Z-Wave Report

Contents

Section I
Introduction to Project and devices

1) Project Overview

For later..

2) Hardware Used in the Project

1) Z-wave PC controller
2) Z-wave Zniffer
3) Aeotec Trisensor
4) Dome Door/Window Sensor
5) Z-wave 700 - zen gecko wireless Starter Kit

3) Detailed Description of Devices

1) Aeotec Trisensor

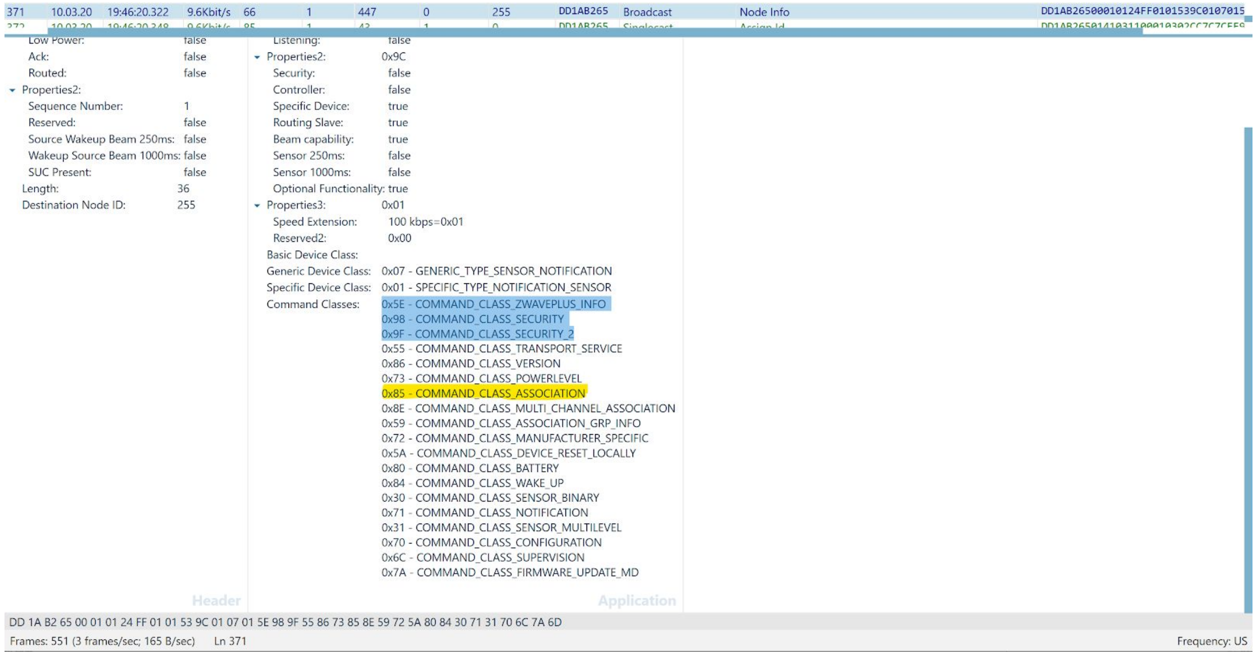
Main Function : **NOTIFICATION SENSOR**

Features:

1) incorporates the following sensor functions:
    a) Motion (upto 9.5 metres / 31 feet)
    b) Temperature (+- 2% accuracy)
    c) Light
2) **Z-wave Security Protocol Supported** : SO, S2
3) **Maximum Wireless range** : 150m / 500ft
4) **Installation locations**:
    a) Ceiling corner
    b) Shelf
    c) Recessed in wall or ceiling

5) **Z-wave certification** : Z-Wave Plus

6)



Above image describes the List of Command Classes supported by this trisensor (This is a **NIF - Node Info Frame** broadcasted by the sensor before the inclusion process occurs - more information on this later)

Main command classes for use in this project:

a) COMMAND_CLASS_Z_WAVEPLUS_INFO
b) COMMAND_CLASS_SECURITY
c) COMMAND_CLASS_SECURITY_2
d) COMMAND_CLASS_ASSOCIATION (used for grouping this sensor with other devices such as light bulbs, cameras etc.**)**

Refer to section of command classes for detailed description for each of them

7) **Encryption supported** = Yes

8) **<u>Real World deployment scenarios</u>:**

a) Detect motion + send notification to LED bulb (cameras recording video in dim lights might have great benefit here)
b) Detect temperature changes + send notifications to heating/cooling systems.
c) <u>Immediate notifications to smartphone when not at home</u>.
   i)Required items are Samsung SmartThings Hub + SmartThings Application
   [Connecting Trisensor to Smartphone]

If a user has added contacts to their Contact Book, SmartApps can prompt a user to select contacts to send notifications to. This allows a user's contacts to be managed independently through the Contact Book, and SmartApps can tap into that feature. This has the advantage that a user does not have to enter in phone numbers for every SmartApp.

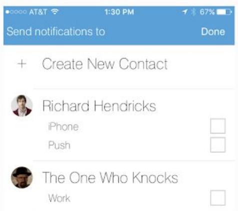
Sending notifications by using the Contact Book feature is the preferred way for sending notifications in a SmartApp.

Selecting Contacts to notify

To allow a user to select from a list of their contacts, use the `"contact"` input type:

```
preferences {
    section("Send Notifications?") {
        input("recipients", "contact", title: "Send notifications to")
    }
}
```

When the user configures this SmartApp, they can then select which contacts they want to notify, and how they should be notified (SMS or push):



The user can have contact book set up in the SmartThings Application so send SMS messages to contacts for notifications.

9) **inclusion instructions**

a) Plug CR123A battery
b) Press add button on Z-wave Gateway (in our case, Z-Wave PC Controller) , then press action button on the sensor
    i) If controller supports S2 security, enter the first 5 digits of Device Specific Key (DSK) of the Trisensor. (more on this later)
    ii) If not, then press Trisensor's action button.

10) **Factory reset instructions**

a) Hold the action button for 15 seconds until the light blinks red.

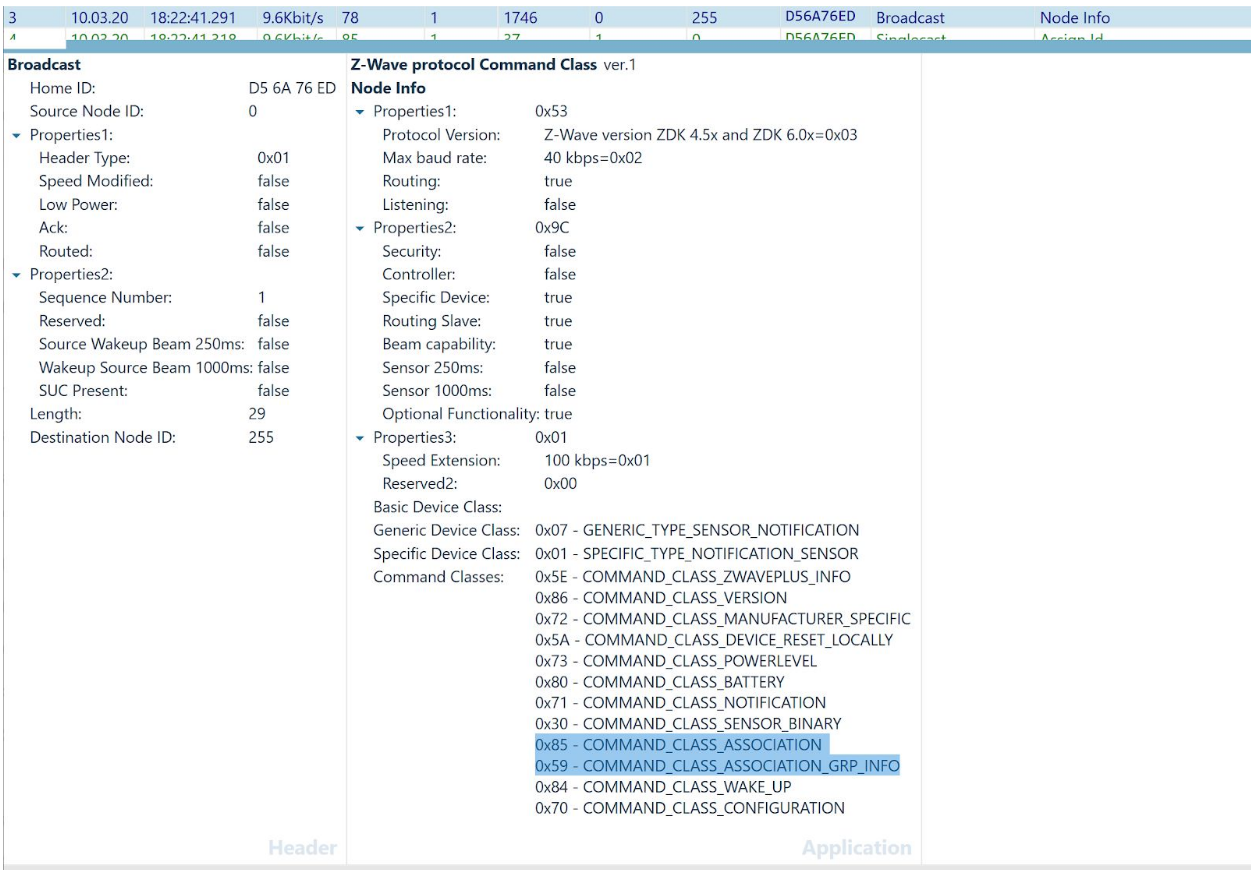
2) Dome Door/Window Sensor

Main Function : **NOTIFICATION SENSOR**
Features:

1) **Capabilities :**
    a) **BINARY SENSOR (ACCESS CONTROL)**
    b) **NOTIFICATION SENSOR (ACCESS CONTROL)**
2) **Range** : upto 150m from the nearest plugged-in device
3) **Installation Locations**:
    a) Door/window
    b) Garage doors
    c) cabinet/drawers
4) **Real World Deployment Scenarios:**
    a) Use with doors, windows and garage doors.
    b) Detect activity with door/window + send notification to Hub + Hub sends notification to alarm
    c) Automate basement lights with this sensor

d) Attach sensor to cabinets etc containing private stuff + receive alert if someone snoops sound.

5) **Z-wave Certification** : Z-wave Plus
6) **Encryption Supported** : No
7) **Z-Wave Security Protocol Supported** : None
8)



Above image describes the command classes supported by this door/window sensor. (This is a **NIF - Node Info Frame** broadcasted by the sensor before the inclusion process occurs - more information on this later)

Main command classes for use in this project:

a) **COMMAND_CLASS_ASSOCIATION (**used for grouping this sensor with other devices such as light bulbs, cameras etc.**)**
b) **COMMAND_CLASS_ASSOCIATION_GRP_INFO**
   Refer to section of command classes for detailed description for each of them

6) **Inclusion instructions**:
   a) Press add button on the Z-wave Gateway(in our case Z-Wave PC Controller)
   b) Press Connect button 3 times in a row (LED will flash 5 times indicating successful inclusion)

7) **Exclusion from the network**:

   a) Press Connect button 3 times in a row(LED will flash 5 times indicating successful exclusion/disconnection from the network)

8) **Factory Reset the sensor**:
   a) Hold the Connect Button for 10 seconds until the LED blinks once, then release the button.

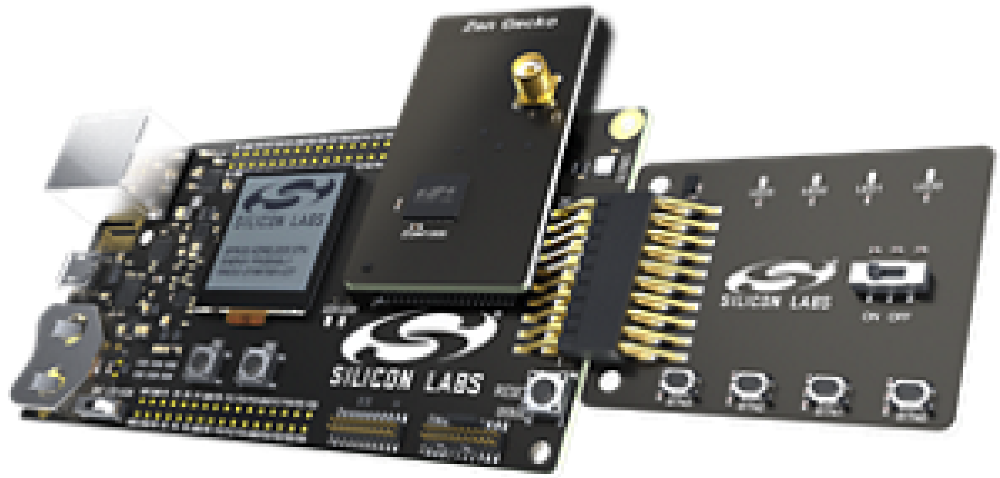
3) Z-wave PC controller

⟵ static/bridge controller for Z-wave

- API for a Static/bridge controller to implement Z-Wave PC application

4) Z-wave Zniffer

5) Z-Wave 700 Development Kit

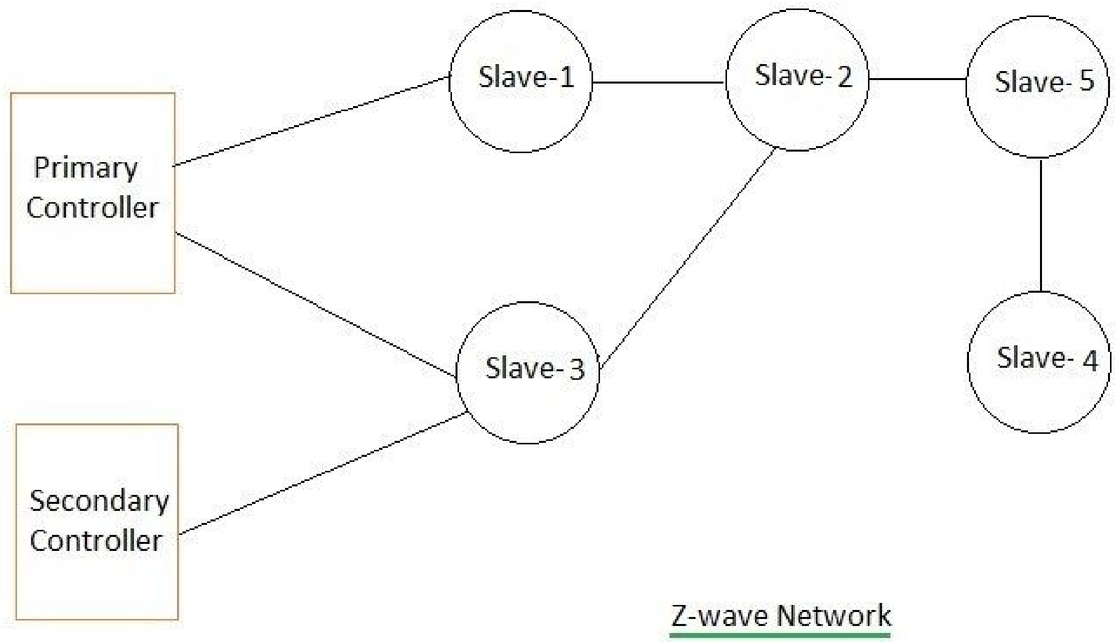


Description Pending...

Section II
Z-wave network concepts

Z-Wave Introduction

- A low power wireless communications protocol
- Uses low frequency RF for communication.
- Primary Use = home automation for wireless control of home appliances
- Uses MESH NETWORKING topology



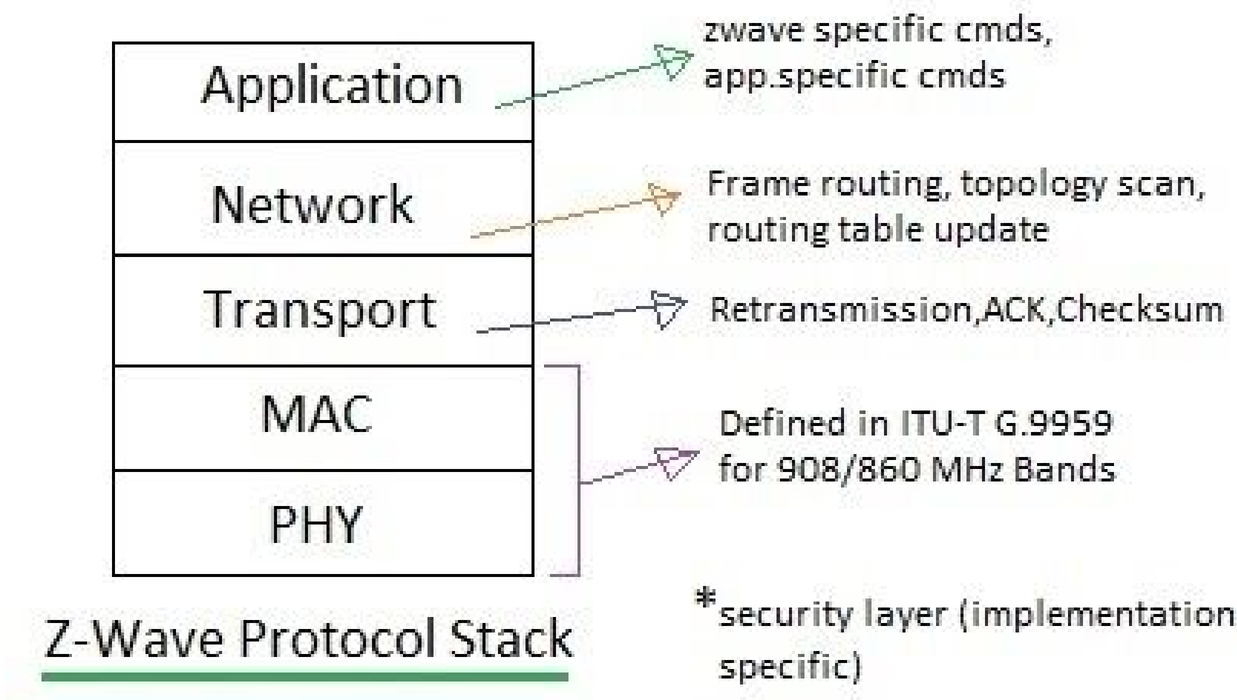
- the image above depicts the basic network topology of a Z-Wave network

- A typical Z-wave network consists of 2 types of devices **:**
  a) Controller devices = nodes in a z-wave network which initiates control commands. They also send out commands to other nodes.
  b) Slave devices = nodes which replies based on the commands received and also execute commands. They also forward commands to other nodes
- Several parts of a Z-wave network with detailed descriptions are as follows:
  a) **Controller** → this device will have a full routing table for this mesh network and it will host it. The controller can communicate with all nodes in the Z-wave network. Controllers are of 2 types :
     - **Primary Controller** → the controller which creates a new Z-wave network initially becomes the primary controller. (There will be only one primary controller in a Z-wave network) . It can include/exclude the nodes in the network. It always keeps the latest topology of the network. It also manages allocation of node

- ❏ **Secondary Controllers** → Controllers added to the network through primary controllers are known as Secondary Controllers. They don't have the capability to include /exclude nodes. They get copies of the routing table from the primary controller.

b) **Slaves** → receive the commands as well as perform actions based on those commands. They cannot transmit information directly to other salve nodes or controllers unless they are instructed to do so through commands. They don't compute routing tables but store them. They can act as a repeater.

Z-Wave Protocol Stack



- **Z-wave protocol Stack** → *low bandwidth + half-duplex protocol to establish reliable wireless communication* . This protocol stack doesn't need to take care of large amounts of data as well as any kind of critical or streaming data.

Physical Layer/ MAC Layer

- ➔ Z-wave devices operate at 908.42 Mhz in USA
- ➔ Z-wave devices operate at 868.42 Mhz in Europe
- ➔ MAC layer works in concert with the Physical Layer
- ➔ MAC data frame contains information from the transport layer
- ➔ **Home ID** = **4-bytes (32-bit**) in length and unique to each Z-wave network. This is generated by the controller randomly during every factory default reset and is not modifiable by the user. **Its uniqueness allows multiple Z-Wave networks to operate within proximity of each other and avoid network crosstalk**.
- ➔ **Node ID** = **1-byte (8 bits) long** and is unique for each device in the network. The controller assigns the node ID to every device during the

inclusion process. Node ID is unique among devices on the logical network.

Transport Layer

- The transport layer information passed down includes frame control information, the destination ID, data length, payload and checksum
- Frame control is 2-bytes in length. Frame control, in combination with the Header type subfield, identifies the frame-type:
- There are 4 frame types : **singlecast, ACK, multicast and broadcast.**
    - **Singlecast** = these frames will be transmitted to one specific node. Transmitter will receive an ACK frame for it. If this or ACK is lost, the singlecast frame will be retransmitted.
    - **ACK** = same as singlecast but absent a payload and is the destination node's acknowledgement of the receipt of the transmission.
    - **Multicast** = they are transmitted to more than one node (max = 232 nodes).No ACK is sent back with it. This frame isn't used for reliability.
    - **Broadcast** = received by all nodes in the network . No ACK is sent back.

- **Length Field** = 1-byte describing length of the entire MPDU(MAC PDU) - this frame is labelled as transport frame.
- **Destination Node ID** = 2-byte Node ID of the device for which transmission is intended.
- **Payload** = defined by frame type. The frame type will also contain data passed down from the application layer.

Network Layer

- Calculates packet routes based on the network routing table.
- Also responsible for topology scans and updating the routing table.
- Consists of 2 frame types:
    - Routed Singlecast frame
    - Routed Acknowledge frame
- These serve the same purpose as the two frame types of the same name in the transport layer.

Application layer

- Consists of instructions intended for the destination node.
- Instructions = Command class, commands and command parameters.
- Total command classes = 74 , as per device’s functionality.
- Command class structure = comparable to OOP structure
- Command classes = OOP object classes
- Commands = OOP methods

Security Layer (Encryption Layer)

- Hardware-based 128-bit AES Encryption (for 400 and 500 series chips)
- Key exchange routine is followed as depicted in the image below:
- For S2 key exchange → a temporary key is exchanged first using the Diffie-Hellman algorithm. That temporary key is then used to encrypt the S2 key. Since, the temporary key is known only to the two nodes participating in the key exchange, the key is unique to those two nodes only.
- Z-wave devices using 400 and newer 500 series chips can use hardware-based 128-bit AES encryption

Application Layer (Device specific commands and parameters , controls payloads in the frames received or to be transmitted , involves decoding and execution of commands in a Z-wave network)
Security Layer (Encryption, Anti-Replay and MAC)
Network Layer (32-bit Home ID, 8-bit Node ID , frame routing, topology scan and routing table updates)
Transport Layer (Error detection, transmission of reception of frames, ACK frame transmission and insertion of checksum)
MAC/ Physical Layer (868.42(EU) / 908.42 (US) MHz, takes care of Home ID and node ID , collision avoidance algorithm and backoff algorithm)

Z-wave Protocol Layers and their functions in a nutshell

Section III
Z-Wave Command Classes

Introduction

Z-Wave command classes are divided into 4 major categories:

1) Application Command Classes
2) Management Command Classes
3) Transport-Encapsulation Command Classes
4) Network-Protocol Command Classes

Command Class Format

PUT COMMAND CLASSES AFTER CLARIFICATION

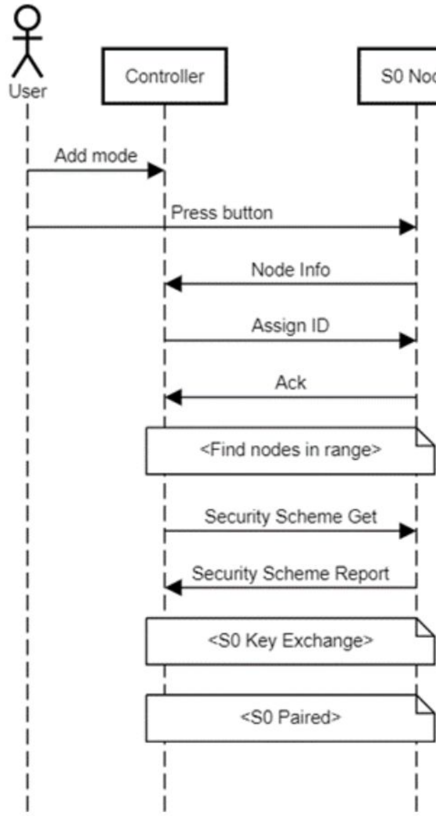
Section IV
Z-Wave S0 Security Framework

Z-wave S0 Security Framework

specifications

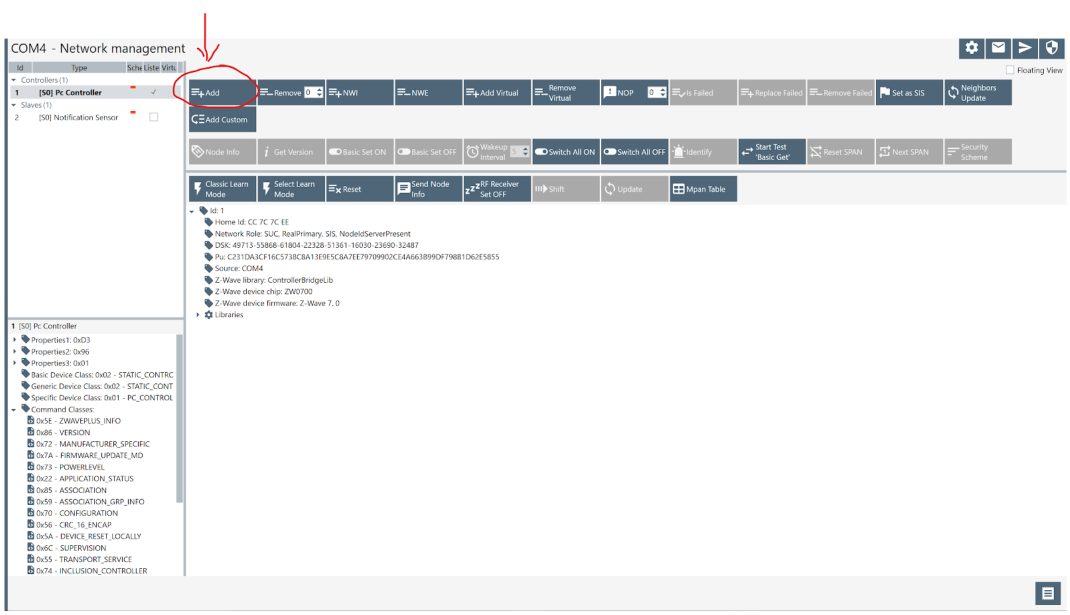
- Involves a single network-wide key .
- Flaw : The network-wide key is encrypted using 16bit 0s temporary key → this makes it vulnerable.
- Uses AES symmetric block cipher with 128-bit key length.
- Allows for both secure as well as non-secure nodes to exist in the same network.
- No security solution on the MAC layer and the routing (network) layer.
- Only supports Singlecast.
- This layer addresses the following areas:
    - Message Integrity (using MAC encryption)
    - Data freshness (using nonces)
    - Confidentiality (using AES 128-bit symmetric block cipher)

S0 Node Inclusion and Key Exchange Processes

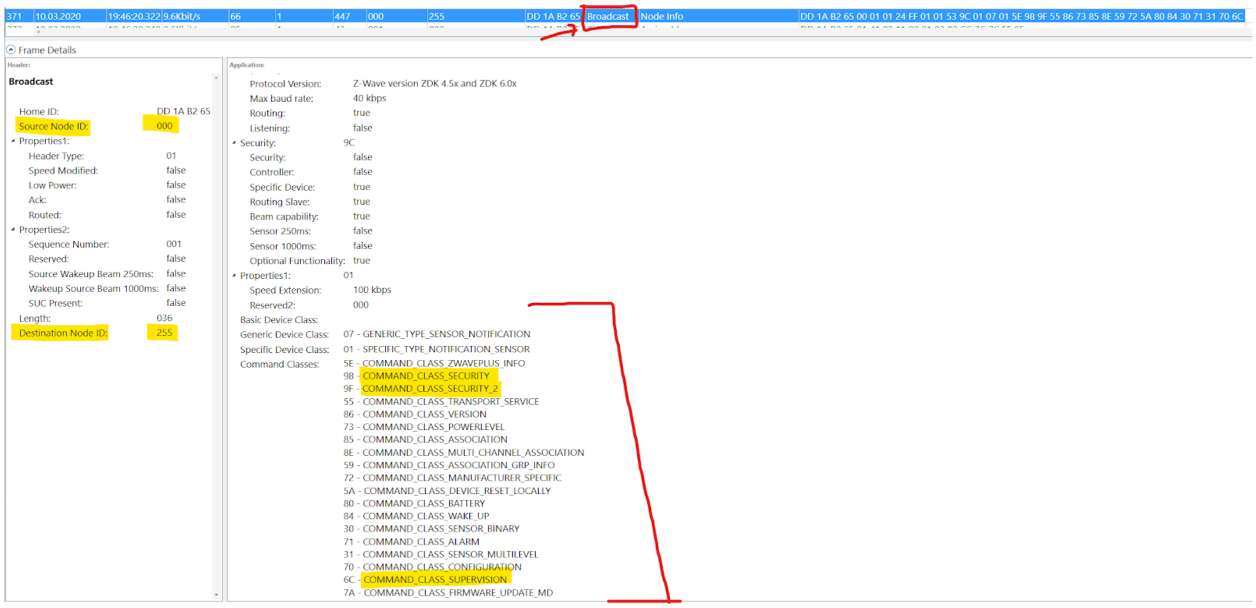


The steps of the node inclusion in the network using S0 are depicted below through captures :

1)



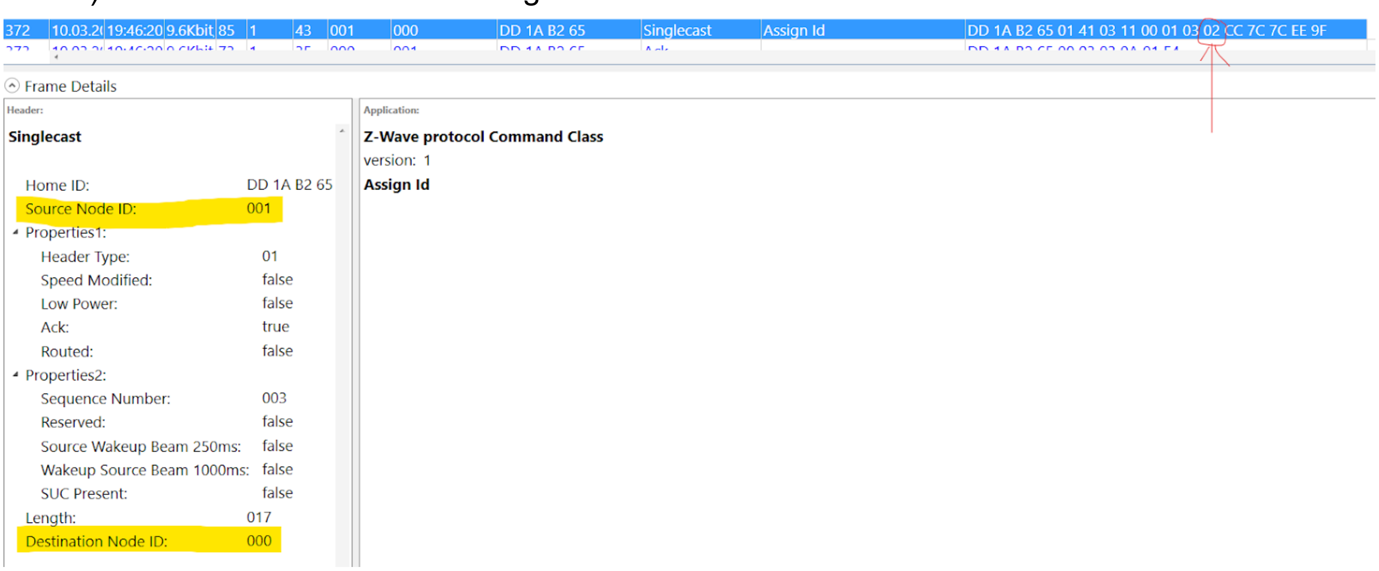
2) Push the add button on the sensor
3) Sensor sends a Node info frame (this frame is of trisensor)



<u>NOTE: </u> there are some important facts about this frame

- The source Node ID is 0. This denotes that this sensor hasn't been assigned yet a Node ID by the controller.
- The destination Node ID is 255. This denotes that the sensor doesn't know the ID of the controller so this is a broadcast. Once senor gets the Node ID, only Singlecast will be used.

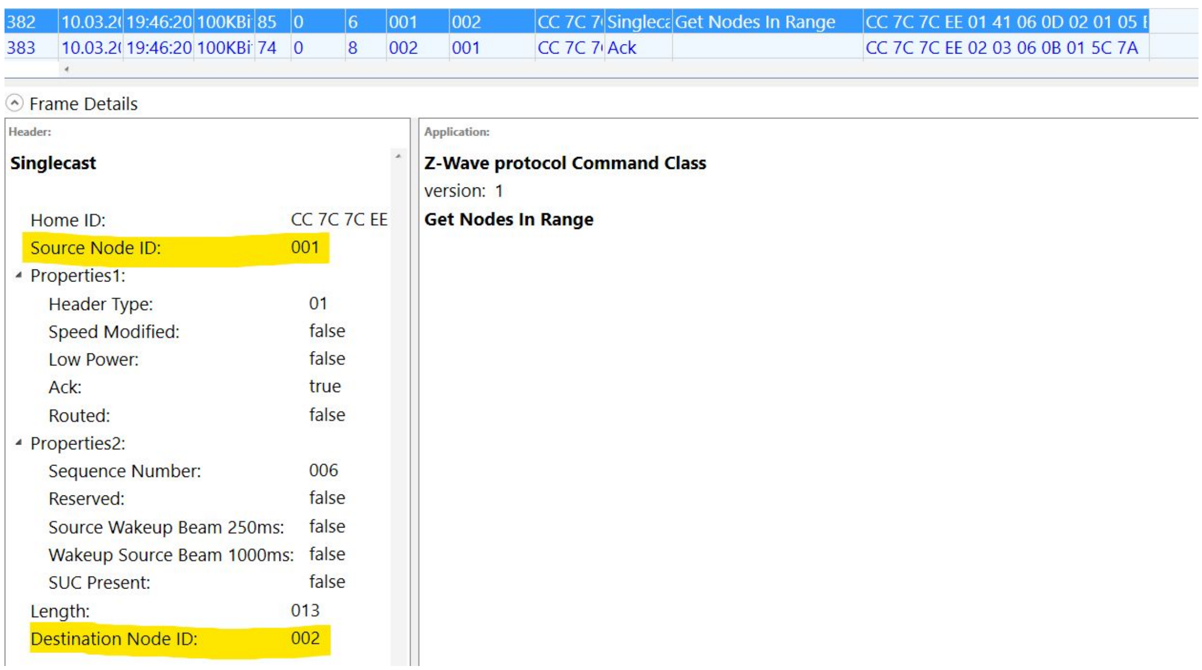
4) The controller sends an Assign Node ID frame to the sensor.



<u>Note:</u> there are some important facts about this frame:

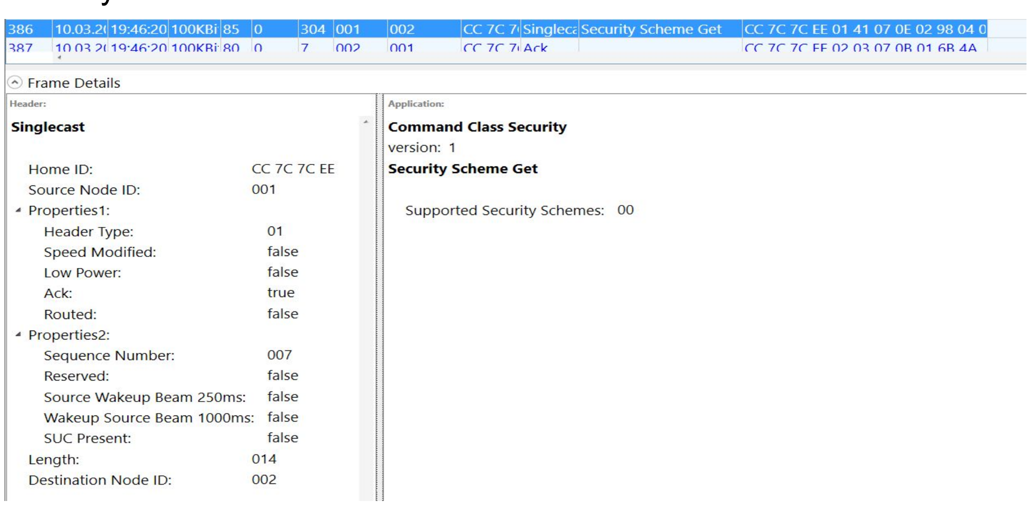
- The source Node ID is 1 ,denoting the controller.
- The destination Node ID is 0,corresponding to the node ID of the sensor the request was actually sent from.
- The payload (red arrow) denotes the value assigned by the controller. In this case, it is value 2. Therefore, the value 2 will replace 0 as the Node ID of the sensor from now onwards.

5) Controller sends Get Nodes in Range to sensor.

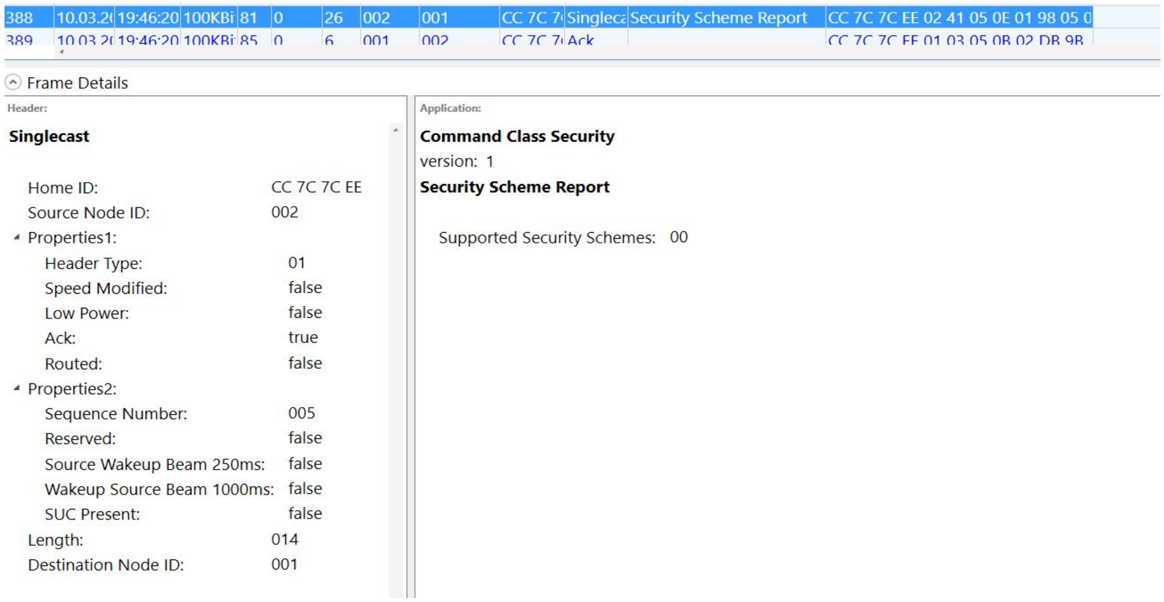


<u>Note:</u> the destination Node ID is now 2

6) Security Scheme Get



7) Security Scheme Report

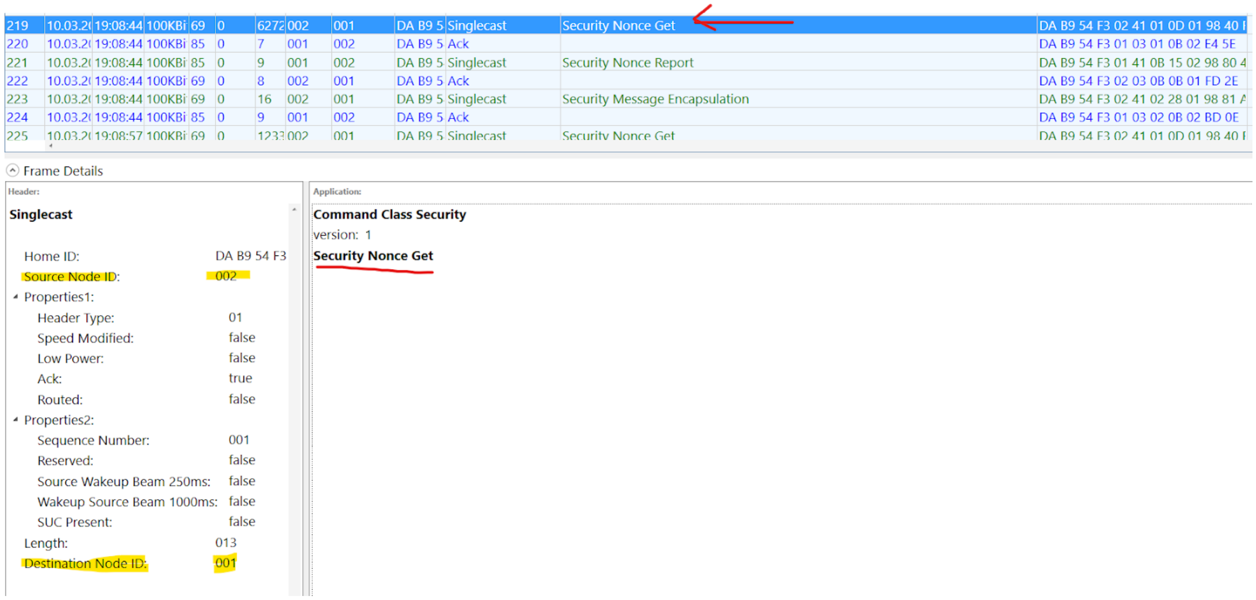


8) S0 Key Exchange

1. Nonce get
2. Nonce report
3. Network key set

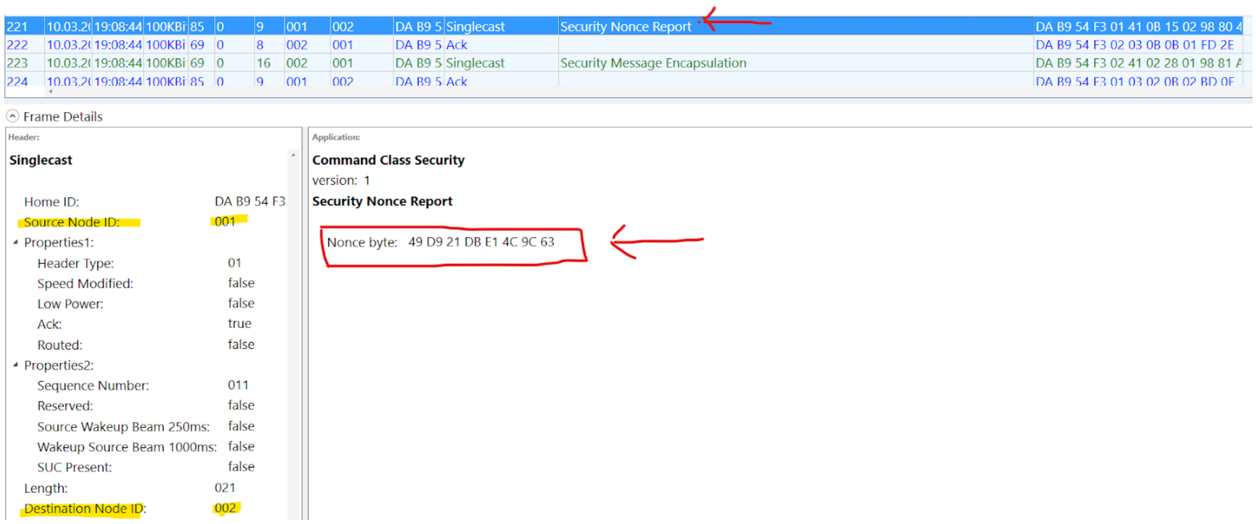
S0 Key Exchange

Step 1)
Sensor sends a Nonce Get frame to Controller



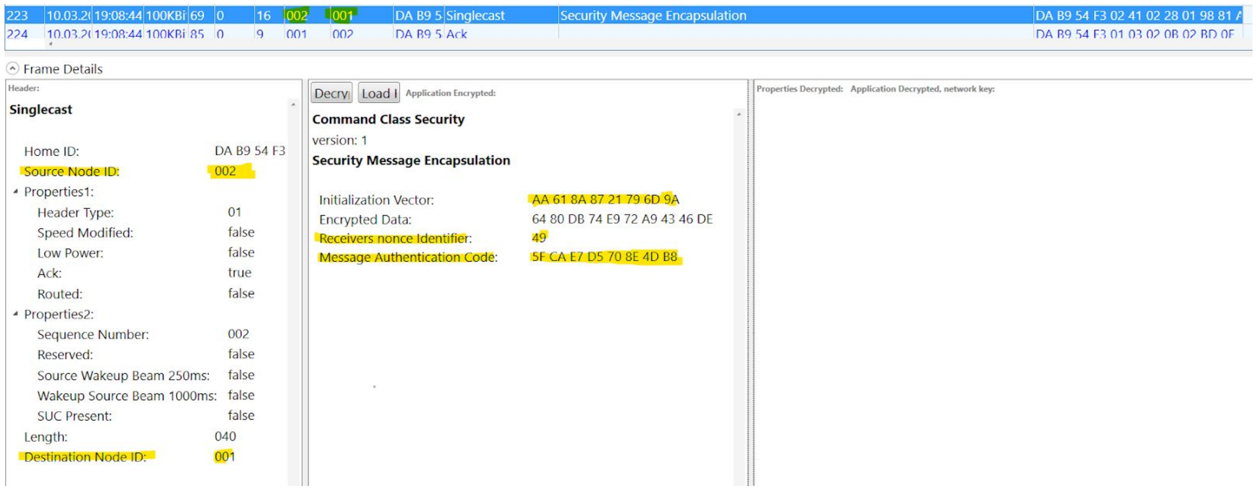
Source = Node ID → 2 = Sensor
Destination = Node ID → 1 = Primary Controller

Step 2)



Controller sends a 8-byte Nonce back to Sensor through Nonce Report frame.
Source = Node ID → 1 = Primary Controller
Destination = Node ID → 2 = Sensor

Step 3)



Sensor encrypts the data, computes the MAC and sends it to the Controller.

Note : Sensor has mentioned Nonce Identifier, which will be used by the Controller to Identify the correct Nonce used by Sensor when computing the IV. Controller checks this value inside his “internal nonce “ table.

If the MAC computed matches the MAC obtained, the controller decrypts it further. If the MAC is different, the controller discards the frame.

S0 Frame Flow

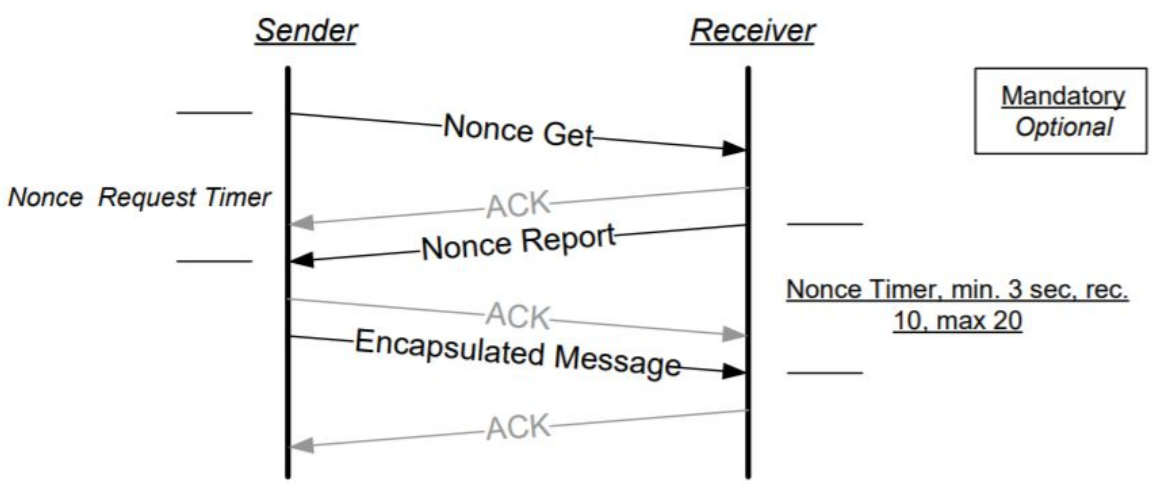


Figure 1, Sending secure messages

The above figure illustrates a 4-step procedure for sending data using S0 protocol. The steps are described below:

NOTE : since the security protocol is S0, all of the associated frames will be Singlecast.

Consider the scenario where A has to send a secure message to B, here are the steps for it:

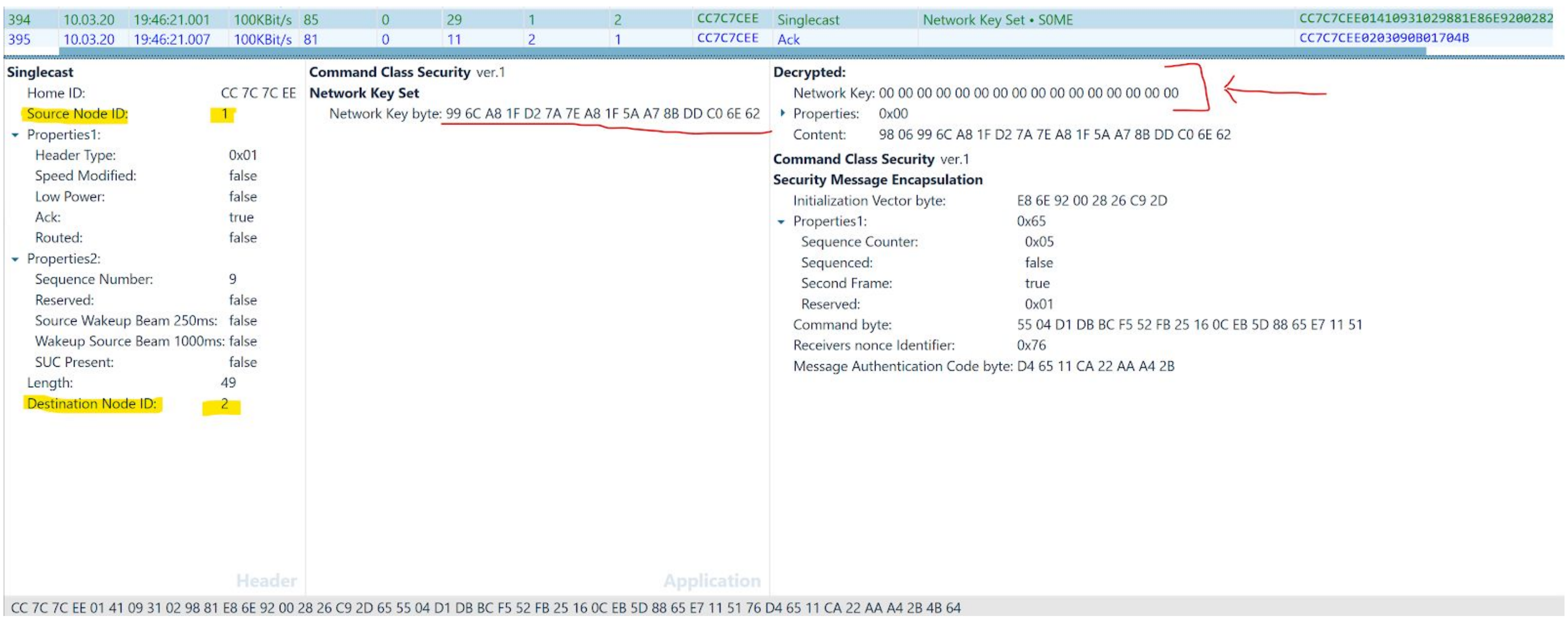
1) A sends a **Nonce get** request to B
2) B uses its PRNG (Pseudo Random Number Generator) to generate a 8-byte nonce and sends it through the **Nonce Report** to A .
3) A generates its own 8-byte nonce , concatenates it with B's Nonce and total forms to 16-byte IV

IV = Sender (A's) Nonce || Receiver (B's) Nonce

   The encryption key Ke (AES key) and Ka (MAC Key) are pre-computed. Therefore, both of them along with IV are used for AES encryption followed by MAC . This frame is called **Security Encapsulation Frame**. A sends it to B.
4) B uses the Nonce identifier (RI) in the received package to identify the nonce of A from its internal nonce table. B computes the MAC . If correct, the frame is discarded. Else, B decrypts the frame and passes it to the application layer.

FLAW IN S0 SECURITY PROTOCOL

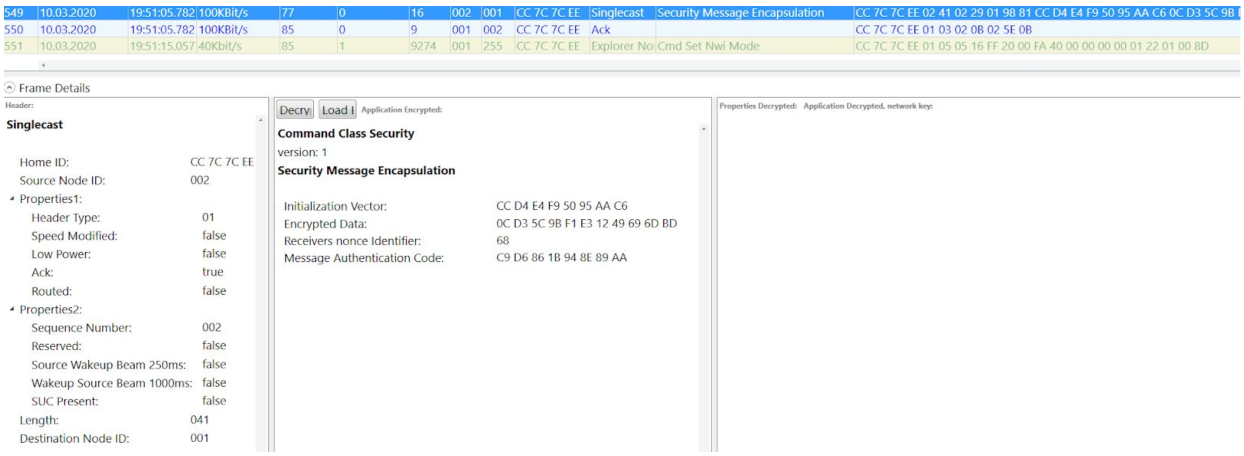
- SO exchanges the shared network key by encrypting it with a fixed 16-byte network key 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 (all 0's).
- An active listener in the RF range can easily grab it and decrypt it to get the original shared network key. Once that's done, the attacker can perform an attack on any device in the network.
-



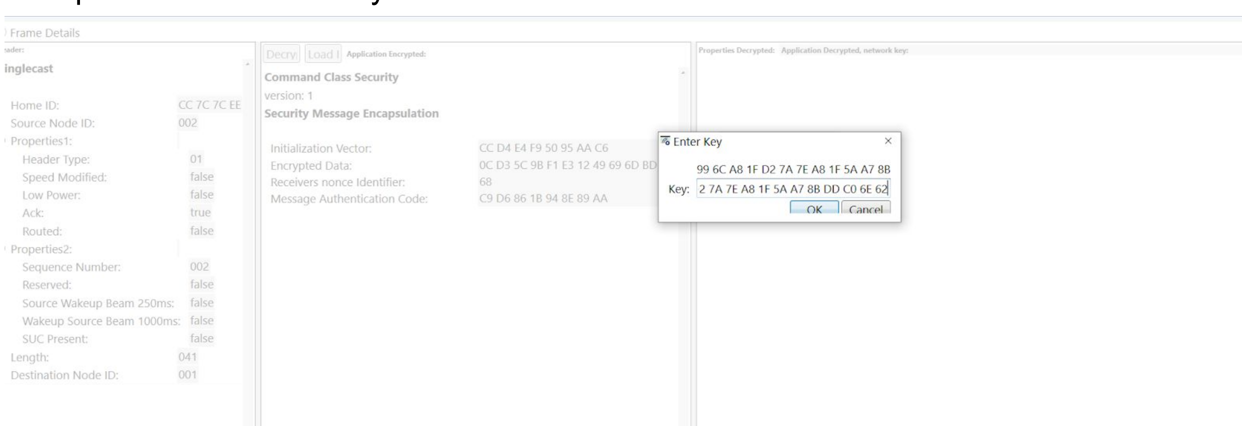
The above capture demonstrates the S0 layer flaw described above. Controller is sending the shared network key encrypted to the sensor. On the right, the network the key used to decrypt the ciphertext is 16 byte 0's key. The shared network key obtained from it (underlined red) can then be used to decrypt any message further exchange between the two.

Testing the decrypted key above,

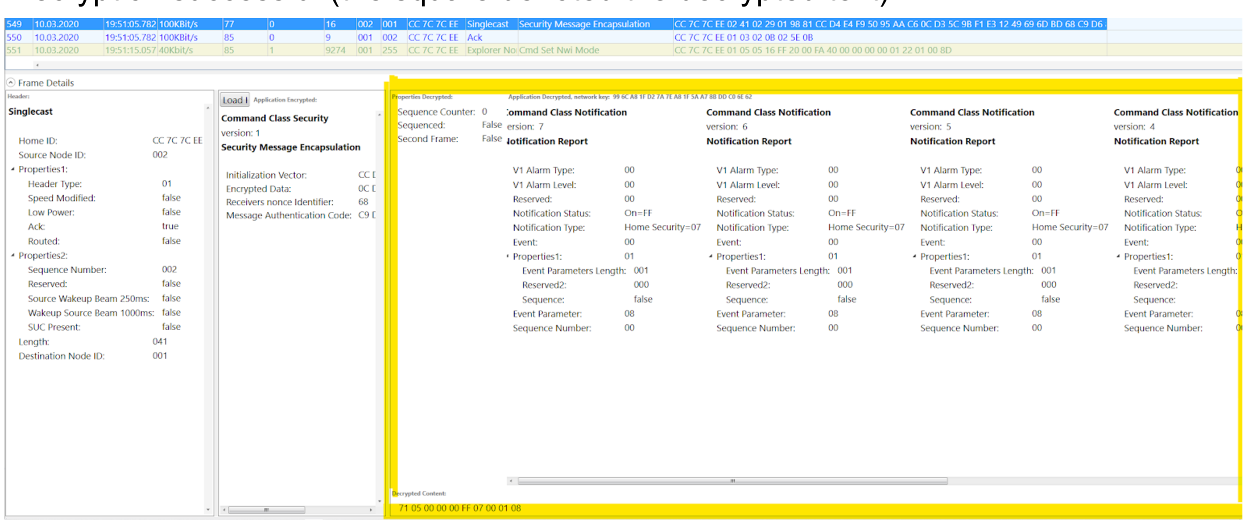
1) The frame below is encrypted (Security Message Encapsulation)



2) We pass the obtained key from above.



3) Decryption successful (the square denoted the decrypted text)



Section V
Z-Wave S2 Security Framework

Z-wave S2 Security Framework

Specifications

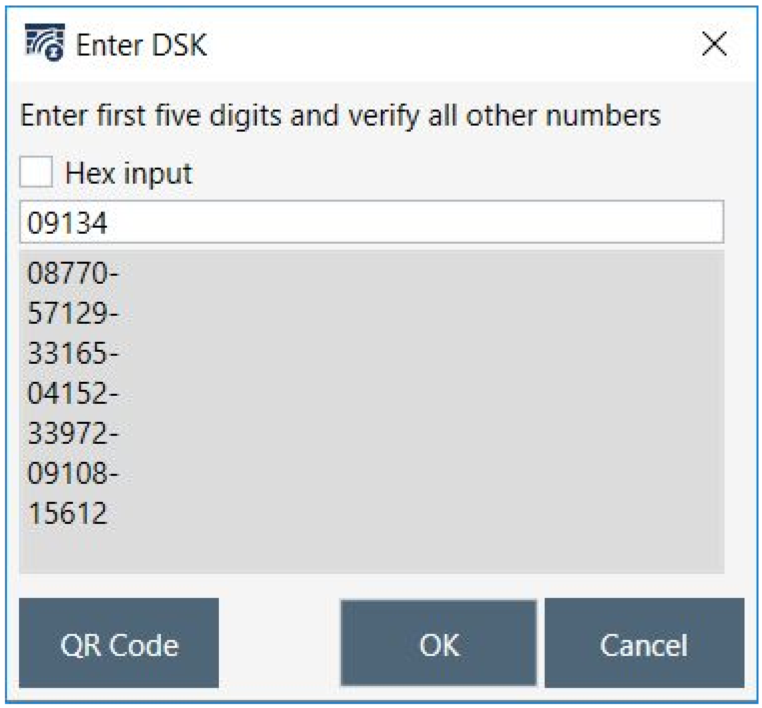
- Builds upon a secure inclusion process and encrypted communication.
- Covers for the flaws in S0 with in-band insecure transmission of shared network key during inclusion.
- Helps device manufacturers to focus more on functionality rather than on security.
- **Involves Device Specific Key (DSK)** , a unique key for every secure device.
    - Unique 40 digits for each device (Printed on the device)
    - 16 bytes in size
    - Also comes in QR code for easy reference
    - First 5 digits (2 bytes) of the key are required to be entered manually.
    - **This allows for validation of device identity.**
    - **Prevents man-in-the-middle attack**

Key Exchange Methods

- **Elliptic Curve Diffie-Hellman (ECDH)** → Provides the capability of shared secret between the two devices and addresses sender integrity.
- NOTE : ECDH public keys are only used during the inclusion process for establishment of a shared secret key (network key) between the primary controller and the authenticated device.
- **Elliptic Curve Cryptography (ECC)** → high cryptographic security + less computation cost in comparison to RSA.

Out of band key exchange for product authentication

- Prevents **eavesdropping** and **man -in-the-middle attack vectors.**
- **Removes the possibility for a man-in-the-middle impersonation of the devices during the inclusion process.**
- Used in S2-Authenticated and S2-AccessControl.
- Visual Scanning of the QR code / Entering the first 5 digits of the DSK in controller allows controller to validate that the device is real / not an impersonated one.
-



The first 5 digits of the DSK are replaced initially with 0s in the RF transmission. User is asked to enter them. This way, a user can confirm the identity of his device by comparing the remaining key with the one displayed.

Multiple Security Groups

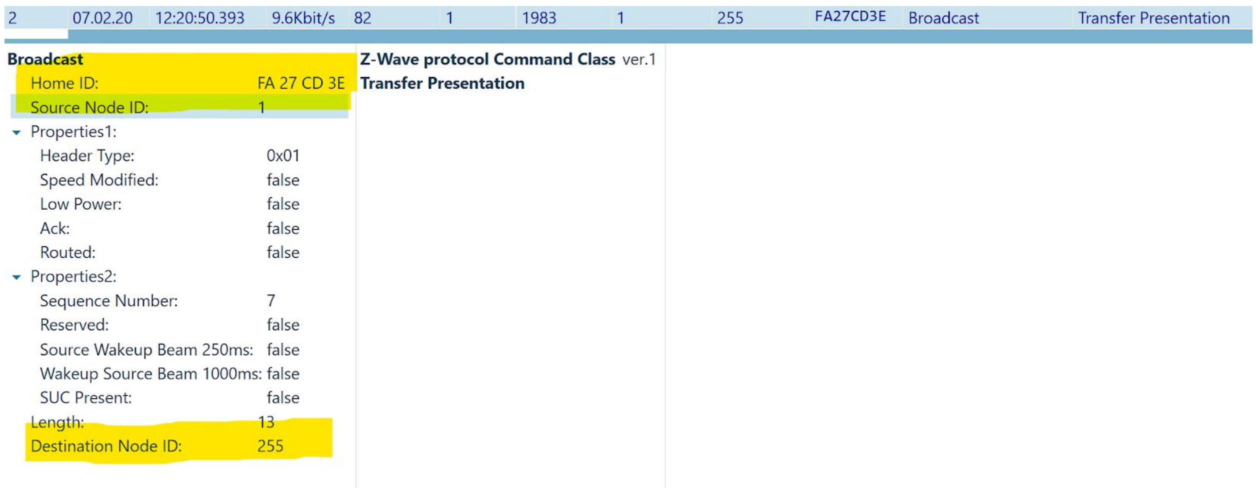
- With SDK 6.71 and later , a total of 5 Security groups have been introduced :
  - Unauthenticated devices
  - S0 Authenticated devices
  - S2 Unauthenticated devices
  - S2 Authenticated devices
  - S2 Access Control devices
- **Compartmentalization of secure devices (increased security)**
  - A device only knows the keys of the group it is a part of.
  - A device cannot control the functions of other devices if they don't belong to the same group.
  - This further gives advantage as **trying to extract secure keys from a non-secure device is not possible** - since the non-secure device has no knowledge of keys of other secure groups**.**
  - **A device can be a member of multiple groups.**
  - **All of the devices use the same 128-bit encryption mechanism. However, the key is different per group.**

    PUT IMAGE OF PC CONTROLLER KEYS HERE!!
- **S2 Unauthenticated**
  - These devices haven't implemented out-of-band inclusion methods (DSK or QR code)
  - These devices are irrelevant to include in the authenticated group. They could be end devices that are physically exposed.
  - They could be controlled by S2 Unauthenticated devices via associations.
- **S2 Authenticated**
  - Secure end devices such as Door/Window Sensors, Switches, Light/Temperature Sensor etc.
  - They could be secondary controllers that do not need to control **access control devices.**
  - **Devices in S2 Authenticated group cannot be controlled by devices in a lower security group.**
- **S2 Access Control**
  - Door locks
  - Garage door openers
  - Controllers that need to control access control devices.

- **Access control devices can only be accessed by the controllers that need to control them.**
- S2 access control group doesn’t have more secure communication than S2 Authenticated group

S2 Node Inclusion and Key Exchange Processes

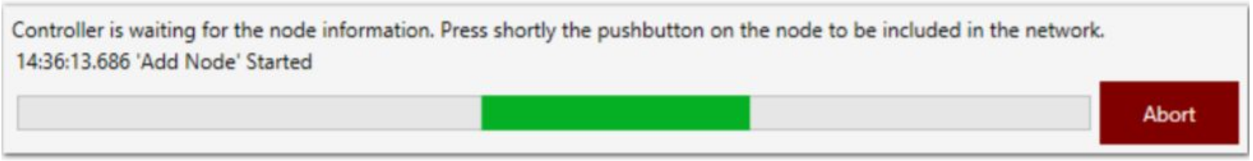
1) Press “Add Device” button on the controller (same as in case of S0)
2) The very first frame is broadcasted by the Z-wave controller in the network to introduce itself. This step involves 3 sub-steps to be completed:
   a) Home ID is generated by the controller (Home ID uniquely identifies a Z-wave network)



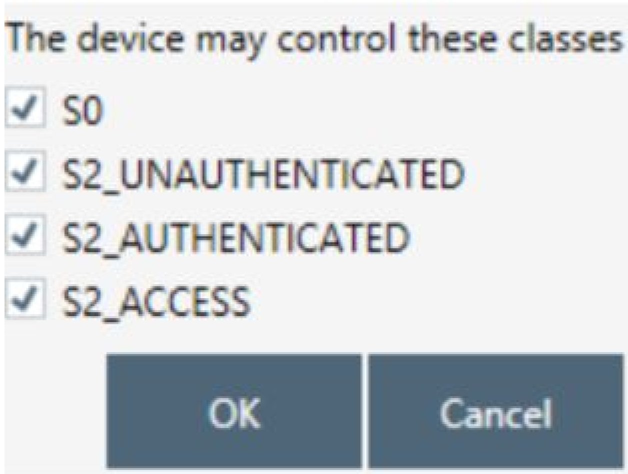
In this case, the Home ID us FA 27 CD 3E

   b) The second line of the Broadcast Frame Header shows the source Node ID, which uniquely identifies a device in a network. The controller has a source Node ID of 1.
   c) The Destination Node ID is 255 which means the frame is meant for all of the nodes in the network. (Total number of possible nodes in a network = 256 -1 = 255, based on 8-bit value of the source Node ID)
3) Press the “action button” on the sensor (Trisensor in this case)
4) For S2 security devices (such as the trisensor we are using in the this experiment) , we either scan the DSK (Device Specific Key) QR code or input the underlined part of the DSK (which is shown on the label)

After Pressing the add button on the controller, it is in listening state.

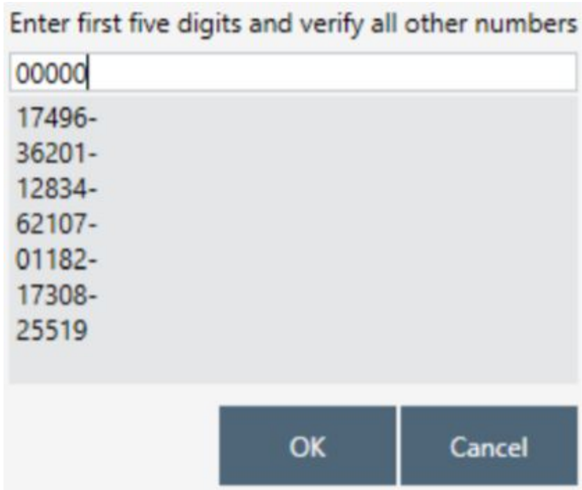


5) When controller reads the signature of the trisensor, it gets the following popup below:



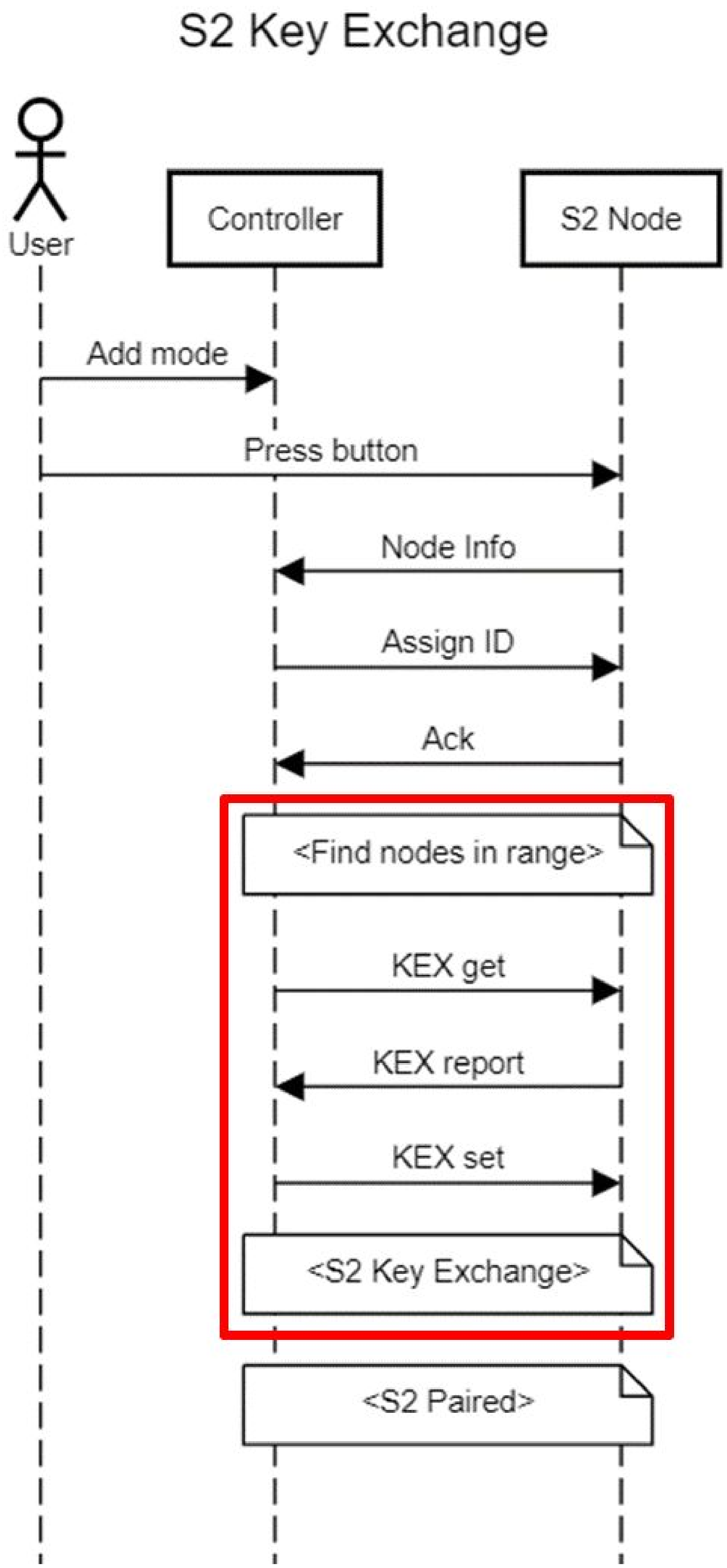
This means that the end-device (trisensor) supports the following four different groups of security (command classes) and asks the controller which groups the device to communicate with the controller . If we press the cancel button, the inclusion process will open non-securely. However, for secure S2 inclusion, we now require the DSK key shown above

6) The popup below shows the DSK request by the controller specific to the end-device which wants to join the network. In our case, we enter the first 5 digits of the DSK in the box below.



Successful inclusion will be denoted by trisensor getting a node ID as well as yellow light blinking , denoting the success as per its manual instructions.

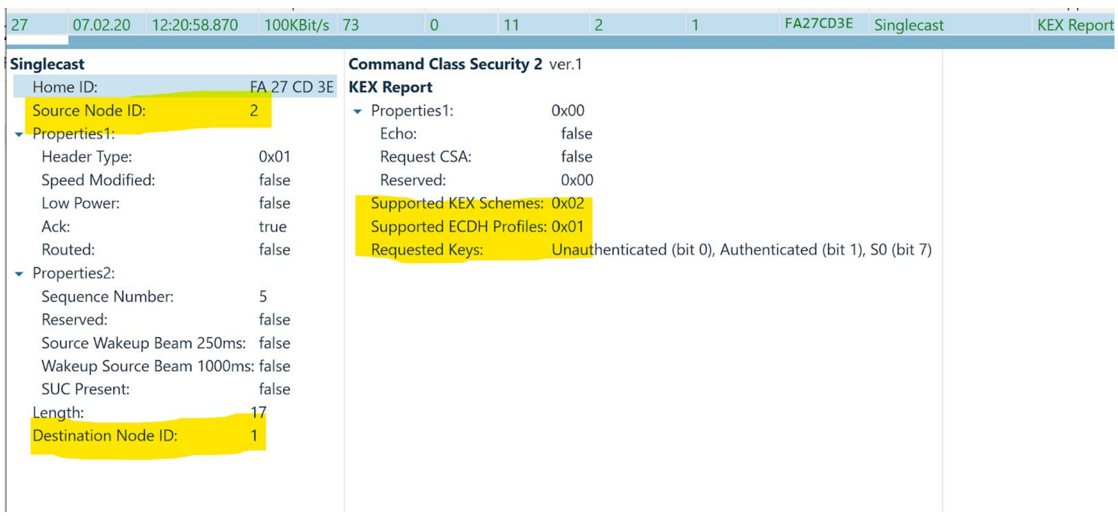
S2 Key Exchange



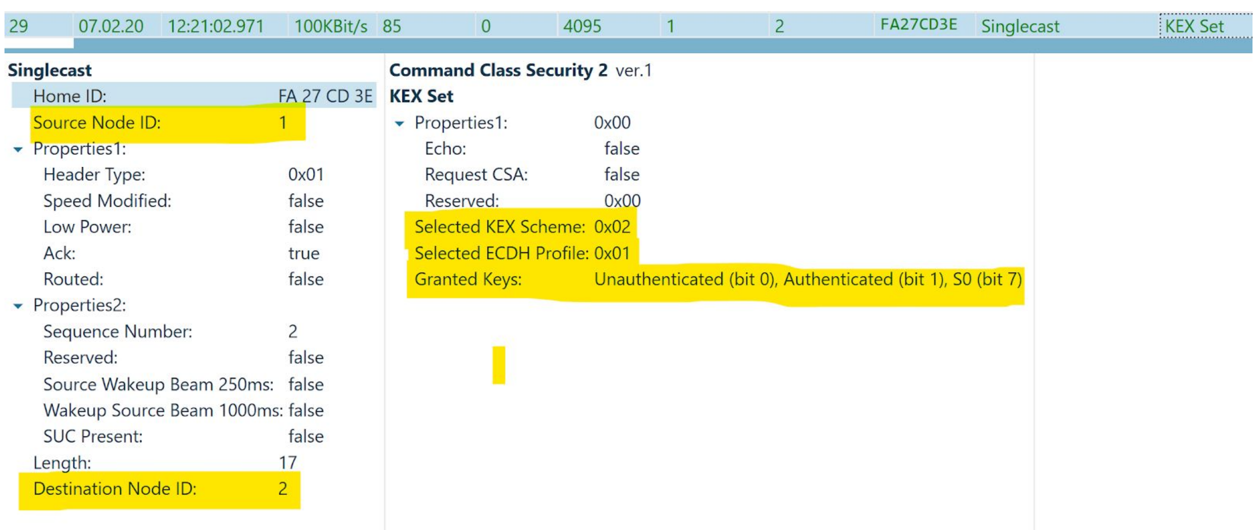
The steps in red rectangle describe the key exchange for S2 which are as follows:

1) Since the inclusion is successful, from this point onwards the frames exchanged between the controller as well as the tri sensor will be singlecast. Now, the controller sends **KEX Get** Application Command to the tri-sensor to get to know. The supported KEX schemes as well as ECDH profiles and the network keys the
   joining node intends to request.
   Command Class = **S2 KEX Get**
   Source Node ID = 1 (Controller)
   Destination Node Id = 2 (Tri-sensor)

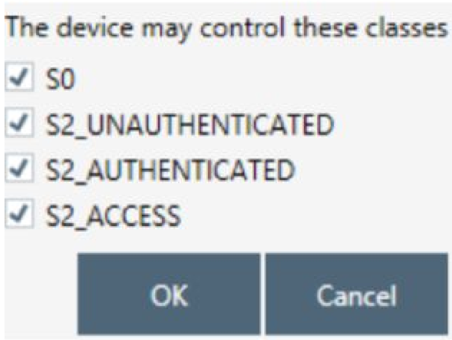
2) The Tri-sensor then sends a **KEX Report** to the controller , including the suite of KEX as well as ECDH profile it supports. The detailed report is as follows:
    - KEX scheme = 0x02
    - ECDH Profile = 0x01
    - Requested Keys =
        - Unauthenticated bit = 0
        - Authenticated bit = 1



3) The controller sends a KEX SET command to trisensor, confirming with the requested suite by the sensor.

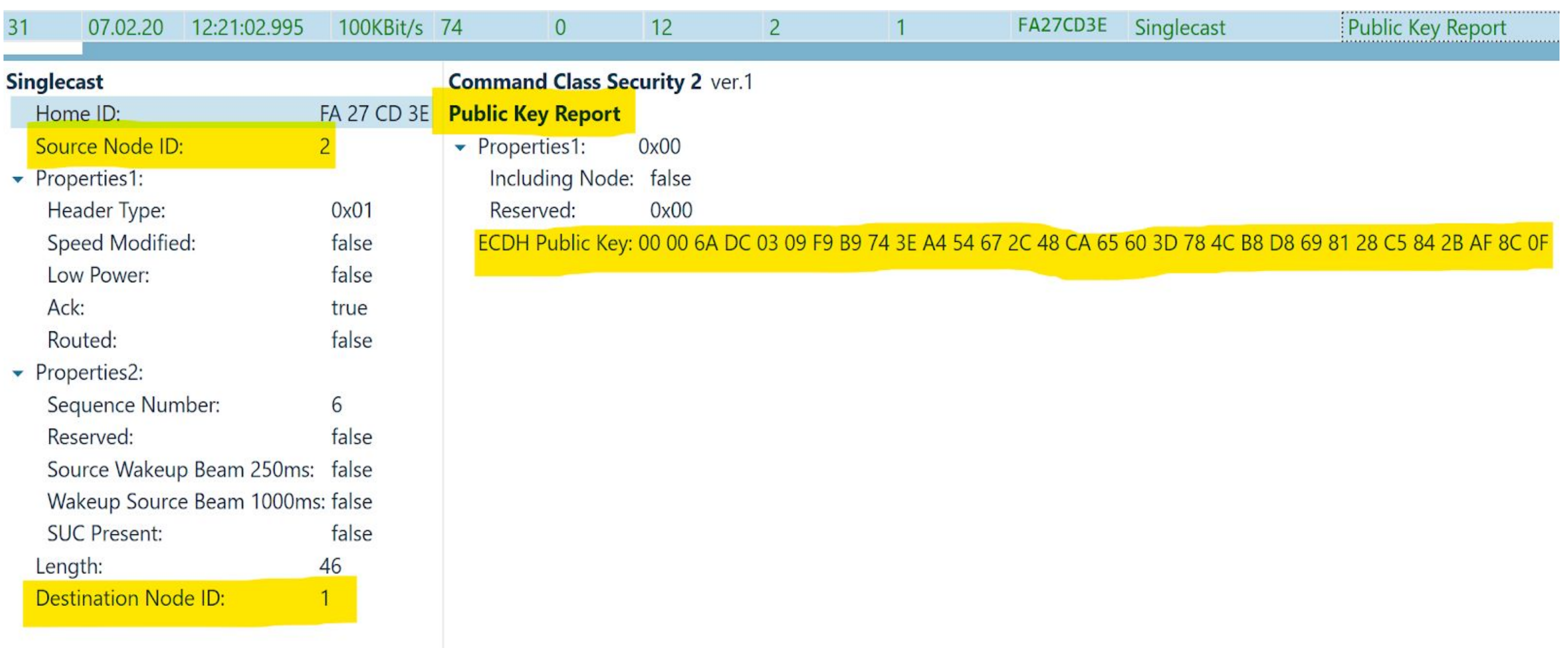


NOTE : in the granted keys section, 3 different keys are granted by the controller to the sensor. This correlates to the groups of security classes we allow the communication to happen between the controller and the sensor.

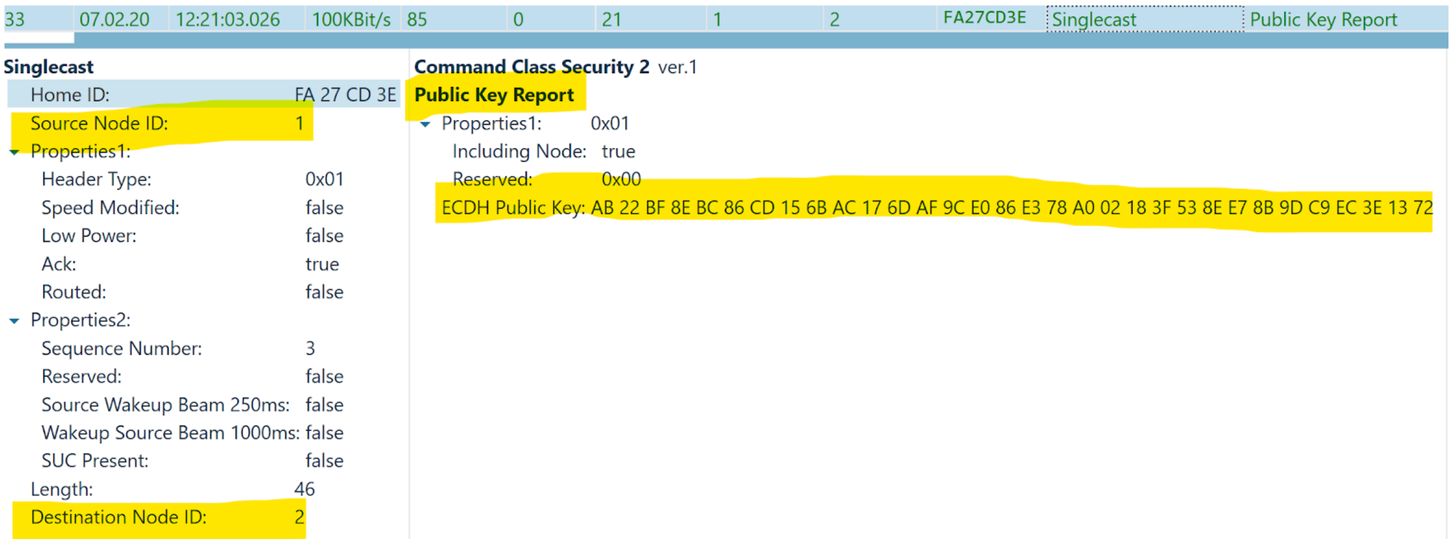


For the above case, imagine the first 3 checkboxes checked during the inclusion process on the controller.

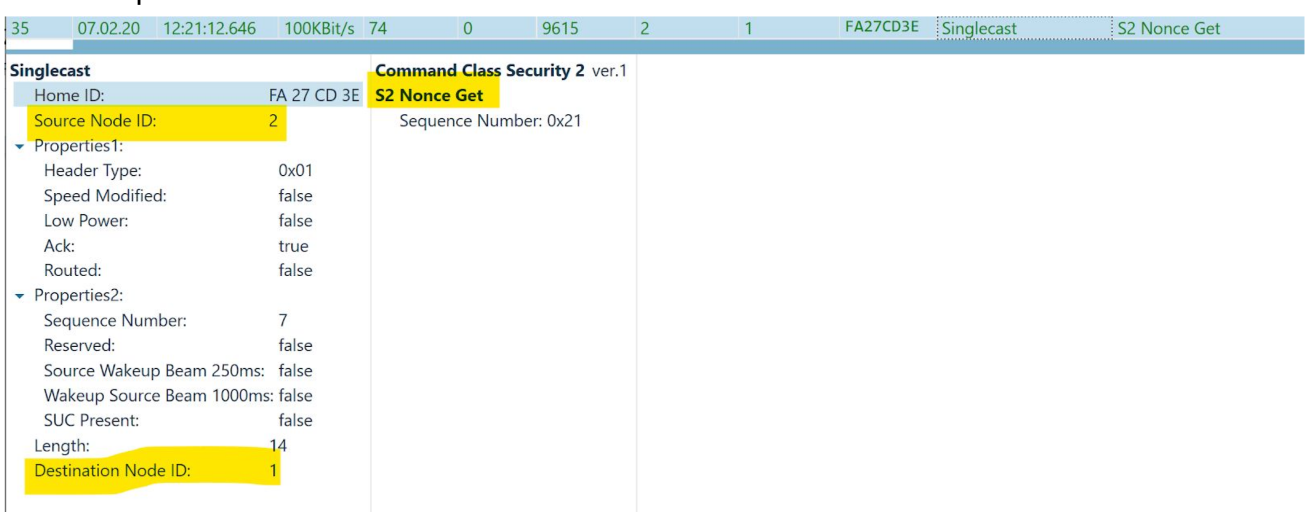
4) Now, the actual Network Key is exchanged between the Controller and the sensor with the help of Diffie Hellman Key Exchange Algorithm
First, Trisensor sends his public ECDH key to the controller.



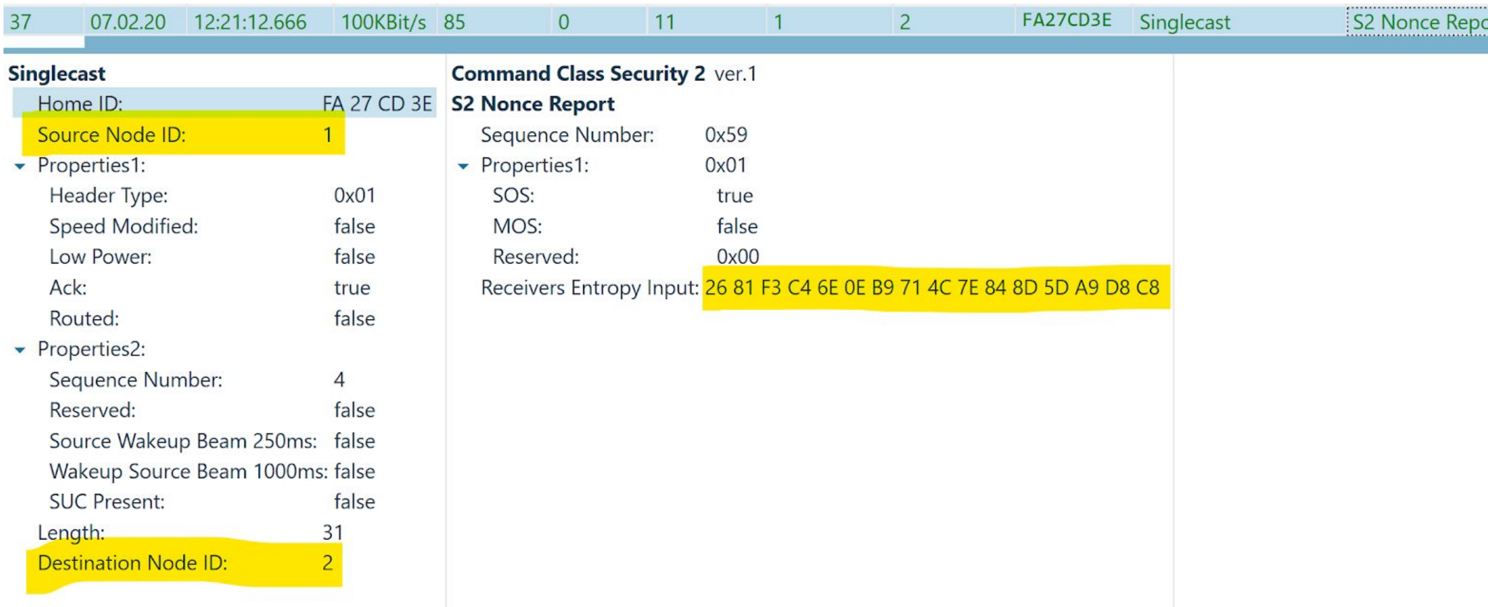
5) Next, the controller sends his ECDH public key to the sensor.



6) Using the DH keys exchanged above, the network key is shared between the controller and the sensor. Next, to ensure freshness as the same in S0, trisensor requests nonce from the controller.



7) Controller sends a Nonce Report to the sensor including its nonce.



8) Now, they follow the same encryption steps as S0