representing its DSK.</td>
</tr>
<tr>
<td></td>
<td>S2 Authenticated</td>
<td>A unique DSK MUST be assigned to each individual node. The node MAY create a new ECDH key pair every after every S2 bootstrapping attempt (regardless whether it was successful or not).</td>
</tr>
<tr>
<td></td>
<td>S2 Unauthenticated</td>
<td></td>
</tr>
</tbody>
</table>

A slave node supporting the S2 Authenticated Class or the S2 Access Control Class MUST have the DSK printed on it in text formatted with the following syntax:

```
  zws2dsk:00000-00000-00000-00000-00000-00000-00000-00000
```

It consists of 8 groups of 5 decimal digits, each group representing 16 bits. The digits blocks are prefixed by ‘zws2dsk:’. An example is given in Figure 10.

```
  ECDH Public Key[1..16]:  0x8E,0xEC,0x5C,0x75,0x51,0xCA,0xB5,0x0A,0x83,0x81,0x07,0xD,0xF,0x5,0x38,0xD
                          0x8E,0x5C,0x75,0x51,0xCA,0xB5,0x0A,0x83,0x81,0x07,0xD,0xF,0x5,0x38,0xD

  DSK text: zws2dsk:34028-23669-20938-46346-33746-07431-56821-14553
```

Figure 10, DSK text format (example)

The first five digits MUST be highlighted or underlined to help the user identify the PIN code portion of the DSK text.
The DSK may be additionally represented with a QR Code as shown in Figure 11.

![QR Code](image)

**Figure 11, DSK label and QR code (example)**

A joining node requesting to join the S2 Access Control Class or the S2 Authenticated Class will obfuscate its Public Key by setting the bytes 1..2 to zeros (0x00) before transferring its key via RF.

A joining node requesting to join only the S2 Unauthenticated Class will send the its full Public Key when transferring the key via RF as the including node has no access to the DSK.

The DSK may be used for out-of-band (OOB) authentication in two ways.

- The including controller may use QR code scanning to read the entire DSK off the joining device and match it with the obfuscated public key received via RF from the joining device.
- Else the including controller will ask the user to enter a 5 digits PIN code (the 5 first digits of the DSK label) in order to substitute the obfuscated bytes of the joining node’s Public Key. The including controller may additionally ask the user to visually validate that the rest of the DSK with the Public Key received via RF.

An including controller with support for the S2 Authenticated Class or the S2 Access Control Class SHOULD provide a QR scanning capability for user friendly inclusion. If scanning capability is not available, the including controller MUST provide an interface that allows the user to enter a 5-digit PIN code in case visual authentication is used. The requested PIN code MUST be the first 5 decimal digits of the above DSK text format.

### 3.1.6.3 Client-Side Authentication

When upgrading existing devices to support Security 2 through an over-the-air (OTA) firmware update, there is no DSK printed on the node and the upgraded node MUST therefore generate its public key and DSK internally with the updated firmware.

A device MAY request the use of Client-Side authentication (CSA) if it does not possess a DSK label. If the joining node requests CSA, the including controller MUST ask the user if this should be permitted, and if permitted, the including controller MUST display its own DSK.

As stated in Table 8, the first 10 digits MUST be input on the Joining Node, meaning it MUST have a method of input, like a keypad.

Compared to controller-side authentication where the entire DSK MUST also be visually verified, CSA does come with a higher risk of having the bootstrapping attempt manipulated by an attacker. The attacker, however, has to guess 4 random bytes in one attempt.
3.1.6.4 Initial Key Exchange

Key Exchange MUST always be initiated physically on the device that is being included into the network. Physical activation includes button press, applying power by inserting batteries, mains etc. Initial Key Exchange MUST be carried out using the ECDH temporary key with the including controller.

The Including Node A MUST create a new ECDH Key Pair for each new bootstrapping process, if it supports the S2 Access Control and/or S2 Authentication Security Classes.

The Key Exchange process MUST follow the frame flow illustrated in Figure 12.
Figure 12, S2 bootstrapping frame flow
The key exchange MUST comply with the following steps:

1. **Network inclusion completed:**
   Immediately following a successful network inclusion or after receiving an Inclusion Controller Initiate Command (refer to [16]), the Security 2 enabled controller A MUST start the S2 bootstrapping.

2. **A->B : KEX Get :**
   Including Node A, requests KEX Report from Joining Node B.

3. **B->A : KEX Report :**
   Sent as response to the KEX Get command, this command contains:
   a. Scheme Report – Schemes supported by Node B
   b. Curve Report – Elliptic Curves supported by Node B
   c. Requested Key List – List of Keys (such as S2 Class keys and Security 0 Network Key) which is requested by Node B
   d. Request for Client-Side authentication

4. **A1:**
   Node A MUST verify the KEX Report and, if required, cancel the S2 bootstrapping as described in Section 3.1.6.4.1.
   a. **Optional:** Node A MAY present a dialog allowing the installer to select which specific keys will be granted to Node B. If presented, the installer MUST either confirm a list of granted keys or cancel the security bootstrapping.
   b. If Client-Side authentication is requested, Node A MUST present a dialog asking if Client-Side authentication should be allowed.
      i. If Client-Side authentication is used, Node A MUST subsequently present a dialog with its own DSK.
      ii. Node A MAY reject Client-Side authentication. In this case, Node A MUST either abort the S2 bootstrapping with a KEX_FAIL_CANCEL or only grant a subset of keys that does not require CSA, e.g. Security 0 and Unauthenticated.

5. **A->B : KEX Set :**
   The KEX Set Command contains parameters selected by Node A. The list of class keys MAY be reduced to a subset of the list that was requested in the previous KEX Report from Node B.
   a. Scheme Set – The selected Scheme by Node A for bootstrapping
   b. Curve Set – The selected Curve by Node A for bootstrapping
   c. Key List Set – The list of granted keys to Node B by Node A
   d. Client-Side authentication – Indicating whether Client-side authentication will be used

6. **B1:**
   Node B MUST verify the KEX Set command and, if required, cancel security bootstrapping as described in Section 3.1.6.4.1.
   a. **Optional:** Node B MAY present the Node A’s DSK for verification or input. If presented, installer MUST either confirm or cancel security bootstrapping. This step MAY also be carried out in step B2 instead is CSA is used.
   b. Node B MUST accept what it has been granted, even if it is only a subset of what was requested.
   c. If Node A wrongfully grants CSA without being requested, Node B MUST cancel security bootstrapping.

7. **B->A : Public Key B :**
   Public Key B is the Elliptic Curve Public Key of Node B and is used for the ECDH Key Exchange.

8. **A2:**
   If authentication is required, Node A MUST request that the user enters the PIN code or scan the QR code from Node B in order to verify the DSK (refer to 3.1.6.2 and 3.1.6.4.1).
   a. If Node A was input a PIN code, it MUST substitute the bytes 1 and 2 of the Node B public key with the 2 bytes received in the PIN code. The user MUST be prompted a dialog to visually validate the bytes 3..16 of Node B’s DSK.
   b. If Node A has received the 16 bytes DSK of Node B via QR scanning, it MUST substitute the first 16 bytes of Node B’s Public Key with the 16 bytes received via QR code.
   c. Node A SHOULD continue verifying the DSK input until A3 to allow ECDH calculations to take place while the user is verifying the DSK.
9. **A-B : Public Key A**:
   Public Key A is the Elliptic Curve Public Key of Node A and will be used for the temporary ECDH Key Exchange.
   a. **Mandatory**: If Client-Side authentication is used, the DSK bytes 1..4 MUST be obfuscated by zeros.

10. **B2**:
    Node B MAY optionally present the DSK of Node A to the user for visual verification.
    a. **Mandatory**: If Client-Side authentication is used, the DSK MUST be input on Node B.
       i. If Node B was input a PIN code, it MUST substitute the 4 first bytes of Node A’s Public Key with the 4 bytes received in the PIN code. The user MAY be prompted a dialog to visually validate the bytes 5..16 of Node A’s DSK.
       ii. If Node B has received the 16 bytes DSK of Node A via QR scanning, it MUST substitute the first 16 bytes of Node A’s Public Key with the 16 bytes received via QR code.

11. **Elliptic Curve Shared Secret Established**:
    If B2 is passed, Node A and Node B have performed an ECDH Key Exchange, resulting in an Elliptic Curve Shared Secret.

12. **Temporary Symmetric Key Established**:
    Both Node A and Node B derive a Temporary Symmetric Key from the ECDH Shared Secret based on CKDF-\text{TempExpand} (refer to 3.1.4.5.2).

13. **B->A : Nonce Get**:
    Node B requests a Nonce from Node A that will allow Node B to send messages securely using the Temporary Symmetric Key.

14. **A->B : Nonce Report**:
    A’s Nonce

15. **From this point all frames sent between Node A and Node B MUST be encrypted using the ECDH Temporary Symmetric Key (With the exception of Nonce Get / Report for each Security Class which MUST NOT be encrypted and the Network Key Verify Command, which MUST be encrypted with the most recently exchanged key. Refer to Section 3.1.6.1).**

16. **B->A : KEX Set (echo)**:
    The KEX Set command received from Node A in step 5 is confirmed via the temporary secure channel.
    a. The frame MUST be retransmitted by node B every 10 seconds for 240 seconds in total until either event is detected:
       i. Step 17, KEX Report(Echo) is received
       ii. KEX Fail is received
    b. If neither events are detected after 240 seconds, Node B times out and aborts S2 bootstrapping silently

17. **A3**:
    Node A MUST abort S2 bootstrapping if the KEX Set(Echo) received in step 16is not identical to KEX Set previously sent by Node A in step 5. Refer to Section 3.1.6.4.1.

18. **A->B : KEX Report (echo)**:
    The KEX Report Command received from Node B in step 3 is confirmed via the temporary secure channel.

19. **B3**:
    Node B MUST abort S2 bootstrapping if the KEX Report(Echo) received in step 18 is not identical to KEX Report previously sent by Node B in step 3. Refer to Section 3.1.6.4.1.
    If a Node B node has been granted zero keys, Node B MUST go to step 30 and indicate to Node A that it is terminating the bootstrapping process by returning an S2 Transfer End Command. Node A and Node B will consider Node B to be included non-securely at the end of the bootstrapping.

**Authentication has been completed, and network key exchange begins. Steps 20 through 29 MUST be repeated for each network key Node A has granted. Key Exchange MUST follow the order described in Section 3.1.6.1.**
20. **B->A : Security 2 Network Key Get:**
   Node B requests a specific Key from Node A.

21. **A4:**
   Node A MUST cancel the S2 bootstrapping if the requested key is not in the Key List granted by Node A or if the requested key does not follow the order indicated in 3.1.6.1. Refer to Section 3.1.6.4.1

22. **A->B : Security 2 Network Key Report:**
   This command returns the requested key.

23. **B4:**
   Node B MUST cancel security bootstrapping if the received key was not requested. Refer to 3.1.6.4.1.

24. **Security 2 Network Key Established:**
   Node A and Node B are now in possession of a shared network key.

25. **B->A : Nonce Get :**
   Node B requests a Nonce from Node A that will allow Node B to send messages securely using the recently exchanged Network Key.

26. **A->B : Nonce Report :**
   A’s Nonce

27. **B->A : Security 2 Network Key Verify :**
   This command MUST be sent encrypted by Node B with the newly received Network Key and the recently established SPAN for this key

28. **A5:**
   Node A MUST verify that it can successfully decrypt the Key Verify command using the newly exchanged key.

29. **A->B : Security 2 Transfer End :**
   If Node A is able to decrypt and verify the Key Verify command, it MUST respond with Security 2 Transfer End command with the field “Key verified” set to ‘1’.
   a. If there are more granted keys to request, Node B continues to step 20 and requests the next granted key
   b. If it was the last granted key, Node B continues to step 30.

   All Keys have been requested.

30. **B->A : Security 2 Transfer End :**
   When Node B has no more keys to request it MUST finish the secure setup by sending a Security 2 Transfer End command with the field “Key Request Complete” set to ‘1’.
   a. If this frame is received before all keys have been requested, the controller MUST consider the S2 Bootstrapping process failed.

   All timeouts described in Section 3.1.6.4.2 MUST be implemented. If a node times out, it MUST silently abort the S2 bootstrapping.
3.1.6.4.1 Security 2 bootstrapping Interrupt Points

The including node and the joining node MUST interrupt the Security 2 bootstrapping process if any of the events outlined in Table 11 occur.

### Table 11, Security 2 bootstrapping Interrupt Points

<table>
<thead>
<tr>
<th>Name</th>
<th>Interrupt Reason</th>
<th>KEX Fail Command Encryption</th>
<th>KEX Fail Type (see description in Table 16)</th>
</tr>
</thead>
</table>
| A1   | Key List, KEX Scheme or Curves are invalid or unsupported. | None | KEX_FAIL_KEY  
|      |                  |                              | KEX_FAIL_SCHEME  
|      |                  |                              | KEX_FAIL_CURVES  
|      |                  |                              | KEX_FAIL_CANCEL |
| A2   | User rejects the requested keys. User input an incorrect DSK | None | KEX_FAIL_CANCEL  
|      |                  |                              | KEX_FAIL_DSK   |
| A3   | KEX Set(Echo) is not identical to the previous KEX Set | Temp. Key | KEX_FAIL_AUTH  
|      |                  |                              | KEX_FAIL_DECRYPT |
| A4   | The requested Key was not granted in the original KEX Set | Temp. Key | KEX_FAIL_KEY_GET |
| A5   | The Key Verify Command cannot be decrypted using most the recent key | Temp. Key | KEX_FAIL_KEY_VERIFY |

### Joining Node Interrupts, B1-4

<table>
<thead>
<tr>
<th>Name</th>
<th>Interrupt Reason</th>
<th>KEXFail Command Encryption</th>
<th>KEX Fail Type (see description in Table 16)</th>
</tr>
</thead>
</table>
| B1   | Key List, KEX Scheme or Curves are invalid or unsupported  
|      | Node A wrongfully grants CSA | None | KEX_FAIL_KEY  
|      |                  |                              | KEX_FAIL_SCHEME  
|      |                  |                              | KEX_FAIL_CURVES  
|      |                  |                              | KEX_FAIL_CANCEL |
| B2   | If using CSA, and user either cancels or input an incorrect DSK | - | KEX_FAIL_CANCEL  
|      |                  |                              | KEX_FAIL_DSK   |
| B3   | KEX Report(Echo) is not identical to the previous KEX Report | Temp. Key | KEX_FAIL_AUTH  
|      |                  |                              | KEX_FAIL_DECRYPT |
| B4   | The assigned key was never requested. | Temp. Key | KEX_FAIL_KEY_REPORT |

An aborted Security 2 bootstrapping process due to one of the interrupt points in Table 11 MUST be followed by a Security 2 KEX Fail Command sent to the other party of the S2 bootstrapping. Encryption MUST be used according to Table 11.

If a command is received unencrypted whereas it had to be sent encrypted with the temporary key, the receiving node MUST return an unencrypted KEX Fail Command with the KEX_FAIL_AUTH Fail Type.
If a command is received encrypted with the temporary key whereas it had to be sent encrypted with a network key, the receiving node MUST return a Security 2 KEX Fail Command with the KEX_FAIL_AUTH Fail Type encrypted with the temporary key.

In any case, a node MUST NOT return any error indication if no S2 bootstrapping process is currently ongoing.

### 3.1.6.4.2 Security 2 bootstrapping Timeouts

The including Node or the joining node MUST apply timeouts according to Table 12.

<table>
<thead>
<tr>
<th>Including Node Timers, TA1-7</th>
<th>Description</th>
<th>Timeout (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA1 KEX Report MUST be received after sending KEX Get</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TA2 Public Key Report MUST be received after sending KEX Set</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TA3 S2 Network Key Get MUST be received after sending KEX Report(Echo)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TA4 S2 Network Key Verify MUST be received after sending S2 Network Key Report</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TA5 S2 Transfer End OR S2 Network Key Get MUST be received after sending S2 Transfer End</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TAI1 User MAY change Key List for Advanced Joining mode</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>TAI2 User MUST have verified / input the DSK</td>
<td>240</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Joining Node Timers, TB1-7</th>
<th>Description</th>
<th>Timeout (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TB1 KEX Get MUST be received after non-secure inclusion</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TB2 KEX Set MUST be received after sending KEX Report</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>TB3 Public Key Report MUST be received after sending Public Key Report</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TB4 S2 Network Key Report MUST be received after sending S2 Network Key Get</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TB5 S2 Transfer End MUST be received after sending S2 Network Key Verify</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>TBI1 If Client-Side Authentication is used, the user MUST have verified / input the DSK</td>
<td>240</td>
<td></td>
</tr>
</tbody>
</table>
3.1.7 Security 2 Key Exchange commands

3.1.7.1 Security 2 KEX Get Command

This command is used by an including node to query the joining node for supported KEX Schemes and ECDH profiles as well as which network keys the joining node intends to request.

This command MUST be ignored if Learn Mode is disabled.

The KEX Report Command MUST be returned in response to this command if Learn Mode is enabled.

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = KEX_GET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.7.2 Security 2 KEX Report Command

This command is used for two purposes during the key exchange:

1) This command is used by a joining node to advertise the network keys which it intends to request from the including node. The including node subsequently grants keys which may be exchanged once a temporary secure channel has been established.

2) After establishment of the temporary secure channel, the including node uses this command to confirm the set of keys that the joining node intends to request.

A receiving node MUST ignore this command if Add Node mode is not enabled.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = KEX_REPORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Request CSA</td>
<td>Echo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supported KEX Schemes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supported ECDH Profiles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requested Keys</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.
Echo (1 bit)

If this flag is set to ‘1’, the fields of this command MUST be a copy of the KEX Report received prior to the establishment of the temporary secure channel. All fields described in this section MUST be included in the echo. For the purpose of echoing only, reserved set bits and the reserved field MUST NOT be ignored or set to zero, but MUST be echoed back as received. If future versions of this Command Class append fields to the KEX Report, those fields MUST be omitted from the echo.

If this flag is set to ‘1’, the command MUST be sent securely via the temporary secure channel, i.e. encrypted using the TempKeyCCM and TempPersonalizationString.

The including node MUST NOT set this flag to ‘0’ when sending this command and MUST ignore this command if this flag is set to ‘1’ when received.

The including node MUST return a KEX Set Command in response to this command if the “Echo” flag is set to ‘0’ and is performing S2 Bootstrapping.

The joining node MUST NOT set this flag to ‘1’ when sending this command and MUST ignore this command if this flag is set to ‘0’ when received.

Request CSA (1 bit)

If this flag is set, the joining node is requesting the use of Client-side Authentication (CSA).

Supported KEX Schemes (1 byte)

This field is used to advertise the KEX Schemes supported by the node.

The field MUST be treated as a bitmask and MUST comply with the format indicated in Table 13:

<table>
<thead>
<tr>
<th>Bit</th>
<th>KEX Scheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 2..7</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>KEX Scheme 1</td>
<td>Indicates if Scheme 1 is supported (Security 2 Class Keys 0..2)</td>
</tr>
</tbody>
</table>

All other bits are reserved and MUST be set to zero by a sending node. Reserved bits MUST be ignored by a receiving node.

If the KEX scheme is supported the corresponding bit MUST be set to ‘1’.

If the KEX scheme is not supported the corresponding bit MUST be set to ‘0’.

A node supported the Security 2 Command Class MUST support KEX Scheme 1.
Supported ECDH Profiles (1 byte)

This field is used to advertise the supported ECDH Profiles by the joining node.

The field MUST be treated as a bitmask and MUST comply with the format indicated in Table 14:

Table 14, Supported ECDH Profiles

<table>
<thead>
<tr>
<th>Bit</th>
<th>ECDH Profile</th>
<th>Description</th>
<th>Public Key length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Curve25519</td>
<td>Indicates support for Curve25519 [14]</td>
<td>32 Bytes</td>
</tr>
</tbody>
</table>

All other bits are reserved and MUST be set to zero by a sending node. Reserved bits MUST be ignored by a receiving node.

If the ECDH Profile is supported the corresponding bit MUST be set to ‘1’
If the ECDH Profile is not supported the corresponding bit MUST be set to ‘0’

Curve25519 MUST be supported by a node supporting the Security 2 Command Class.

Requested Keys (1 byte)

This field is used by a joining node to advertise the keys that the manufacturer finds most appropriate for the actual type of product.

The joining node MAY request a subset of the keys defined in Table 15.

The field MUST be treated as a bitmask and MUST comply with the format indicated in Table 15:

Table 15, Requested Keys

<table>
<thead>
<tr>
<th>Security Level (1 highest - 4 lowest)</th>
<th>Bit</th>
<th>KEX Scheme</th>
<th>Key</th>
<th>Indicates support for</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>KEX Scheme 1</td>
<td>S2.2</td>
<td>S2 Access Control Class</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>KEX Scheme 1</td>
<td>S2.1</td>
<td>S2 Authenticated Class</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>KEX Scheme 1</td>
<td>S2.0</td>
<td>S2 Unauthenticated Class</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>KEX Scheme 1</td>
<td>S0</td>
<td>S0 Secure legacy devices</td>
</tr>
</tbody>
</table>

All other bits are reserved and MUST be set to zero by a sending node. Reserved bits MUST be ignored by a receiving node.

If the Key is requested the corresponding bit MUST be set to ‘1’.
If the Key is not requested the corresponding bit MUST be set to ‘0’.

A node supporting the Security 2 Command Class MUST support at least one Key. A node may support multiple keys.
3.1.7.3  Security 2 KEX Set Command

This command is used for two purposes:

1) During initial key exchange this command is used by an including node to grant network keys to a joining node. The joining node subsequently requests the granted keys once a temporary secure channel has been established. The including node MUST send the command non-securely.

2) After establishment of the temporary secure channel, the joining node issues this command to the including node to securely state its intention to request the keys that were granted previously. The joining node MUST send the command securely via the temporary secure channel, i.e. encapsulated using the TempKeyCCM and TempPersonalizationString.

This command MUST be ignored if Learn mode and Add Node mode are both disabled.

The including node MUST return the Security 2 KEX Report Command in response to this command unless it is to be ignored.

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = KEX_SET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Request CSA</td>
<td>Echo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selected KEX Scheme</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Selected ECDH Profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Granted Keys</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Echo (1 bit)

If this flag is set to ‘1’, the fields of this command MUST be an identical copy of the KEX Set Command received prior to the establishment of the temporary secure channel.

The joining node MUST set this flag to ‘1’.

The including node MUST abort the S2 bootstrapping if it receives a KEX Set Command with this flag set to ‘0’.

If the “Echo” flag is set to ‘1’ and the Add Node mode is enabled, the including node MUST return a KEX Report Command with the “Echo” flag set to ‘1’ in response to this command.

The including node MUST set this flag to ‘0’.

The joining node MUST abort the S2 bootstrapping if it receives a KEX Set Command with this flag set to ‘1’.

---

Sigma Designs Inc.  Command Class Definition  Page 51 of 60
Request CSA (1 bit)

If this flag is set to ‘1’, the including node is permitting the use of Client-side Authentication (CSA).

Selected KEX Scheme (1 byte)

This field is used to specify the Scheme that the including node is granting to the joining node. Exactly one bit MUST be set to ‘1’. Reserved bits MUST be examined for the purpose of verifying that exactly one bit is set. Several bits set MUST trigger an error as indicated in 3.1.6.4.1.

For field format, refer to Table 13.

Selected ECDH Profile (1 byte)

This field specifies the ECDH Profile selected by the including node for ECDH Key Exchange. The field MUST carry exactly one of the bits listed in Table 14. Reserved bits MUST be examined for the purpose of verifying that exactly one bit is set.

For field format, refer to Table 14.

Granted Keys (1 byte)

If the Echo field is set to ‘0’, this field MUST specify the keys which the including node is granting to the joining node.

The value of this field MUST be the same set or a subset of the Requested Keys field advertised in the KEX Report by the joining node during initial S2 bootstrapping.

If the Echo field is set to ‘1’, this field MUST advertise the keys which the including node granted to the joining node in the previous KEX Set Command sent by the including node to the joining node.

For field format, refer to Table 15.

A joining node MUST NOT issue Network Key Get requests for keys that are not specified in this field. An including node MUST return a Security 2 KEX Fail Command and abort S2 bootstrapping if it subsequently receives a Network Key Get request for keys that are not specified in this field.

3.1.7.4 Security 2 KEX Fail Command

This command is used to advertise an error condition to the other party of an S2 bootstrapping process.

The interrupt points that may trigger the transmission of this command are defined in section 3.1.6.3.

This command MUST be ignored if Learn mode and Add Node mode are both disabled.

KEX Fail Type (1 byte)

This field MUST advertise one of the types defined in Table 16.
Table 16, Security 2 KEX Fail::KEX Fail Type

<table>
<thead>
<tr>
<th>Value</th>
<th>KEX Fail Type Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>KEX_FAIL_KEX_KEY</td>
<td>Key failure indicating that no match exists between requested/granted keys in the network.</td>
</tr>
<tr>
<td>0x02</td>
<td>KEX_FAIL_KEX_SCHEME</td>
<td>Scheme failure indicating that no scheme is supported by controller or joining node specified an invalid scheme.</td>
</tr>
<tr>
<td>0x03</td>
<td>KEX_FAIL_KEX_CURVES</td>
<td>Curve failure indicating that no curve is supported by controller or joining node specified an invalid curve.</td>
</tr>
<tr>
<td>0x05</td>
<td>KEX_FAIL_DECRYPT</td>
<td>Node failed to decrypt received frame.</td>
</tr>
<tr>
<td>0x06</td>
<td>KEX_FAIL_CANCEL</td>
<td>User has cancelled the S2 bootstrapping.</td>
</tr>
<tr>
<td>0x07</td>
<td>KEX_FAIL_AUTH</td>
<td>The Echo KEX Set/Report frame did not match the earlier exchanged frame.</td>
</tr>
<tr>
<td>0x08</td>
<td>KEX_FAIL_KEY_GET</td>
<td>The joining node has requested a key, which was not granted by the including node at an earlier stage.</td>
</tr>
<tr>
<td>0x09</td>
<td>KEX_FAIL_KEY_VERIFY</td>
<td>Including node failed to decrypt and hence verify the received frame encrypted with exchanged key.</td>
</tr>
<tr>
<td>0x0A</td>
<td>KEX_FAIL_KEY_REPORT</td>
<td>The including node has transmitted a frame containing a different key than what is currently being exchanged.</td>
</tr>
<tr>
<td>0x0B</td>
<td>KEX_FAIL_DSK</td>
<td>The DSK input by the user is incorrect or the DSK user visual validation indicated a mismatch</td>
</tr>
</tbody>
</table>

3.1.7.5 Security 2 Public Key Report Command

This command is used by both the including and the joining node to establish the Elliptic Curve Shared Secret. This is needed to establish the temporary secure channel that enables transfer of all other keys.

This command MUST be ignored if Learn mode and Add Node mode are both disabled.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = PUBLIC_KEY_REPORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Including Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDH Public Key 1 (MSB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECDH Public Key N (LSB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.
Including Node (1 bit)

The Including Node flag advertises if the sending node is the including node or the joining node.

When sent by the including node this flag MUST be set to ‘1’.
The joining node MUST abort S2 bootstrapping if this flag is set to ‘0’ in a received command.

When sent by the joining node this flag MUST be set to ‘0’.
The including node MUST abort S2 bootstrapping if this flag is set to ‘1’ in a received command.

ECDH Public Key (N bytes)

Device Specific ECDH Public Key of the sending node.

The public key MUST be unique for each node. The length of this field is determined by the chosen ECDH profile. Refer to Table 14.

If Controller-Side Authentication is used, the DSK bytes 1..2 MUST be obfuscated with zeros (0x00) by the joining node
If Client-Side Authentication (CSA) is used, the DSK bytes 1..4 MUST be obfuscated with zeros (0x00) by the including node.

If no authentication is used (S2 Unauthenticated and/or S0 classes are granted only), both joining and including nodes ECDH Public Keys MUST be transmitted in their integrality

3.1.7.6 Security 2 Network Key Get Command

This command is used by a joining node to request one key from the including node. One instance of this command MUST be sent for each key that was granted by the including node.

The command MUST be sent security encapsulated using the TempKeyCCM and TempPersonalizationString.

This command MUST be ignored unless the Add Node mode is enabled.

The Security 2 Network Key Report Command MUST be returned in response to this command unless this command is ignored.

This command MUST NOT be issued via multicast addressing.

A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2_NETWORK_KEY_GET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requested Key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Requested Key (1 byte)

This field is used to request a network key.
Only one key MUST be requested at a time, i.e. only 1 bit MUST be set to ‘1’. This field MUST be encoded according to Table 15.

### 3.1.7.7 Security 2 Network Key Report Command

This command is used by an including node to transfer one key to the joining node.

The command MUST be sent security encapsulated using the TempKeyCCM and TempPersonalizationString.

This command MUST be ignored unless the Learn mode is enabled.

The joining node MUST store the received network key in non-volatile memory.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2_NETWORK_KEY_REPORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Granted Key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Key byte 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Key byte 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Granted Key (1 byte)**

This field is used to indicate which Network Key is carried in the command and MUST be encoded according to Table 15.

**Network Key (16 bytes)**

This field carries the granted Network Key.

### 3.1.7.8 Security 2 Network Key Verify Command

This command is used by a joining node to verify a newly exchanged key with the including node.

This command MUST be sent security encapsulated using the Key that was just exchanged and MUST be encrypted using the derived KeyCCM and PersonalizationString.

The Security 2 Transfer End Command MUST be returned in response to this command unless it is to be ignored.

The joining node MUST NOT consider the actual key exchange to be complete until a Security 2 Transfer End command has been received from the including node.

This command MUST be ignored if Learn mode and Add Node mode are both disabled.
3.1.7.9 Security 2 Transfer End Command

This command is used by the including node to complete the verification of each individual key exchange while the joining node uses this command to complete the S2 bootstrapping process after all granted keys have been successfully exchanged.

The joining node MUST send this command after all granted keys have been verified.

This command MUST be ignored if Learn mode and Add Node mode are both disabled.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2TRANSFER_END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2TRANSFER_END</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reserved

This field MUST be set to 0 by a sending node and MUST be ignored by a receiving node.

Key Verified (1 bit)

The including node MUST set this flag to ‘1’ if it has successfully verified the Key Verify Command using the newly exchanged network key.

This flag MUST be set to ‘0’ in all other cases.

Key Request Complete (1 bit)

The joining node MUST set this flag to ‘1’ if it has completed exchanging all granted keys.

This flag MUST be set to ‘0’ in all other cases.

The including node MUST consider S2 bootstrapping to be successfully completed if it receives this command with this flag set to ‘1’ and all keys have been exchanged.

3.1.8 Discovery of Security capabilities commands

A controlling node may discover the Command Class capabilities of a securely included node.

The advertised capabilities MAY depend on the actual security level used to request capabilities. Likewise, the advertised capabilities at a given security level may depend on the S2 bootstrapping security level.
3.1.8.1 Security 2 Commands Supported Get Command

This command is used to query the command classes that a joining node supports via secure communication.

Security 2 encryption MUST be used when transmitting this command.

The Security 2 Commands Supported Report Command MUST be returned in response to this command. The response MUST be sent encrypted, using the same Security 2 Class key and encapsulation as was used for this command.

A node receiving this command on its highest supported Security Class MUST respond with the Security 2 Commands Supported Report Command containing all supported secure command classes.

A node receiving this command on any other Security Class than its highest supported Security Class MUST respond with the Security 2 Commands Supported Report Command containing no command classes.

A node receiving the Security Commands Supported Get of the (non S2) Security 0 Command Class, MUST respond with the Security 0 Commands Supported Report Command containing no command classes.

This command MUST NOT be issued via multicast addressing. A receiving node MUST NOT return a response if this command is received via multicast addressing. The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

---

3.1.8.2 Security 2 Commands Supported Report Command

This command is used to advertise the commands that a joining node supports via secure communication. Security 2 encryption MUST be used when transmitting this command.

Transport Service [7] segmentation MUST be used if the command is longer than the available payload length of a single Z-Wave MAC frame.

---

This command MUST NOT advertise command classes that the joining node can control in other nodes. Network management user interfaces SHOULD discover control capabilities of a node via the Association Group Information (AGI) Command Class.
As per section 3.1.6, a node may be assigned up to four keys for the S2 Unauthenticated, S2 Authenticated, S2 Access Control and S0 Classes, respectively.

For instance, a light dimmer may accept non-securely transmitted commands if it is not S2 bootstrapped, while the same dimmer will require secure communication for commands if it is S2 bootstrapped. This allows a Security 2 enabled light dimmer to be compatible with first-generation non-secure Z-Wave networks while it will operate fully secure when included in a S2 network at a later time.

In another example, a door lock may be designed to work only with the S0 network key or the S2 Access Control Class key. Inclusion in a network without security bootstrapping will not allow operation of the lock and if bootstrapped with the S2 Access Control Class, S0 encrypted commands will also be ignored.

Supported command classes advertised by this command MUST comply with Table 17.

<table>
<thead>
<tr>
<th>Category</th>
<th>Advertising in this command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command classes that the node supports via non-secure (lower than the highest bootstrapped security class) communication</td>
<td>MUST be advertised</td>
</tr>
<tr>
<td>Command classes that the node only supports via secure communication</td>
<td>MUST be advertised</td>
</tr>
<tr>
<td>Command classes that the node controls via non-secure communication</td>
<td>MUST NOT be advertised</td>
</tr>
<tr>
<td>Command classes that the node only controls via secure communication</td>
<td>MUST NOT be advertised</td>
</tr>
</tbody>
</table>

The Node Info Frame MUST advertise the Security 2 Command Class before inclusion. The Node Info Frame MUST NOT advertise the Security 2 Command Class after S2 bootstrapped inclusion or failed S2 bootstrapping.

The Security 2 Command Class MUST NOT be advertised in this command. The Transport Service Command Class MUST NOT be advertised in this command.

A sending node MAY terminate the list of supported command classes with the COMMAND_CLASS_MARK command class identifier.

A receiving node MUST stop parsing the list of supported command classes if it detects the COMMAND_CLASS_MARK command class identifier in the Security 2 Commands Supported Report.

**Command Class (N bytes)**

This field advertises the command classes that the node supports.

A normal Command Class identifier MUST be one byte long in the range (0x20 – 0xEE). An extended Command Class identifier MUST be two bytes long where the first byte is in the range (0xF1 – 0xFF), while the second byte is in the range 0x00 – 0xFF.

A joining node MAY advertise extended command classes. An including node MUST accept extended command classes.
3.1.8.3 Security 2 Capabilities Get Command

This command is used to request S2 communication specific capabilities of a node.

The Security 2 Capabilities Report Command MUST be returned in response to this command.

This command MUST NOT be issued via multicast addressing.
A receiving node MUST NOT return a response if this command is received via multicast addressing.
The Z-Wave Multicast frame, the broadcast NodeID and the Multi Channel multi-End Point destination are all considered multicast addressing methods.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = COMMAND_CLASS_SECURITY_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2_CAPABILITIES_GET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.8.4 Security 2 Capabilities Report Command

This command is used to advertise S2 communication specific capabilities of a node.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Class = SECURITY_2_CAPABILITIES_GET</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command = SECURITY_2_CAPABILITIES_REPORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supported SPANs

This field is used to advertise the number of concurrent singlecast sessions this node supports.
The value of this field MUST be in the range 5..255.

Supported MPANs

This field is used to advertise the number of concurrent multicast sessions this node supports.
The value of this field MUST be in the range 5..255.
REFERENCES

[10] NIST, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Special Publication 800-38B.
[16] Sigma Designs, SDS13673, Inclusion Controller Command Class