

## Introduction

Whether you're a homeowner or a manager of a mission critical automatic industrial production line, security is an issue that you should be concerned about. Tech savvy burglars can use the unsecured transmission from your garage door remote to easily enter your home. Saboteurs can cripple a production line by targeting an unsecured wireless control system. Security is an issue for every wireless system.

## Objectives

- Security modes
- Cryptographic key, transport, establishment
- Joining a secure network

\*\*\* by security, do you mean, like, a hug? \*\*\*

---

# Module Topics

<b>ZigBee Security.....</b>	<b>10-1</b>
<i>Module Topics.....</i>	<i>10-3</i>
Security and Modes .....	10-5
<i>Frames with Security .....</i>	<i>10-6</i>
<i>Trust Center .....</i>	<i>10-7</i>
<i>Cryptographic Keys .....</i>	<i>10-8</i>
<i>Joining a Secure Network .....</i>	<i>10-9</i>
<i>Features .....</i>	<i>10-11</i>
<i>Lab10- Security .....</i>	<i>10-13</i>
Hardware list: .....	10-14
Software list:.....	10-14
Procedure.....	10-15
Initial setup .....	10-15
Turn Security On .....	10-17
Network Access Control.....	10-21

\*\*\* but hey, it looks good \*\*\*

## Security and Modes

### 802.15.4 Security

Defined security services include:

- ◆ **Access Control**
  - ◆ Communications based on received frame address
- ◆ **Data Encryption**
  - ◆ Frames can be encrypted with a symmetric key
- ◆ **Frame Integrity**
  - ◆ Prevents messages from being altered in transit
- ◆ **Sequential Freshness**
  - ◆ Counters prevent replay attacks



Security Modes ...

2

### Security Modes

- ◆ **Unsecured**
  - ◆ Messages are sent in the clear
- ◆ **Access Control List**
  - ◆ Only listed senders will be communicated with
- ◆ **Secured Mode**



Security Attributes	Data Encryption	Frame Integrity
none	OFF	NO
MIC-32	OFF	YES
MIC-64	OFF	YES
MIC-128	OFF	YES
ENC	ON	NO
ENC-MIC-32	ON	YES
ENC-MIC-64	ON	YES
ENC-MIC-128	ON	YES

MIC=Message Integrity Code

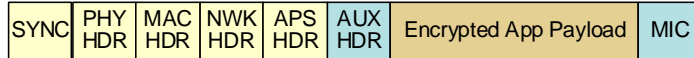
ENC=AES Encryption (CCM – Combined encryption and authentication block Cipher Mode)

Frames ...

3

## Frames with Security

### IEEE 802.15.4 Frames with Security



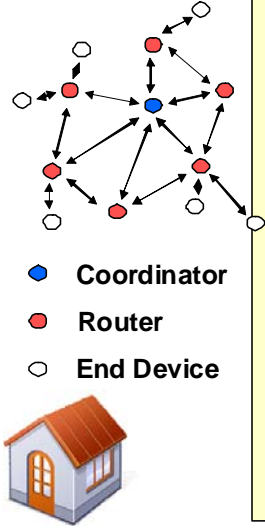
- ◆ Security adds Auxiliary Header and Message Integrity Code (MIC)
- ◆ Data in front of the MIC is integrity protected
- ◆ 27 byte overhead (Depending on security level selected)

Trust Center ...

4

# Trust Center

## Trust Center



◆ **Coordinator**  
● **Router**  
○ **End Device**

- ◆ Coordinator is assumed to be the Trust Center (TC) and provides:
  - Cryptographic key establishment
  - Key transport
  - Frame protection
  - Device management
- ◆ Two security modes
  - **Standard** – TC maintains a standard network key and controls network admittance. Memory required does not scale with network size (ex: Residential)
  - **High** – TC maintains a list of devices and keys. It provides network key updates. Memory required scales with network size. (ex: Commercial)

[Keys ...](#)

5

## Cryptographic Keys

### Cryptographic Keys

- ◆ **Master** – Basis for long term security  
Used for symmetric key establishment (1,3)
- ◆ **Link** – Shared exclusively between two network peers for Unicast communication (1,2,3)
- ◆ **Network** - Used for broadcast communication security (1,3)  
Alternate Network key employed for key rotation

Keys are obtained by:

1. Key-transport (key is sent from one device to another)
2. Key-establishment (key is derived locally)
3. Pre-installation (during programming)



Joining a Secure Network ...

6

# Joining a Secure Network

### Joining a Secure Network

**Secure Network**

**New Device**

- Coordinator
- Router
- End Device

- ◆ Keys need to be set with new devices that join the network

7

### Joining a Secure Network

**Secure Network**

**New Device**

- Coordinator
- Router
- End Device

- ◆ Keys need to be set with new devices that join the network
- ◆ Over the air key setup is unsecured and vulnerable to a one time eavesdropper attack

8

### Joining a Secure Network

**Secure Network**

● Coordinator  
● Router  
○ End Device

- ◆ Keys need to be set with new devices that join the network
- ◆ Over the air key setup is unsecured and vulnerable to a one time eavesdropper attack
- ◆ After joining, device may need to store multiple keys
- ◆ Solutions: out of band programming  
controlled joining  
device deactivation

Security features ...

9

## Features

### Security Features

- ◆ **Freshness**
  - ◆ Prevents replay attacks
  - ◆ Incoming/Outgoing freshness counters
- ◆ **Message Integrity**
  - ◆ Message cannot be modified in transit
  - ◆ 0, 32, 64 or 128-bit integrity
- ◆ **Authentication**
  - ◆ Assurance of Originator
  - ◆ Access Control
- ◆ **Confidentiality**
  - ◆ Prevents eavesdropping – 128-bit AES encryption
  - ◆ Can be turned off without affecting freshness, integrity or authentication

Lab time ...

10

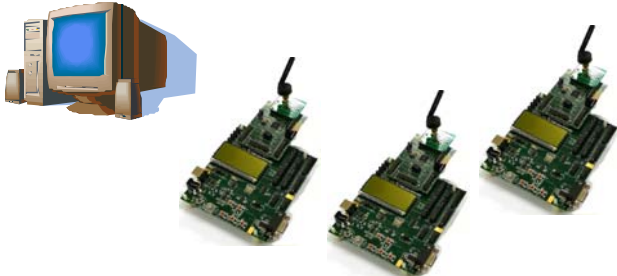

\*\*\* why can't we do this outside in the sunshine? \*\*\*

# Lab10- Security

**Lab 10 –ZigBee Security**

- ◆ **Transmit data securely**
- ◆ **Enable preconfigured keys**
- ◆ **Control network access**

	<u>Channel</u>	<u>PAN ID</u>
1	0x0C (12)	0x0AAA
2	0x0D (13)	0x0BEE
3	0x0E (14)	0x0CEE
4	0x0F (15)	0x0DEE
5	0x10 (16)	0x0EEE
6	0x11 (17)	0x0EFF
7	0x12 (18)	0x0BAB
8	0x13 (19)	0x0ACE

11

## Hardware list:

- ✓ 3 SmartRF05EB boards
- ✓ 3 CCMSP-EM430F2618 boards
- ✓ 3 CC2520EM boards
- ✓ 1 SmartRF04EB board
- ✓ 1 CC2430EM board
- ✓ 4 antennas
- ✓ MSP-FET430UIF and ribbon cable
- ✓ 5 USB A/B cables
- ✓ 2 AA batteries
- ✓ Post-it™ flags

## Software list:

- ✓ IAR Embedded Workbench for MSP430 version 4.11D
- ✓ TI Packet Sniffer version 2.10.1
- ✓ Daintree Sensor Network Analyzer version 2.3.0.8

(You will find shortcuts for the above applications on the desktop)

- ✓ Z-Stack version 2.1.0 – 1.3.0

## Procedure

### Initial setup

#### 1. Hardware

We'll be using the same setup as we did in module 8. Make sure they're all properly connected and labeled so you know which board has what purpose.

**NOTE:** If you use the Daintree SNA, be aware that it will capture the openly transmitted security key and decrypt the messages it displays. It will also store the last 5 successful keys, so if you try to fool the analyzer by starting it after the security key has been transmitted, the analyzer will still be successfully decrypting your messages based on previous keys. If you change the key and start the analyzer after the key has been transmitted, it will be unable to decrypt the messages. The ChipCon Packet Sniffer is unable to decrypt secure messages.

#### 2. Lab Software

Open the **GenericApp.eww** workspace from:

**C:\Texas Instruments\ZStack-2.1.0-1.3.0\Projects\zstack\Samples\Lab10\CC2520DB**

Modify the channel number and PAN ID as we've done before.

This lab software is the GenericApp project with a few minor changes:

- The "Hello World" message has been changed to "ABCDEFGH"
- Switch 1 (**joystick up**) and switch 3 (**joystick down**) have been programmed on the Router and Coordinator to control the **Permit Join** field. Switch 1 turns Permit Joining **ON**, while Switch 3 turns Permit Joining **OFF**. This will be useful later to ensure the End Device joins the Router instead of joining directly to the Coordinator.
- The message counter code has been added

### 3. Build, Download and Test without Security

Build and download Lab10 to all three boards, then power them all off. Security is not enabled in the code at this time, so all transmissions will be “in the clear”. Make sure that the **TI Packet Sniffer** is running and set to the correct channel.

Use the following procedure to start the boards:

1. In order, power the **Coordinator, Router** and **End Device**.
2. Press **SW4** (joystick left) on the **End Device** to initiate **automatch** binding. Like earlier, the Coordinator will respond and receive the first periodic transmission from the End Device, but the Router’s **match descriptor response** will be received by the **End Device** immediately afterwards (since it had to pass through the Coordinator). So the **Router** and the **End Device** will be bound to each other. Note that the message count increments on the Router.

Note the joining process and the content of over the air packets:

MAC payload															
48	00	01	00	6F	79	1D	23	00	0A	01	00				
04	0F	0A	09	41	42	43	44	45	46	47	00				

**41 42 43 44 45 46 47** are the letters **ABCDEFG** in hex.

## Turn Security On

### 4. Turn Security Key Exchange On

Open **f8wConfig.cfg** in the **Tools workspace folder**. At the top of the file set **-DSECURE=1**.

Open **ZGlobals.c** in the **NWK workspace group** and note that **gPreConfigKeys = FALSE;**

This parameter configures the system so that only the **Coordinator** stores the preconfigured key. During the joining process the key is passed (once per join) in the **clear**. Be aware of the security implications of passing the key in the clear (an out-of-band transfer is the preferred method).

When **gPreConfigKeys = TRUE;**, all devices in the network must be preconfigured at build time with the security key.

Note the setting of **defaultTCLinkKey** in **nwk\_globals.c**. in the **NWK** group. Replace the **Key for In-House Testing** with the following:

```
// Key for In-House Testing
0x54, 0x45, 0x58, 0x41, 0x53, 0x49, 0x4E, 0x53,
0x54, 0x52, 0x55, 0x4D, 0x45, 0x4E, 0x54, 0x53
```

You can cut/paste this from **Lab10\_Code.txt**. By the way, the key is **TEXASINSTRUMENTS** in hex.

You should also be aware that **nwk\_globals.c**, like **f8wconfig.cfg**, is a component file to the project rather than a source file. There is only a single copy of this file for all the project workspaces we've used.

### 5. Build, Download and Test

**Build** and **download** the project to all three devices, now with Key Exchange security enabled. Now using the **Daintree SNA**, clear and start the sniffer and observe the packets.

Turn on the **Coordinator**, then the **Router**. Notice the joining process is now extended with a few additional “handshakes”. Here’s a screen shot that I took from this process:

Se...	Channel	T.	Time Delta	MAC Src	MAC Dest	NWK Src	NWK Dest	Protocol	Packet Type	Security	FCS
7	11	..	+00:00:00.004	0x0000				IEEE 802.15.4	Beacon: BO: 15, SO: 15, PC: 1,..		✓
8	11	..	+00:00:00.507	10:d6:b..	0x0000			IEEE 802.15.4	Command: Association Request		✓
9	11	..	+00:00:00.001					IEEE 802.15.4	Acknowledgment		✓
10	11	..	+00:00:00.493	10:d6:b..	0x0000			IEEE 802.15.4	Command: Data Request		✓
11	11	..	+00:00:00.001					IEEE 802.15.4	Acknowledgment		✓
12	11	..	+00:00:00.002	20:f5:6..	10:d6:b4..			IEEE 802.15.4	Command: Association Response		✓
13	11	..	+00:00:00.001					IEEE 802.15.4	Acknowledgment		✓
14	11	..	+00:00:00.107	0x0000	0x0001	0x0000	0x0001	Zigbee APS	APS Command: Transport Key		✓
15	11	..	+00:00:00.002					IEEE 802.15.4	Acknowledgment		✓
16	11	..	+00:00:00.017	0x0001	0xffff	0x0001	0xffffd	Zigbee APS Data	ZDP:EndDeviceAnnce	NWK	✓
17	11	..	+00:00:00.021	0x0000	0xffff	0x0001	0xffffd	Zigbee APS Data	ZDP:EndDeviceAnnce	NWK	✓

Somewhere about message number **14**, you’ll see the **APS Command: Transport Key** in the **Packet List** window (mine was a lovely shade of pink). Double-click on the message to open it in the **Packet Decode** window. Expand out the message contents:

```

Frame 14 (Length = 56 bytes)
+ IEEE 802.15.4
+ ZigBee NWK
- ZigBee APS
  - Frame Control: 0x01
  - Counter: 0x00
  - APS Payload
    - APS Command Identifier = Transport Key: (0x05)
    - KeyTransport.CommandPayload
      - Key Type: Network Key (0x01)
      - Key: 53:54:4e:45:4d:55:52:54:53:4e:49:53:41:58:45:54
      - Sequence Number: 0
      - Destination Address: 10:d6:b4:7d:6b:7f:73:6c
      - Source Address: 20:f5:68:5a:23:21:41:61
  
```

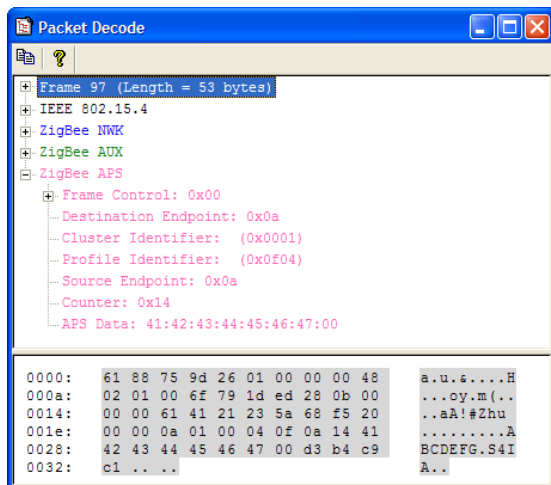
0000:	61 88 67 9d 26 01 00 00 00 48	a.g.s...H
000a:	00 01 00 00 00 1e a2 01 00 05	....."
0014:	01 54 45 58 41 53 49 4e 53 54	.TEXASINST
001e:	52 55 4d 45 4e 54 53 00 6c 73	RUMENTIS.1s
0028:	7f 6b 7d b4 d6 10 61 41 21 23	.k}4V.aA!#
0032:	5a 68 f5 20 .. ..	Zhu ..

Note that this is the message that transports the preconfigured key, in the clear, to the Router. Turn on the **End Device** and watch the same process occur between it and the **Coordinator**. In both cases, note the addresses of the source and destination.

## 6. Secure Messages

Power on the **End Device** and press **SW4** (joystick left) to enable **automatch** binding. Watch the LCD screen on the Router as the “**ABCDEF**” messages are sent every 5 seconds.

In the **Daintree SNA**, look for a **ZigBee APS Data** message. Use the **Packet Decoder** to look at the contents:



The Daintree SNA applies the security key that it detected to decode subsequent messages, but we can disable that capability.

## 7. Restart Network

Shut all the network boards down and **stop** the **Daintree SNA**. On the Daintree SNA menu bar, click **Settings → Options... → Security tab**. Click the **Disable automatic detection of keys** checkbox, then click **OK**. Start capturing packets.

Power on the **Coordinator**, **Router** and **End Device**. Press **SW4** (Joystick left) on the **End Device**.

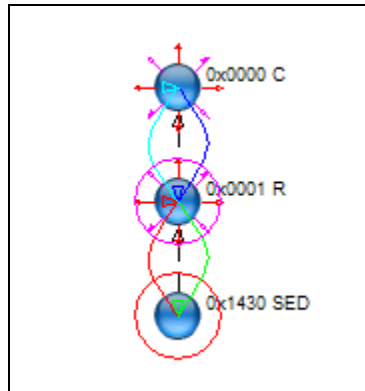
Now look for **ZigBee NWK** messages and note that these messages are secured ... the SNA cannot decrypt the payloads.

### 8. Transmission of key over Multiple Hops

Shut all the network boards down and **restart** the **Daintree SNA**. Power on the **Coordinator** first, then the **Router** – but **DO NOT** turn the **End Device** on yet.

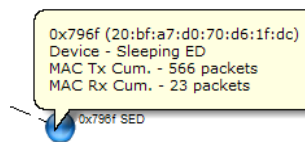
Press **SW3** (joystick **down**) on the **Coordinator** to turn **Permit Join OFF** (forcing the End Device to join directly to the Router). Power on the **End Device** on and watch the association with the **Router** on the **Packet Sniffer**. Analyze the Packet Sniffer output and identify the difference in the full joining process. Watch for the handshake that goes all the way back to the Coordinator the get the key. Note that only the last hop (to the End Device) is in the clear.

Since the **End Device** is now the first **ED** on the **Router**, its address should be **0x1430**. You'll also note that the Visual Device Tree looks different than before. By the way, the circles around the Router and End Device indicate secured devices.



### 9. Write down the IEEE address

In a later lab step, we're going to need the IEEE address of the **End Device**. In the **Daintree SNA Visual Device Tree**, **hover** over the **End Device** (SED) and you should see a pop-up window like the one below. The IEEE number is within the parentheses. Write yours in the space provided below:



IEEE#								
Byte	1	2	3	4	5	6	7	8

### 10. Optional step: Preconfigured Keys

**If time permits**, we can change the process to use preconfigured keys on all devices. The only change you need to make is to set **gPreConfigKeys = TRUE** in **ZGlobals.c**. Build and download to all the devices and test the changes. In the Packet Sniffer you should now note the absence of the key transmission in the clear.

In this case all devices are expected to be preconfigured with security keys, so the application packet that follows the association is encrypted with the key, as is all subsequent communication. This reduces the security implications of transmitting the key in the clear, but certainly increases the production and maintenance issues.

## Network Access Control

### 11. Turn Preconfigured Keys Off

In a secure network, the **Coordinator** is informed when a device joins the network. The **Coordinator** therefore has the option of allowing that device to remain on the network or deny network access immediately after joining. By default we provide an example where the security manager can maintain a list of restricted devices which it does not allow on the network.

Before moving forward, assure that `gPreConfigKeys = FALSE` in `ZGlobals.c` so that key exchange is performed.

### 12. Configure Network Access Control

Find the `ZDSecMgrDeviceValidate()` function (not `SKKE`, `RM` or `CM`) in `ZDSecMgr.c` in the `ZDO` workspace folder. Note that here is where the decision is made to either run **high** security mode (`ZDSecMgrDeviceValidateCM()`) or **normal** security mode (`ZDSecMgrDeviceValidateRM()`).

Since we're not running the high security mode, find the `ZDSecMgrDeviceValidateRM()` function. The `zgSecurityMode` variable (set in `ZGlobals.c`) can force the Coordinator to reject any newly joining device. Right now that variable is set to **true**, and that's not really what we want to do here. So let's enable the white/black list feature instead:

**Comment out** the `#if 0` and the `#endif` that surround the `ZDSecMgrDeviceValidateRM()` code.

You can probably see for yourself that it would be a pretty trivial exercise to change this code from a black-list to a white-list. Right now, if the joining device address matches the list, it is disallowed (black). You could quickly change this to an allowed-device list (white). You could also alter the address comparison to only compare a portion of the IEEE address ... comparing the part of the address that all your devices have in common.

Also note that the `ZDSecMgr.c` file, like the `fw8Config.cfg` file is a **Component** file. They are not in the project folder, but in a separate Components folder common to all of your projects. Oh yeah, that can bite you for sure.

### 13. Setting the Black IEEE Addresses

Find the list of IEEE addresses at `ZDSecMgrStoredDeviceList`. **Comment out** the `#if 0` and the `#endif` that surround the array. There are three example IEEE addresses in the list, each one broken into 8 groups of 8-bit addresses (64-bits total).

Unfortunately the comparison function is **backwards**, so you'll need to enter the IEEE address you recorded in lab step 9 **backwards** (from byte 8 to 1). In the example below, my End Device's IEEE address was **20:bf:a7:d0:70:d6:1f:dc**.

```

///if 0 // Taken out because the following functionality is only used for test
// purpose. A more efficient (above) way is used. It can be put
// back in if customers request for a white/black list feature.
uint8 ZDSecMgrStoredDeviceList[ZDSECMGR_STORED_DEVICES][Z_EXTADDR_LEN] =
{
  { 0xdc, 0x1f, 0xd6, 0x70, 0xd0, 0xa7, 0xbf, 0x20 },
  { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01 },
  { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },
};
///endif

```

### 14. Build, Download and Test

Rebuild and download the **Coordinator** project only. Start your Sniffer, then power the **Coordinator** and the **Router**. The Router will receive the preconfigured security key from the Coordinator normally. Watching the Sniffer display carefully, power up the End Device and verify that the device is not allowed to join the network.

Looking at the **Packet Sniffer** display you will notice that the **End Device** will accomplish an association, but will never receive a packet from the **Coordinator** providing the network key. As a result the device fails to participate in the network, even though the **End Device** application code (**ZDO**) continues attempting to rejoin (this can be turned off as desired by modifying the **ZDO**).

### 15. Power down

Close all open windows on your desktop and power off all the boards.



You're done