

DEPLOYMENT AND ANALYSIS OF Z-WAVE IOT DEVICES

RISHAB DHAMIJA



What are IoT ?

- Internet of Things (IoT) are physical devices connected to the internet.
- Used for the purpose of collecting and sharing data without involvement of a human being.
- Can be connected to one or more sensors to provide them with an added level of digital intelligence and to enable them to communicate real-time data.



EXAMPLES



What is Z-Wave?

- Developed by a Danish company named Zensys.
- A low power, low bandwidth **half-duplex two-way** protocol.
- Designed as a wireless communication technology for residential homes.
- Advantages over other IoT networking technologies:
 - Operates at sub-1GHz frequency avoiding heavily congested 2.4 GHz and 5 GHz bands where Wi-Fi and ZigBee are positioned.
 - Ensures 100% interoperability→ all devices that implement Z-wave will work together in one single network and can be controlled from every controller that also uses Z-wave.
 - Offers Security of Communication as well as low radio emission.



Figure 1.6: The first Z-Wave Controller made by Zensys in 2001

Source: Z-Wave Alliance



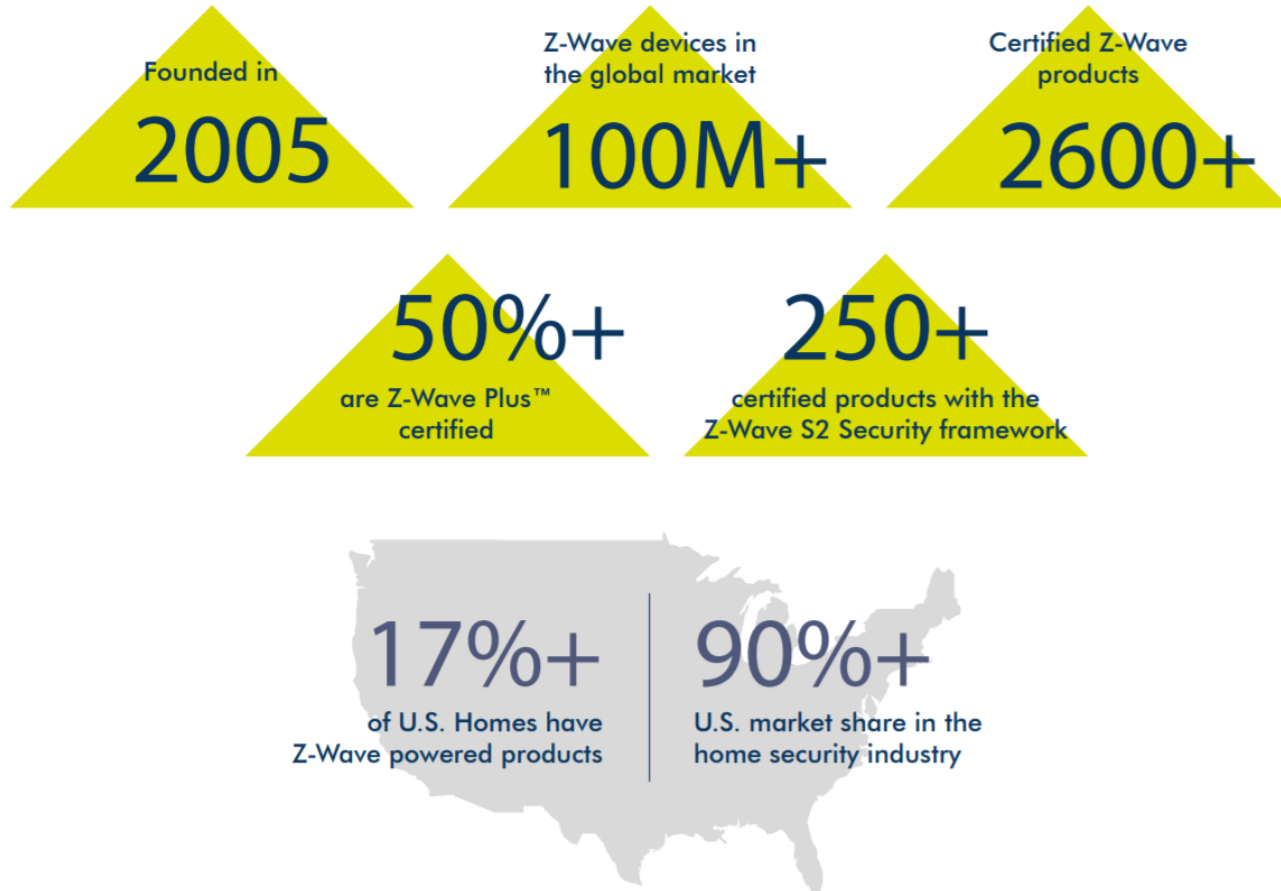
Z-Wave Alliance

- Established in 2005.
- Comprised of industry leaders across the globe that are dedicated to the development and extension of Z-Wave as the key enabling technology for “smart” home and businesses applications.
- An alliance of 700 companies with over 3000 products, all interoperable with each other.
- Assures the each Z-Wave product meets the standard for complete compliance with all other devices and controls.
- Assures reliability to the consumers, dealers and manufacturers that these.



Z-WAVE BY THE NUMBERS

AS AN EARLY LEADER, THE Z-WAVE ALLIANCE HAS ACHIEVED SUBSTANTIAL GROWTH AND MOMENTUM AS THE SMART HOME MARKET EXPANDS.



Z-Wave statistics 2018



Z-WAVE IN REAL WORLD

- Operates at 900 MHz (a clear advantage over other wireless protocols as they all operate at the same Frequency band)

Variable	Wi-Fi	Z-Wave	ZigBee	Thread	BLE
Year first launched in Market	1997	2003	2003	2015	2010
PHY/MAC Standard	IEEE 802.11.1	ITU-T G.9959	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.1
Frequency Band	2.4 GHz	900 MHz*	2.4 GHz	2.4 GHz	2.4 GHz
Nominal Range (0 dBm)	100 m	30 – 100 m	10 – 100 m	10 – 100 m	30 m
Maximum Data Rate	54 Mbit/s	40-100 kbit/s	250 kbit/s	250 kbit/s	1 Mbit/s
Topology	Star	Mesh	Mesh	Mesh	Scatternet
Power Usage	High	Low	Low	Low	Low
Alliance	Wi-Fi Alliance	Z-Wave Alliance	ZigBee Alliance	Thread Group	Bluetooth SIG

- Plays a major contribution in terms of choice of Radio Protocol for most of the IoT devices as shown in the second image on the right.

Product Category	Sample Size (# Products Purchasable in the US)	Number 1 Radio Protocol		Number 2 Radio Protocol		Number 3 Radio Protocol	
 Gateway / Hub	29	Wi-Fi	24%	Z-Wave	23%	ZigBee	17%
 Plug	41	Z-Wave	43%	Wi-Fi	36%	Bluetooth	5%
 Sensor: Door	26	Z-Wave	41%	ZigBee	24%	Wi-Fi	3%
 Thermostat	23	Wi-Fi	58%	Z-Wave	21%	ZigBee	17%
 Door Lock	19	Z-Wave	62%	Bluetooth	29%	Wi-Fi	5%
 Light Bulb: Color	19	Bluetooth	42%	ZigBee	37%	Wi-Fi	11%
 Scale	13	Bluetooth	67%	Wi-Fi	33%	N/A	
 Light Switch	12	Wi-Fi	36%	Z-Wave	36%	Bluetooth	14%
 Smoke Detector	9	Wi-Fi	33%	Z-Wave	22%	433 MHz	22%

Motivation

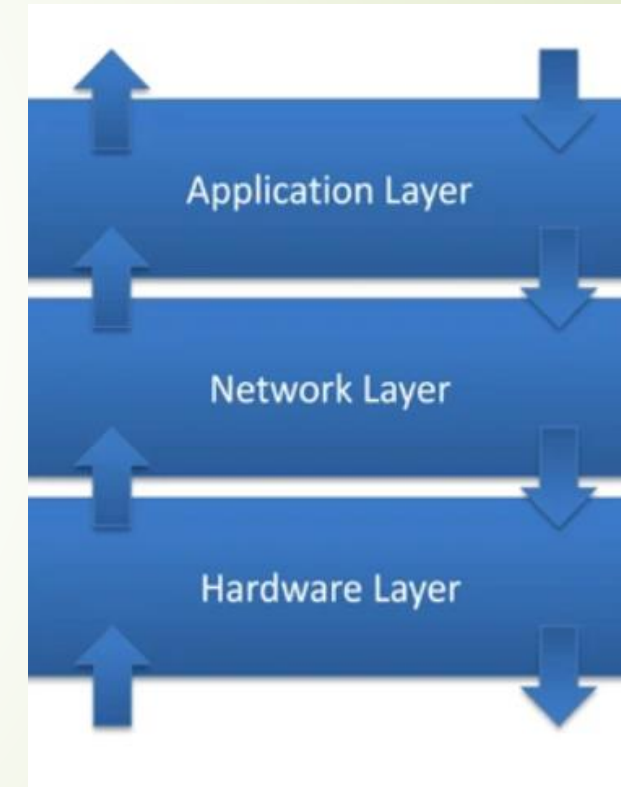
- The sky rocketing growth of Z-Wave devices being used in the world
- Very few research studies of the security of Z-Wave protocol based on real-world experimentation have been published.
- The primary motive behind this research to bridge that gap between the two.



Z-Wave Protocol Layers

Z-Wave comprises of three major layers :

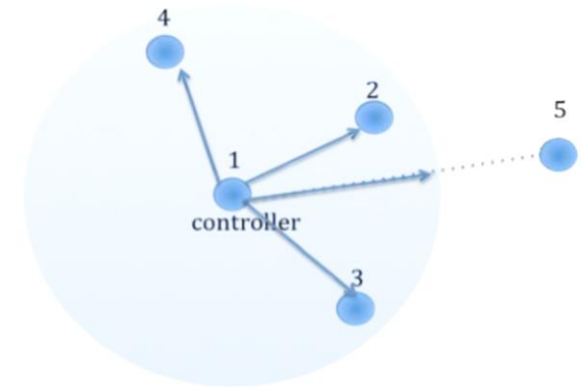
- **Radio Layer**: defines the way a signal is exchanged between network as well as physical radio hardware. Includes frequency, encoding, hardware access etc.
- **Network Layer**: defines how the data is exchanged between two devices or nodes. This layer's functionality is further divided into 3 sublayers:
 - **Media Access Layer (MAC)**
 - **Transport Layer**
 - **Routing Layer**
- **Application Layer**: defines which messages are to be handled by which specific applications to accomplish tasks such as switching a light on/off or changing the temperature of thermostat.



Meshing and Routing

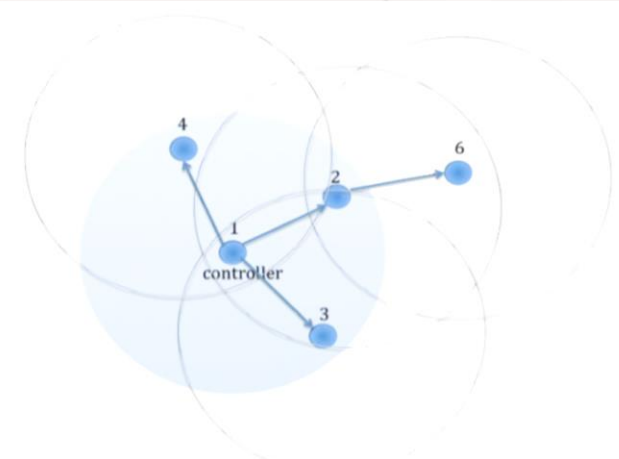
Network without Routing

- Node labelled 1 is the controller in the network.
- Only nodes 2, 3 and 4 lie within the radio ranges of the controller. Therefore, only these can be reached directly by the controller.
- Node 5, outside the radio range, cannot be reached by the controller.



Z-Wave network with routing

- Z-wave offers a powerful mechanism to overcome the problem highlighted above.
- Z-Wave nodes can forward and repeat messages that are not in the direct range of the controller.
- Node 1 is controller. It can communicate directly with nodes 2, 3 and 4.
- For node 6 being outside the radio range of controller, node 1 can communicate to 6 via node 2.
- A possible route = Node 1 → Node 2 → Node 6



SO Security Protocol

Involves Encryption + Message Authentication Code (MAC)

- **Message Integrity:** assures receiver that message sent is by a secure node in the network and is unaltered.
- **Data Freshness:** resembles that message has been sent recently (using Nonce).
- **Confidentiality:** only secure nodes in the network can read the actual contents of the message.

- **Flaw :** the shared network key is exchanged using 16-byte temporary key of all 0, which is publicly known.
- **Insecure against :** DoS attack , hardware side-channel attack, traffic analysis, protocol side-channel and application-layer security attacks.



S2

Security Protocol

- Uses the Diffie-Hellman Key Exchange process for exchanging the Network Key.
- **Elliptic Curve Diffie-Hellman (ECDH)** → Provides the capability of shared secret between the two devices and addresses sender integrity.
- Helps device manufacturers to focus more on the functionality rather than on the security of the device.
- **Involves Device Specific Key (DSK)** , a unique key for every secure device.
 - Unique 40 digits for each device (Printed on the device)
 - Also comes in QR code for easy reference
 - **This allows for validation of device identity.**
 - **Prevents man-in-the-middle attack**



Main Objectives

- Create a real-world Z-wave IoT network comprising of 2 devices (Trisensor as well as door/window sensor) and a controller.
- Build a framework for Z-Wave traffic/packet monitoring and analysis.
- Observe the use of S0 and S2 security modes in real-world Z-Wave network and identify the potential vulnerabilities of the two modes.



Experimental Framework

Deployed IoT devices:

- 1) Aeotec Trisensor (end-device #1)
- 2) Dome Door/Window Sensor (end-device #2)
- 3) Z-Wave Controller UZB-7
- 4) Z-Wave Zniffer UZB3-S

Software:

- 1) Z-Wave Zniffer
- 2) Z-wave PC Controller
- 3) Simplicity Studio (IDE)
- 4) Windows 10 (OS)



Challenges with Packet Sniffer

Primary choice of packet sniffer was Wireshark with added functionalities like dissector.

However, switching to Zniffer was result of several issues:

- 1) Wireshark does not support Z-wave primarily so using addons on top of it such as custom scripts and repositories wasn't guaranteed success, which in our case was a failure.
- 2) there was no specific resource or guide as how to make Wireshark as Zniffer as most of the results published on the web are just customization done by people as per their convenience. Those results are not standard and changed from one environment to another.



Challenges with Software/IDE



- The initial steps of research focused on using Ubuntu with customized scripts and software such as GNU Radio and Audacity.
- Second (and used) choice for Software and hardware was Silicon Labs with dedicated development Kits for developing and testing Z-wave networks.
- Kit used for our research is : [Z-Wave 700 Development kit](#)
- Pros of this approach:
 1. Standardized among all Z-wave devices.
 2. A reliable and resourceful platform for information regarding building network and testing it.

Cons of this approach:

1. The only way to learn for the dedicated software ([Simplicity Studio, Z-Wave PC Controller as well as Z-Wave Ziffer](#)) and hardware was to read manuals for each of them and correlating them.
2. No support exists for asking questions outside their community forum.
3. Every documentation, tutorial or resource is strictly tied to their website. So this is the only source of research.



S0 Protocol

DURING NETWORK KEY SHARING

The controller generates the Network Key and wishes to share it with the trisensor for future messages.

1. Controller → Nonce Get command to trisensor.
2. trisensor → Nonce Report command to controller.
3. Controller generates its own nonce, concatenates it with trisensor's Nonce to form 16-byte IV
4. IV = Sender (A's) Nonce || Receiver (B's) Nonce
5. Controller uses the temporary key **0x00 times 16** along with IV to encrypt and MAC the Network key Nk . It sends it to trisensor.
6. Trisensor decrypts it using the temporary key, and gets the main Network key Nk.

THIS IS THE ONLY SECTION WHERE THE TEMPORARY KEY WAS USED.

AFTER NETWORK KEY IS SHARED

1. trisensor → Nonce Get command to controller.
2. Controller → Nonce Report command to trisensor.
3. trisensor generates its own 8-byte nonce, concatenates it with controller's Nonce to form 16-byte IV
4. IV = Sender (A's) Nonce || Receiver (B's) Nonce
5. Since, the network key is already exchanged , 2 new keys are derived from it as follows (This is done on both trisensor and controller):

1. AES-encryption key = AES-ECB(Nk, Password-e)
2. MAC-authentication key = AES-ECB(Nk, Password-m)

Where Password-c is **0x55 times 16** and Password-m is **0xAA times 16**. These values are hardcoded in the Z-wave firmware.

6. Trisensor uses these 2 keys along with the IV computed to encrypt and MAC-authenticate the frame and sends it to controller.
7. Controller also uses these 2 keys along with IV to authenticate and then decrypt the frame.



Obtained Results : S0

Frames / sec	Line	Speed	RSSI	Ch	Delta	Src	Dst	Home	Data	Application	Hex Data
2	361 21	9.6Kbit/s	78	1	93	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241010E010118F0009F
1	362 35	9.6Kbit/s	78	1	213	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241010E010118F0009F
6	363 39	9.6Kbit/s	78	1	134	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241020E010118F0009C
5	364 32	9.6Kbit/s	78	1	213	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241020E010118F0009C
4	365 74	9.6Kbit/s	78	1	94	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241030E010118F0009C
2	366 79	9.6Kbit/s	78	1	134	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241030E010118F0009D
0	367 33	9.6Kbit/s	78	1	273	2	1	DAB954F3	Singlecast	NOP Power	DAB954F30241030E010118F0009D
0	368 37	40Kbit/s	85	1	13119	1	255	CC7C7CEE	Explorer Normal	Cmd Set Nwi Mode	CC7C7CEE01050316FF2000FA400000000012200008A
0	369 33	9.6Kbit/s	86	1	1982	1	255	CC7C7CEE	Broadcast	Transfer Presentation	CC7C7CEE01010100FF01080522
0	370 35	9.6Kbit/s	86	1	1972	1	255	CC7C7CEE	Broadcast	Transfer Presentation	CC7C7CEE01010100FF01080522
1	371 22	9.6Kbit/s	66	1	447	0	255	DD1AB265	Broadcast	Node Info	DD1AB2650010124FF0101539C0107015E989F558673E
0	372 48	9.6Kbit/s	85	1	43	1	0	DD1AB265	Singlecast	Assign Id	DD1AB2650141031100010302CC7C7CEE9F
0	373 77	9.6Kbit/s	73	1	35	0	1	DD1AB265	Ack		DD1AB2650003030A01E4
1	374 30	9.6Kbit/s	85	1	20	1	2	CC7C7CEE	Singlecast	NOP	CC7C7CEE01410400020090
0	375 49	9.6Kbit/s	73	1	53	2	1	CC7C7CEE	Ack		CC7C7CEE0203040A01D3
0	376 33	100Kbit/s	85	0	18	1	2	CC7C7CEE	Singlecast	Find Nodes In Range	CC7C7CEE0141051102010401010003107A
0	377 71	100Kbit/s	74	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE0203050001052A
1	378 31	100Kbit/s	73	0	20	2	1	CC7C7CEE	Singlecast	NOP Power	CC7C7CEE0241020F01011860066C4B
0	379 30	100Kbit/s	85	0	7	1	2	CC7C7CEE	Ack		CC7C7CEE01030200025E0B
0	380 38	100Kbit/s	74	0	10	2	1	CC7C7CEE	Singlecast	Command Complete	CC7C7CEE0241030E01010700BFC3
1	381 15	100Kbit/s	85	0	7	1	2	CC7C7CEE	Ack		CC7C7CEE0103030002693B
0	382 21	100Kbit/s	85	0	6	1	2	CC7C7CEE	Singlecast	Get Nodes In Range	CC7C7CEE01410600020105B080
1	383 28	100Kbit/s	74	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE02030600015C7A
0	384 40	100Kbit/s	77	0	10	2	1	CC7C7CEE	Singlecast	Node Range Info	CC7C7CEE0241040F0101060101DDF0
1	385 46	100Kbit/s	85	0	7	1	2	CC7C7CEE	Ack		CC7C7CEE0103040002ECAB
0	386 50	100Kbit/s	85	0	304	1	2	CC7C7CEE	Singlecast	Security Scheme Get	CC7C7CEE0141070E029804005930
0	387 57	100Kbit/s	80	0	7	2	1	CC7C7CEE	Ack		CC7C7CEE02030700016B4A
0	388 33	100Kbit/s	81	0	26	2	1	CC7C7CEE	Singlecast	Security Scheme Report	CC7C7CEE0241050E01980500B2E8
0	389 30	100Kbit/s	85	0	6	1	2	CC7C7CEE	Ack		CC7C7CEE0103050002D89B
0	390 39	100Kbit/s	85	0	50	1	2	CC7C7CEE	Singlecast	Security Nonce Get	CC7C7CEE01410800029840C53A
0	391 46	100Kbit/s	81	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE0203080001477B
0	392 52	100Kbit/s	81	0	13	2	1	CC7C7CEE	Singlecast	Security Nonce Report	CC7C7CEE0241061501980769C63D670373597A69
0	393 57	100Kbit/s	85	0	8	1	2	CC7C7CEE	Ack		CC7C7CEE010306000282CB
0	394 21	100Kbit/s	85	0	29	1	2	CC7C7CEE	Singlecast	Network Key Set • SOME	CC7C7CEE01410931029881E86E92002826C92D655504E
0	395 37	100Kbit/s	81	0	11	2	1	CC7C7CEE	Ack		CC7C7CEE0203090001704B
0	396 39	100Kbit/s	81	0	32	2	1	CC7C7CEE	Singlecast	Security Nonce Get	CC7C7CEE024107000198402111

Singlecast

Home ID:	CC 7C 7C EE
Source Node ID:	1
Properties1:	
Header Type:	0x01
Speed Modified:	false
Low Power:	false
Ack:	true
Routed:	false
Properties2:	
Sequence Number:	9
Reserved:	false
Source Wakeup Beam 250ms:	false
Wakeup Source Beam 1000ms:	false
SUC Present:	false
Length:	49
Destination Node ID:	2

Command Class Security ver.1

Network Key Set

Network Key byte: 99 6C A8 1F D2 7A 7E A8 1F 5A A7 8B DD C0 6E 62

Decrypted:

Network Key: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Properties: 0x00

Content: 98 06 99 6C A8 1F D2 7A 7E A8 1F 5A A7 8B DD C0 6E 62

Command Class Security ver.1

Security Message Encapsulation

Initialization Vector byte: E8 6E 92 00 28 26 C9 2D

Properties1:

Sequence Counter: 0x05

Sequenced: false

Second Frame: true

Reserved: 0x01

Command byte: 55 04 D1 DB BC F5 52 FB 25 16 0C EB 5D 88 65 E7 11 51

Receivers nonce Identifier: 0x76

Message Authentication Code byte: D4 65 11 CA 22 AA A4 2B

Based on the observed results above, the shared network key was exchanged using the **temporary 16 byte all 0's network key**. Since, this key is known to everyone, it proved to be a major flaw as seen in the slides further.

Taking note from above, the network key obtained is **99 6C A8 1F D2 7A 7E A8 1F 5A A7 8B DD C0 6E 62**.

Lets use this key to decrypt the future encapsulated messages between the controller and the trisensor.



Obtained Results : S0

Here, we use the key obtained from the previous step to decrypt traffic between controller and the Trisensor.

Entering the key

99 6C A8 1F D2 7A 7E
A8 1F 5A A7 8B DD C0
6E 62.

The image shows a Wireshark packet capture interface. The main pane displays a list of frames with columns for Line, Speed, RSSI, Ch, Delta, Src, Dst, Home, Data, Application, and Hex Data. The frames are numbered from 2 to 404. The 'Data' column shows various protocols like Singlecast, Broadcast, and Explorer Normal. The 'Application' column shows various actions like NOP Power, Transfer Presentation, Node Info, Assign Id, Find Nodes In Range, Command Complete, Get Nodes In Range, Node Range Info, Security Scheme Get, Security Scheme Report, Security Nonce Get, Security Nonce Report, Network Key Set - SOME, Security Nonce Get, Security Nonce Report, Security Message Encapsulation, and Request Node Info. The 'Hex Data' column shows the corresponding hexadecimal data for each frame.

On the right side, there is a 'Singlecast' pane with fields for Home ID, Source Node ID, Properties1, Properties2, and Sequence Number. The 'Decrypted' pane shows the decrypted data for the selected frame, including the Command Class Security ver.1 Security Message Encapsulation.

A 'Enter Key' dialog box is open in the foreground, prompting the user to enter a key. The key is displayed as 99 6C A8 1F D2 7A 7E A8 1F 5A A7 8B DD C0 6E 62. The dialog box has a 'Key:' label and a 'Is Temp' checkbox.



Obtained Results : S0

Frames / sec	Line	Speed	RSSI	Ch	Delta	Src	Dst	Home	Data	Application	Hex Data
2	373 77	9.6Kbit/s	73	1	35	0	1	DD1AB265	Ack		DD1AB2650003030A01E4
1	374 38	9.6Kbit/s	85	1	20	1	2	CC7C7CEE	Singlecast	NOP	CC7C7CEE0141040B020090
6	375 49	9.6Kbit/s	73	1	53	2	1	CC7C7CEE	Ack		CC7C7CEE0203040A01D3
5	376 53	100Kbit/s	85	0	18	1	2	CC7C7CEE	Singlecast	Find Nodes In Range	CC7C7CEE0141051102010401010003107A
4	377 71	100Kbit/s	74	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE0203050B01052A
2	378 31	100Kbit/s	73	0	20	2	1	CC7C7CEE	Singlecast	NOP Power	CC7C7CEE0241020F01011860066C4B
0	379 38	100Kbit/s	85	0	7	1	2	CC7C7CEE	Ack		CC7C7CEE0103020B025E0B
0	380 38	100Kbit/s	74	0	10	2	1	CC7C7CEE	Singlecast	Command Complete	CC7C7CEE0241030E01010700BFC3
0	381 15	100Kbit/s	85	0	7	1	2	CC7C7CEE	Ack		CC7C7CEE0103030B02693B
0	382 21	100Kbit/s	85	0	6	1	2	CC7C7CEE	Singlecast	Get Nodes In Range	CC7C7CEE0141060D020105B080
0	383 28	100Kbit/s	74	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE0203060B015C7A
0	384 40	100Kbit/s	77	0	10	2	1	CC7C7CEE	Singlecast	Node Range Info	CC7C7CEE0241040F0101060101DDF0
0	385 46	100Kbit/s	85	0	7	1	2	CC7C7CEE	Ack		CC7C7CEE0103040B02ECAB
0	386 50	100Kbit/s	85	0	304	1	2	CC7C7CEE	Singlecast	Security Scheme Get	CC7C7CEE0141070E029804005930
0	387 57	100Kbit/s	80	0	7	2	1	CC7C7CEE	Ack		CC7C7CEE0203070B016B4A
0	388 33	100Kbit/s	81	0	26	2	1	CC7C7CEE	Singlecast	Security Scheme Report	CC7C7CEE0241050E01980500B2E8
0	389 30	100Kbit/s	85	0	6	1	2	CC7C7CEE	Ack		CC7C7CEE0103050B02DB9B
0	390 39	100Kbit/s	85	0	50	1	2	CC7C7CEE	Singlecast	Security Nonce Get	CC7C7CEE0141080D029840C53A
0	391 46	100Kbit/s	81	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE0203080B01477B
0	392 52	100Kbit/s	81	0	13	2	1	CC7C7CEE	Singlecast	Security Nonce Report	CC7C7CEE02410615019880769C6C3D670373597A69
0	393 57	100Kbit/s	85	0	8	1	2	CC7C7CEE	Ack		CC7C7CEE0103060B0282CB
1	394 31	100Kbit/s	85	0	29	1	2	CC7C7CEE	Singlecast	Network Key Set • SOME	CC7C7CEE01410931029881E86E92002826C92D655504C
1	395 37	100Kbit/s	81	0	11	2	1	CC7C7CEE	Ack		CC7C7CEE0203090B01704B
0	396 39	100Kbit/s	81	0	32	2	1	CC7C7CEE	Singlecast	Security Nonce Get	CC7C7CEE0241070D0198402111
1	397 46	100Kbit/s	85	0	6	1	2	CC7C7CEE	Ack		CC7C7CEE0103070B02B5FB
0	398 52	100Kbit/s	85	0	14	1	2	CC7C7CEE	Singlecast	Security Nonce Report	CC7C7CEE01410A15029880453B1850D1C36E1B2B2E
1	399 58	100Kbit/s	81	0	9	2	1	CC7C7CEE	Ack		CC7C7CEE02030A0B01291B
1	400 37	100Kbit/s	81	0	16	2	1	CC7C7CEE	Singlecast	Network Key Verify • SOME	CC7C7CEE02410B21019881A9D2E78F0B4D0922A020294
0	401 33	100Kbit/s	85	0	8	1	2	CC7C7CEE	Ack		CC7C7CEE01030B0B0299CA
0	402 56	100Kbit/s	85	0	562	1	2	CC7C7CEE	Singlecast	Request Node Info	CC7C7CEE01410B0D020102E11D
0	403 30	100Kbit/s	81	0	8	2	1	CC7C7CEE	Ack		CC7C7CEE02030B0B011E2B

Singlecast
Home ID: CC 7C 7C EE
Source Node ID: 2
Properties1:
Header Type: 0x01
Speed Modified: false
Low Power: false
Ack: true
Routed: false
Properties2:
Sequence Number: 8
Reserved: false
Source Wakeup Beam 250ms: false
Wakeup Source Beam 1000ms: false
SUC Present: false
Length: 33
Destination Node ID: 1

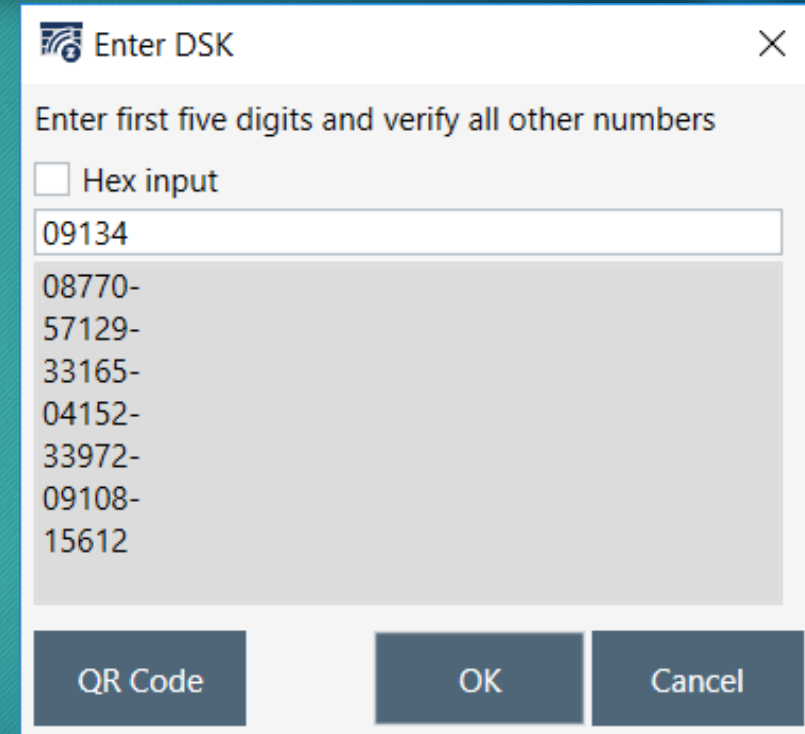
Command Class Security ver.1
Network Key Verify
Decrypted:
Network Key: 99 6C A8 1F D2 7A 7E A8 1F 5A A7 8B DD C0 6E 62
Properties: 0x00
Content: 98 07
Command Class Security ver.1
Security Message Encapsulation
Initialization Vector byte: A9 D2 E7 8F 0B 4D 09 22
Properties1:
Sequence Counter: 0xA0
Sequenced: false
Second Frame: true
Reserved: 0x02
Command byte: 20 29
Receivers nonce Identifier: 0x45
Message Authentication Code byte: 47 1E 10 9D 67 AD EA 81

Success!! Using the key 99 6C A8 1F D2 7A 7E A8 1F 5A A7 8B DD C0 6E 62,
We were able to decrypt the traffic.



S2 Protocol: 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.
- 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.



Enter DSK

Enter first five digits and verify all other numbers

☐ Hex input

09134

08770-
57129-
33165-
04152-
33972-
09108-
15612

QR Code OK Cancel

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.

S2 Node Inclusion

Step 1: Press “Add Device” button on the controller (same as in case of S0)

Step 2: The very first frame is broadcasted by the Z-wave controller in the network to introduce itself.

Step 3: Press the “action button” on the sensor (Trisensor in this case)

Step 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).

2	07.02.20	12:20:50.393	9.6Kbit/s	82	1	1983	1	255	FA27CD3E	Broadcast	Transfer Presentation
Broadcast											
Z-Wave protocol Command Class ver.1											
Transfer Presentation											
Home ID: FA 27 CD 3E											
Source Node ID: 1											
▼ Properties1:											
Header Type: 0x01											
Speed Modified: false											
Low Power: false											
Ack: false											
Routed: false											
▼ Properties2:											
Sequence Number: 7											
Reserved: false											
Source Wakeup Beam 250ms: false											
Wakeup Source Beam 1000ms: false											
SUC Present: false											
Length: 13											
Destination Node ID: 255											



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



← --- Note: This is the vulnerable step as attacker can uncheck the S2 boxes and make device connect in S0 (Z-Shave attack)



Enter first five digits and verify all other numbers

- 17496-
- 36201-
- 12834-
- 62107-
- 01182-
- 17308-
- 25519

OK Cancel

The popup 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 on the left.



S2 Key Exchange using Diffie-Hellman

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.

31	07.02.20	12:21:02.995	100KBit/s	74	0	12	2	1	FA27CD3E	Singlecast	Public Key Report
Singlecast											
Home ID: FA 27 CD 3E											
Source Node ID: 2											
Properties1:											
Header Type: 0x01											
Speed Modified: false											
Low Power: false											
Ack: true											
Routed: false											
Properties2:											
Sequence Number: 6											
Reserved: false											
Source Wakeup Beam 250ms: false											
Wakeup Source Beam 1000ms: false											
SUC Present: false											
Length: 46											
Destination Node ID: 1											
Command Class Security 2 ver.1											
Public Key Report											
Properties1: 0x00											
Including Node: false											
Reserved: 0x00											
ECDH Public Key: 00 00 6A DC 03 09 F9 B9 74 3E A4 54 67 2C 48 CA 65 60 3D 78 4C B8 D8 69 81 28 C5 84 2B AF 8C 0F											

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

33	07.02.20	12:21:03.026	100KBit/s	85	0	21	1	2	FA27CD3E	Singlecast	Public Key Report
Singlecast											
Home ID: FA 27 CD 3E											
Source Node ID: 1											
Properties1:											
Header Type: 0x01											
Speed Modified: false											
Low Power: false											
Ack: true											
Routed: false											
Properties2:											
Sequence Number: 3											
Reserved: false											
Source Wakeup Beam 250ms: false											
Wakeup Source Beam 1000ms: false											
SUC Present: false											
Length: 46											
Destination Node ID: 2											
Command Class Security 2 ver.1											
Public Key Report											
Properties1: 0x01											
Including Node: true											
Reserved: 0x00											
ECDH Public Key: AB 22 BF 8E BC 86 CD 15 6B AC 17 6D AF 9C E0 86 E3 78 A0 02 18 3F 53 8E E7 8B 9D C9 EC 3E 13 72											



Obtained Results: S2

- Decrypting the traffic using S2 is not possible since the network key is exchanged using Diffie-Hellman, contrary to the unsecure approach used in S0.

- Moreover, S2 kept all of the features of S0 in addition to its Out of Band Key Exchange and DH , making it even more secure.

41	07.02.20	12:21:12.729	100KBit/s	85	0	14	1	2	FA27CD3E	Singlecast	S2 Message Encapsulation
----	----------	--------------	-----------	----	---	----	---	---	----------	------------	--------------------------

Singlecast
Home ID: FA 27 CD 3E
Source Node ID: 1
▼ Properties1:
Header Type: 0x01
Speed Modified: false
Low Power: false
Ack: true
Routed: false
▼ Properties2:
Sequence Number: 5
Reserved: false
Source Wakeup Beam 250ms: false
Wakeup Source Beam 1000ms: false
SUC Present: false
Length: 29
Destination Node ID: 2

Decrypted:
Receiver Nonce: 26 81 F3 C4 6E 0E B9 71 4C 7E 84 8D 5D A9 D8 C8
Sender Nonce: BC 5A CA D8 BE 65 C9 34 DC 13 00 DA 2A C8 A6 FC
Command Class Security 2 ver.1
S2 Message Encapsulation
Sequence Number: 0x5A
▼ Properties1: 0x00
Extension: false
Encrypted Extension: false
Reserved: 0x00
vg1:
CCM Ciphertext Object: 62 91 AD BD F0 30 8D FA E2 EA 9D E7 C6 73



Future Work

Additional attacks that can be tested in the developed framework include:

- **DoS and DDoS attacks:** can be can be actively performed on the trisensor.
- **Hijacking Attack :** Spoofing a controller, especially in case of unsecure devices in highly favourable. An attacker can capture traffic of some other node in the network and get useful information such as Node ID of the controller as well as the Home ID of the network. The attacker can intervene by capturing the frame from controller to the slave, change the source Node ID from 1 (the original primary controller's ID) to his own (for example 2) and send the packet. The attacker can now wait and listen for any incoming packets on his Node ID.
- **Z-Shave Attack :** this attack behaves like man-in-the-middle attack by downgrading the security protocols supported response from the IoT device to the controller. Downgrading the protocol from S2 to S0 makes it vulnerable to the flaw we covered in the results of S0. **This attack is of high concern in the market as it puts millions of IoT devices at risk.**





Thank You

