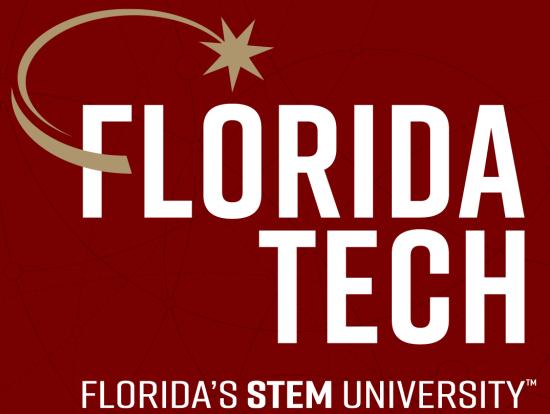


Department of Computer Science



CSE 4820: Wireless and Mobile Security

1. Preliminaries

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Background: Computer Networks

Background: Cybersecurity Basics

Computer Networks

Connects two or more computing devices

- Computers, phones, IoT

Various protocols between different device set

- Protocol define the rules of how they interact

Example daily uses:

- Virtual classrooms, messaging, emails, social media posts, etc.



Protocols

- “The official procedure or system of rules governing affairs of state or diplomatic occasions”

PROTOCOL = Set of rules to communicate.

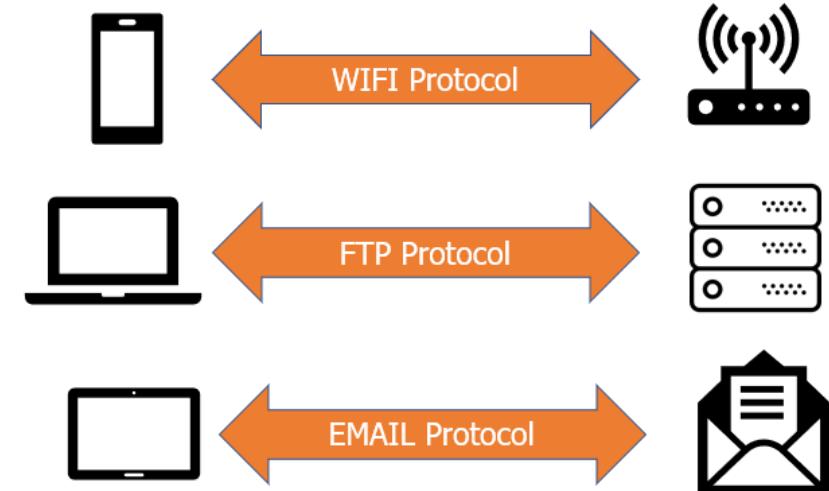


- A communication protocol:
 - System of rules that allow two or more entities of a communications system to transmit information via any kind of variation of a physical quantity
 - Defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods

Communication Protocols

- Each protocol defines different set of:
 - Format of messages
 - Order
 - Actions
 - Security?
- Protocol runs on multiple nodes, and implements certain functionality of a single layer
 - Works through packet header

PROTOCOL = Set of rules to communicate.



How to Design Network Protocols

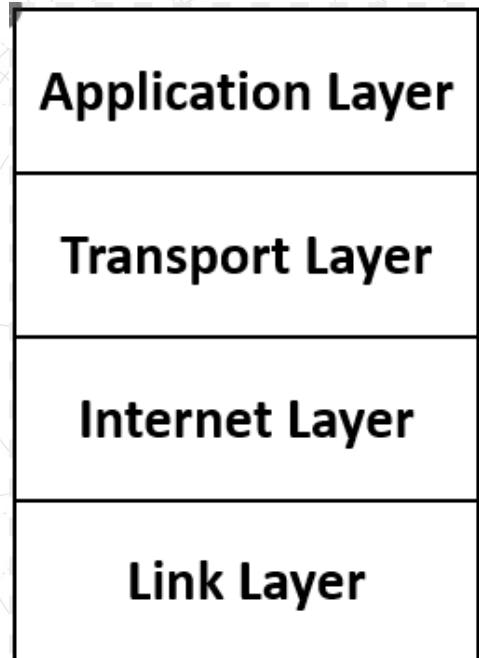
- Internet has billions of devices
- How to manage the different devices to talk to each other
- How to deal with maintenance, scalability, accountability of them?
- Solution: Divide and control

Layering

- A way of abstracting and organizing functionality
 - Without specifying implementation details
- Eases maintenance, updating of system
- Provides scalability
- => Leads to design protocol stack by creating different layers for different tasks

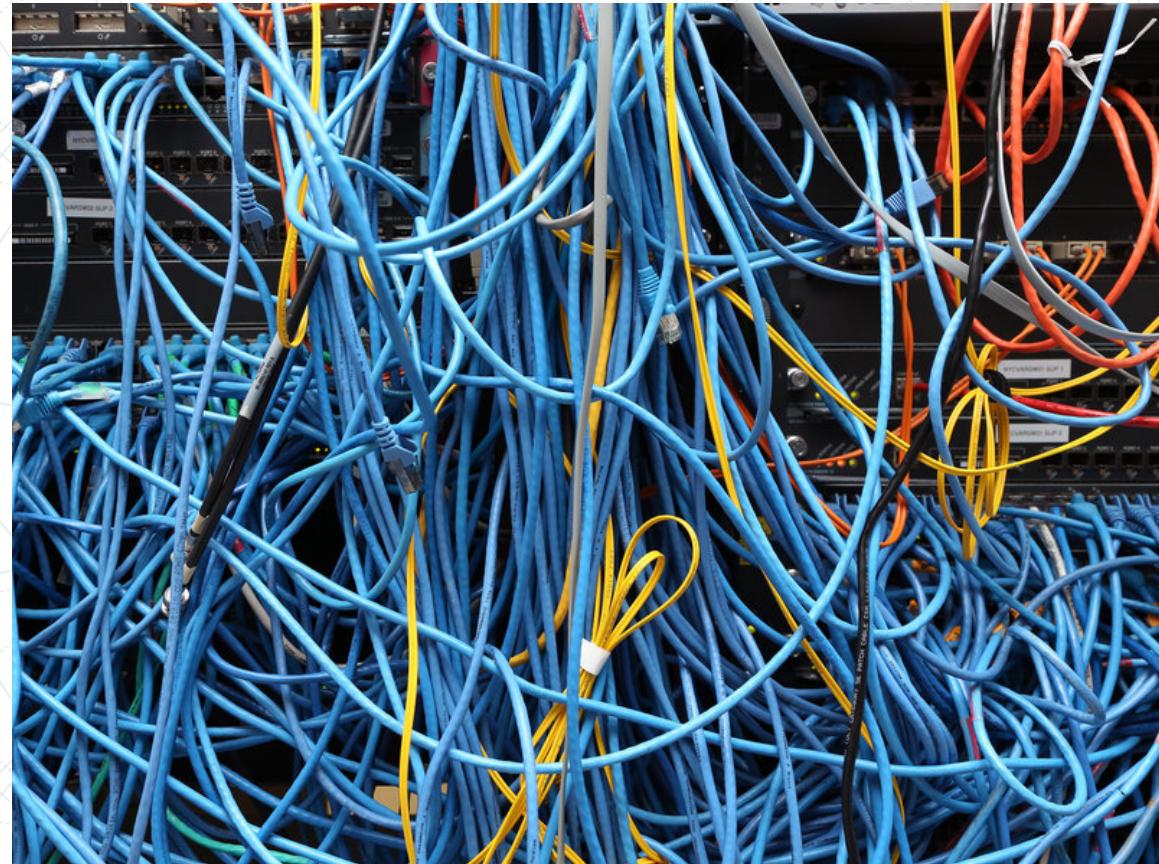
Network Protocols

- **Application Layers:** End-user applications
- **Transport Layer:** Data transfer from end to end
- **Internet (Network) Layer:** Routing of data from source to destination
- **Link (Physical) Layer:** Physical media carrying the data



Link Layer

- Link = Medium + Adapters
- Enable host-to-host communication within a single local area network (LAN)



Some “Link” Examples

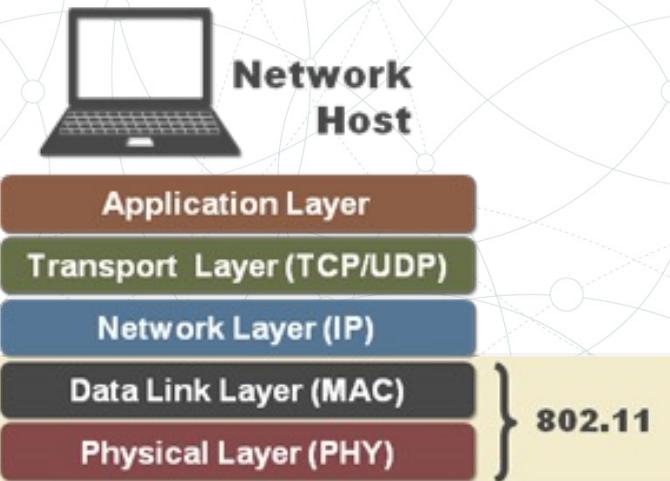
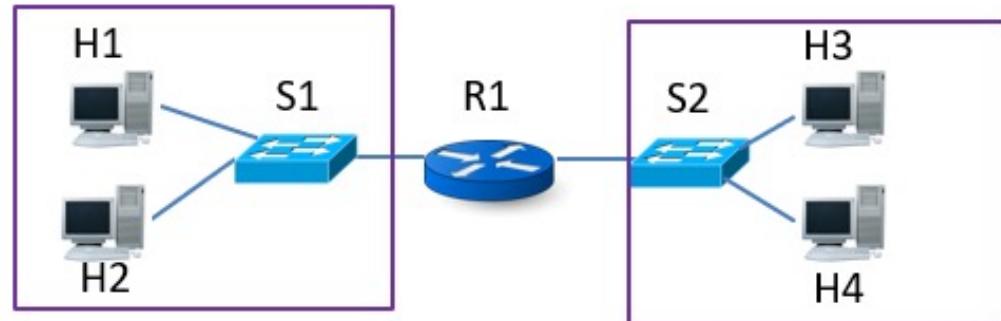


Some “Network Adapters”



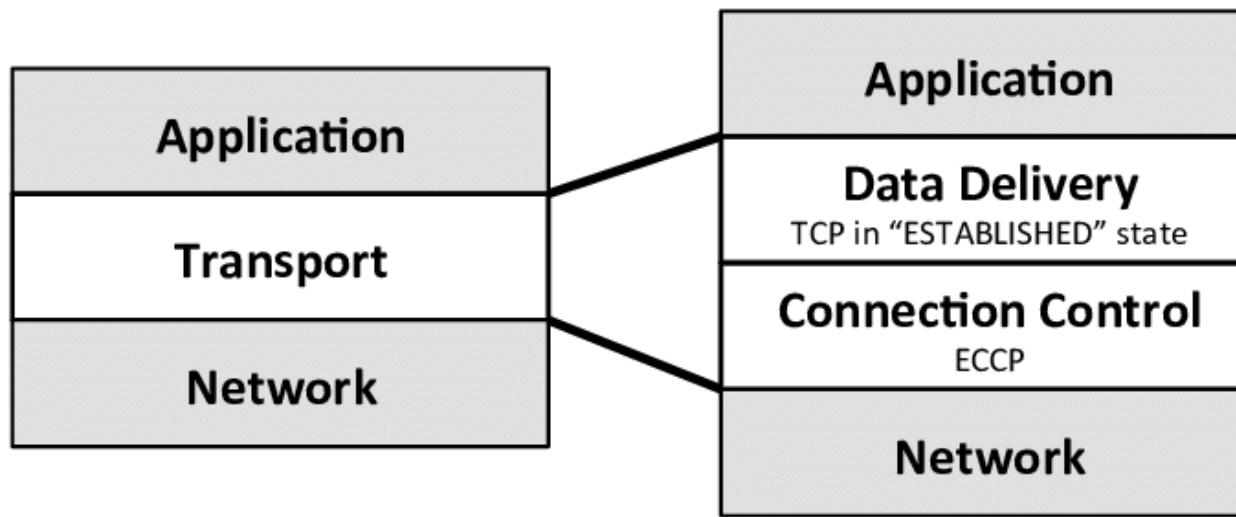
Network Layer

- Connecting networks
 - Forwarding: move from one port to another
 - Routing: calculate the route it should take to arrive destination
- Internet Protocol (IP): IP addresses
 - IPv4, e.g. 157.23.54.201
 - IPv6, e.g. 3002:0db8:85a3:ffff:0000:8a2e:0370:7114



Transport Layer

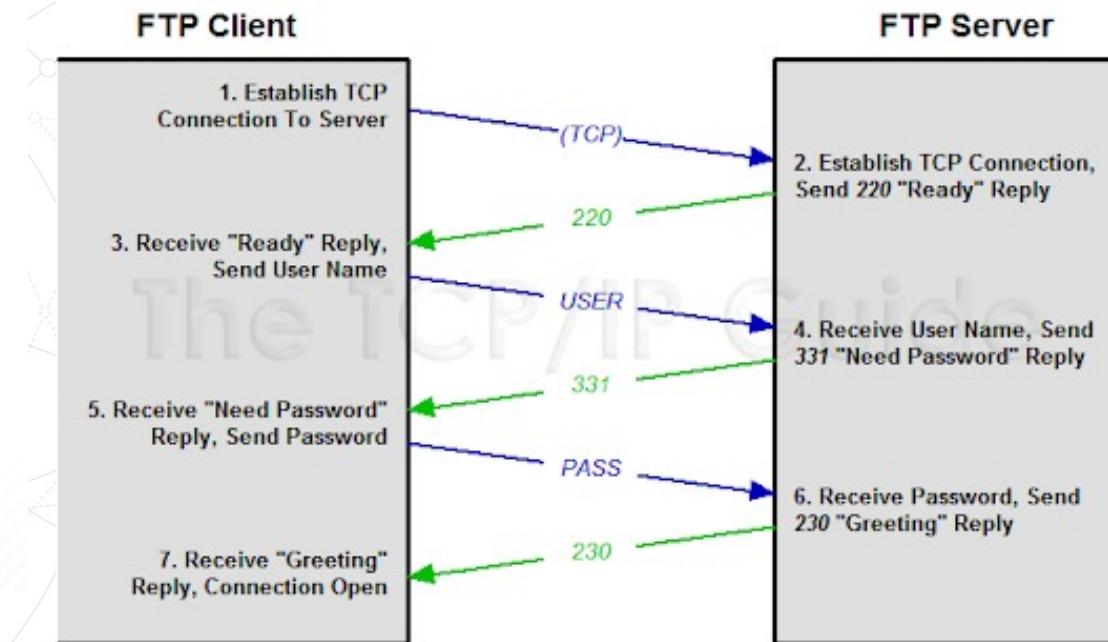
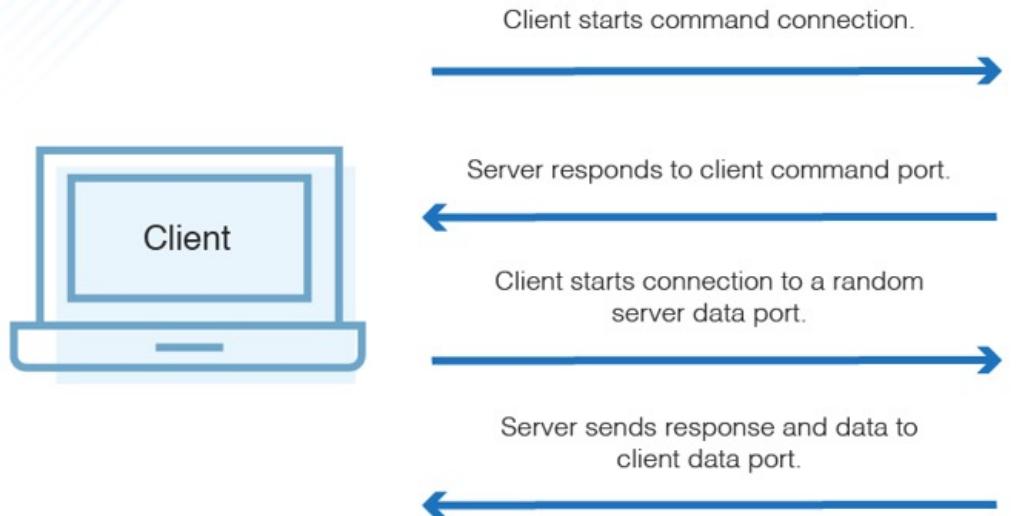
- Layer 1 & 2 deal with forwarding packets from one place to another
 - Mechanisms for finding paths, locating destination, etc.
- Layer 3 provides two extra functionality on top of forwarding



Application Layer

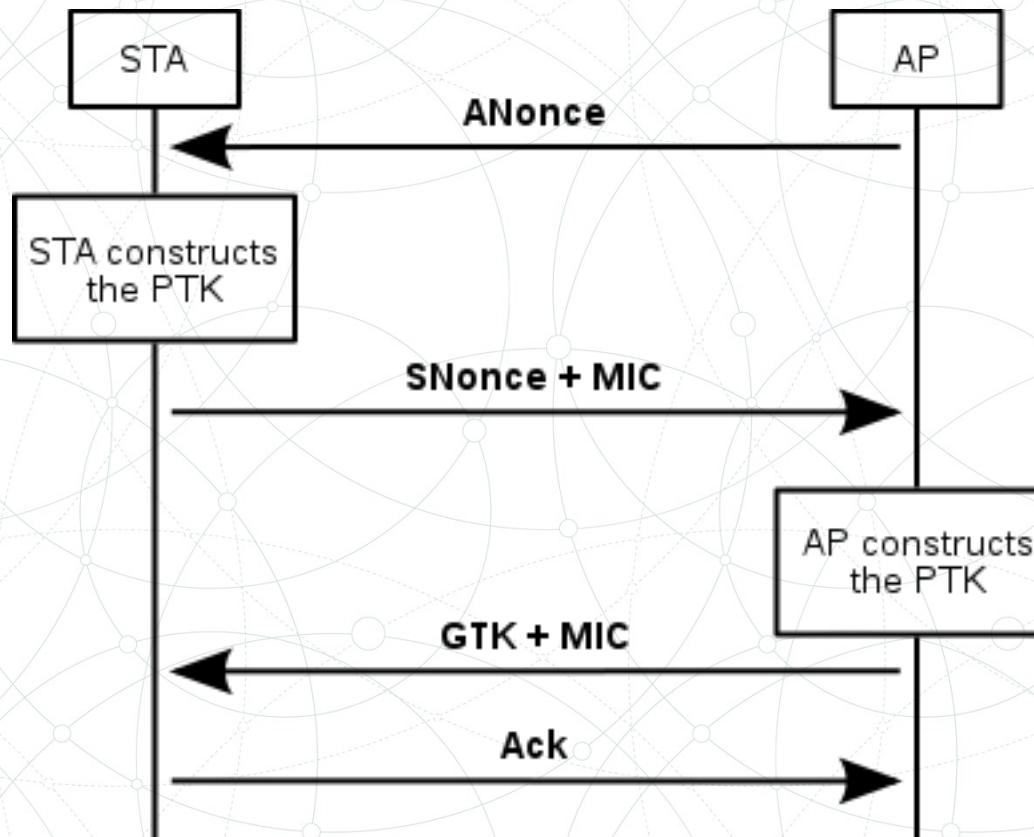
- It defines:
 - types of messages exchanged,
 - request, response
 - message syntax:
 - what fields in messages & how fields are delineated
 - message semantics
 - meaning of information in fields
 - rules for when and how processes send & respond to messages
- Open protocols:
 - defined in RFCs
 - allows for interoperability
 - e.g., HTTP
- Proprietary protocols:
 - e.g., Zoom, Skype

File Transfer Protocol (FTP) Steps



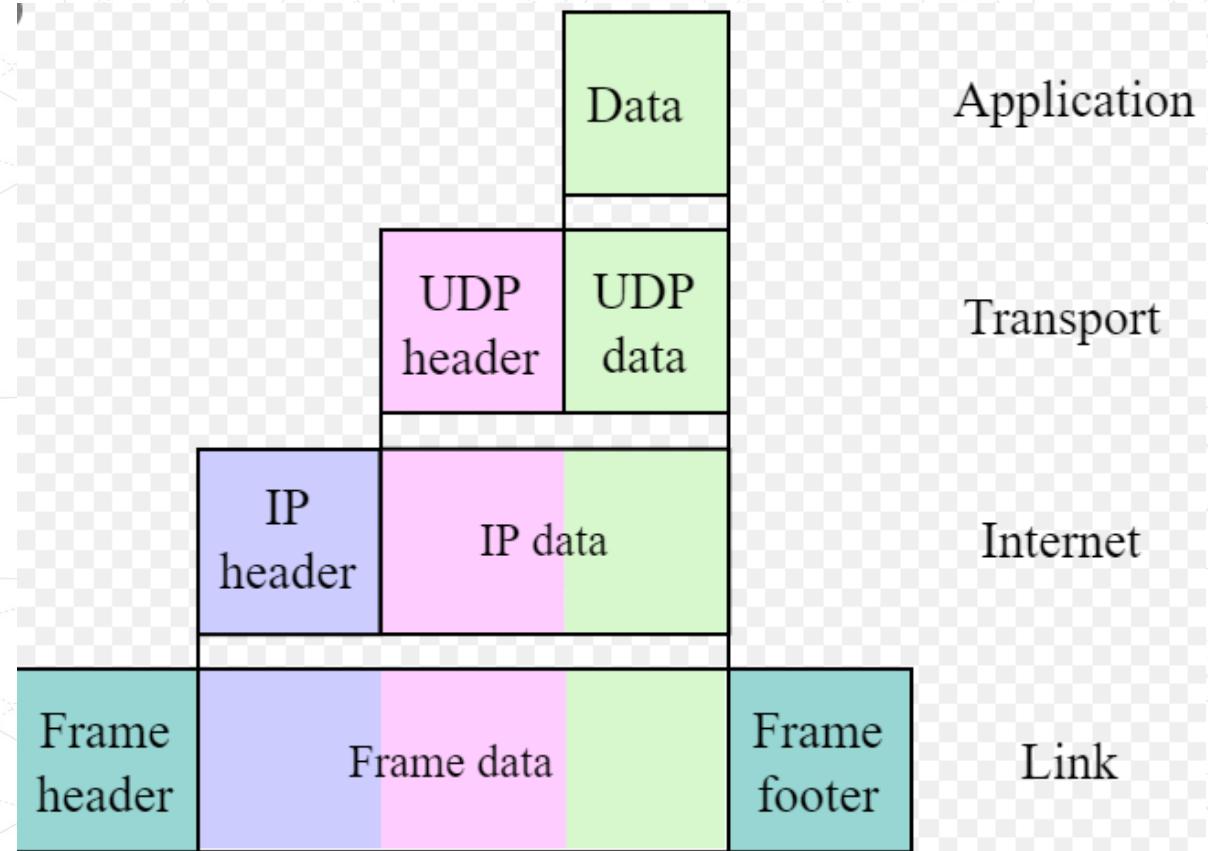
<https://datatracker.ietf.org/doc/html/rfc959>

WiFi (IEEE 802.11)



Data Communication via Networks

- Application data; user input
- Transport layer adds its header
 - TCP or UDP
- Internet layer adds
 - IP header (or ICMP)
- Link Layer adds extra info
 - Such as error correction, etc.



Data in Layers

- Full data communicated:

```
Frame 5438: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface 0
Interface id: 0 (\Device\NPF_{[REDACTED]-2B8C-40B9-BA81-[REDACTED]})  

Encapsulation type: Ethernet (1)  

Arrival Time: Aug 17, 2021 21:35:53.945308000 Central Daylight Time  

[Time shift for this packet: 0.000000000 seconds]  

Epoch Time: 1629254153.945308000 seconds  

[Time delta from previous captured frame: 0.004450000 seconds]  

[Time delta from previous displayed frame: 0.004450000 seconds]  

[Time since reference or first frame: 38.141095000 seconds]  

Frame Number: 5438  

Frame Length: 94 bytes (752 bits)  

Capture Length: 94 bytes (752 bits)  

[Frame is marked: False]  

[Frame is ignored: False]  

[Protocols in frame: eth:ethertype:ip:tcp:ssl]  

[Coloring Rule Name: TCP]  

[Coloring Rule String: tcp]
```

Data in Layers

- Data in physical layer (or MAC layer):

```
Ethernet II, Src: D-LinkIn_e9:[REDACTED] (c0:a0:bb:e9:[REDACTED]), Dst: 06:39:a5:[REDACTED] (06:39:a5:[REDACTED])
  > Destination: 06:39:a5:[REDACTED] (06:39:a5:[REDACTED])
  > Source: D-LinkIn_e9:91:de (c0:a0:bb:e9:[REDACTED])
  Type: IPv4 (0x0800)
```

- Data in network layer

```
Internet Protocol Version 4, Src: 157.254.[REDACTED], Dst: 192.168.[REDACTED]
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 80
  Identification: 0x6214 (25108)
  > Flags: 0x02 (Don't Fragment)
  Fragment offset: 0
  Time to live: 88
  Protocol: TCP (6)
  Header checksum: 0x5f88 [validation disabled]
  [Header checksum status: Unverified]
  Source: 157.254.[REDACTED]
  Destination: 192.168.[REDACTED]
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
```

Data in Layers

- Data in transmission layer:
- Data in application layer (encrypted):

Transmission Control Protocol, Src Port: 443, Dst Port: 53226, Seq: 1815, Ack: 5217, Len: 40

Source Port: 443
Destination Port: 53226
[Stream index: 20]
[TCP Segment Len: 40]
Sequence number: 1815 (relative sequence number)
[Next sequence number: 1855 (relative sequence number)]
Acknowledgment number: 5217 (relative ack number)
Header Length: 20 bytes
Flags: 0x018 (PSH, ACK)
Window size value: 369
[Calculated window size: 94464]
[Window size scaling factor: 256]
Checksum: 0xe2c0 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[SEQ/ACK analysis]

Secure Sockets Layer

▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
Content Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 35
Encrypted Application Data: 40264ebace1718dcd3a9513764ed6d2664d01ce2118a54d3...

Wifi Packet Example

24 1.567097

Apple_98:f0:6f... 802... 39 Acknowledgement, Flags=.....C

- › Frame 24: 39 bytes on wire (312 bits), 39 bytes captured (312 bits) on interface unknown, id 0
- › Radiotap Header v0, Length 25
- › 802.11 radio information
- › IEEE 802.11 Acknowledgement, Flags:C

Frame 24: 39 bytes on wire (312 bits), 39 bytes captured (312 bits) on interface unknown, id 0

Interface id: 0 (unknown)
Encapsulation type: IEEE 802.11 plus radiotap radio header (23)
Arrival Time: Jul 10, 2012 00:12:59.887086000 EDT
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1341893579.887086000 seconds
[Time delta from previous captured frame: 0.000005000 seconds]
[Time delta from previous displayed frame: 0.000005000 seconds]
[Time since reference or first frame: 1.567097000 seconds]
Frame Number: 24
Frame Length: 39 bytes (312 bits)
Capture Length: 39 bytes (312 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: radiotap:wlan_radio:wlan]

Wifi Packet Example

Radiotap Header v0, Length 25

Header revision: 0
Header pad: 0
Header length: 25
> Present flags
MAC timestamp: 2443214761
> Flags: 0x12
Data Rate: 24.0 Mb/s
Channel frequency: 2462 [BG 11]
> Channel flags: 0x0480, 2 GHz spectrum, Dynamic CCK-OFDM
Antenna signal: -55 dBm
Antenna noise: -91 dBm
Antenna: 0

802.11 radio information

PHY type: 802.11g (ERP) (6)
Proprietary mode: None (0)
Data rate: 24.0 Mb/s
Channel: 11
Frequency: 2462MHz
Signal strength (dBm): -55 dBm
Noise level (dBm): -91 dBm
Signal/noise ratio (dB): 36 dB
TSF timestamp: 2443214761
> [Duration: 28 μ s]

IEEE 802.11 Acknowledgement, Flags:C

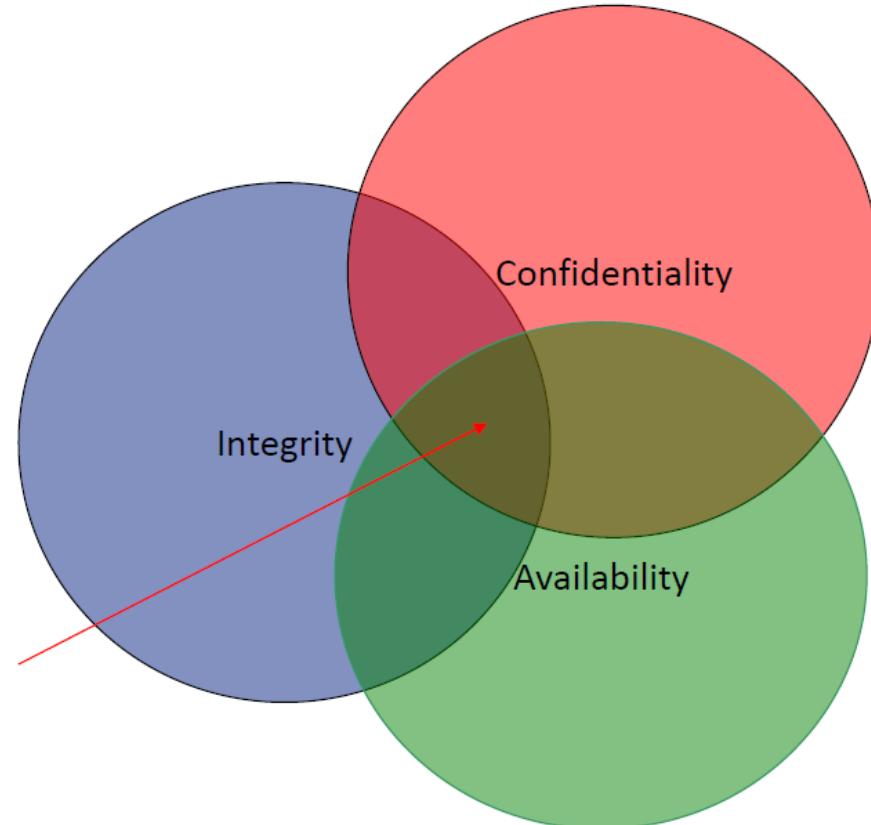
Type/Subtype: Acknowledgement (0x001d)
> Frame Control Field: 0xd400
.000 0000 0000 = Duration: 0 microseconds
Receiver address: Apple_98:f0:6f (00:17:f2:98:f0:6f)
Frame check sequence: 0x716ea6c7 [unverified]
[FCS Status: Unverified]

Background: Cybersecurity Basics

Cybersecurity goals

CIA Triad:

- Confidentiality
- Integrity
- Availability



Confidentiality

Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information

Data Confidentiality:

- Private or confidential information is not revealed to unauthorized individuals

Privacy:

- Users control what information about them can be
 - Collected
 - Stored
 - By whom

Integrity

Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity

Data Integrity

- Information and programs are changed only in specified and authorized manner

System Integrity

- System performs intended function free from unauthorized system manipulation

Availability

- Ensuring timely and reliable access to and use of information
- Actions by an attacker do not prevent users from having access to use of the system
 - Enable access to data and resources
 - Timely response
 - Fair resource allocation

Type of Cyberattacks: Attacker's Location

- Insider vs. Outsider Attacks:
 - An inside attack is an attack initiated by an entity inside the security perimeter (an "insider"),
 - An entity that is authorized to access system resources but uses them in a way not approved by those who granted the authorization
 - An outside attack is initiated from outside the perimeter, by an unauthorized or illegitimate user of the system (an "outsider")

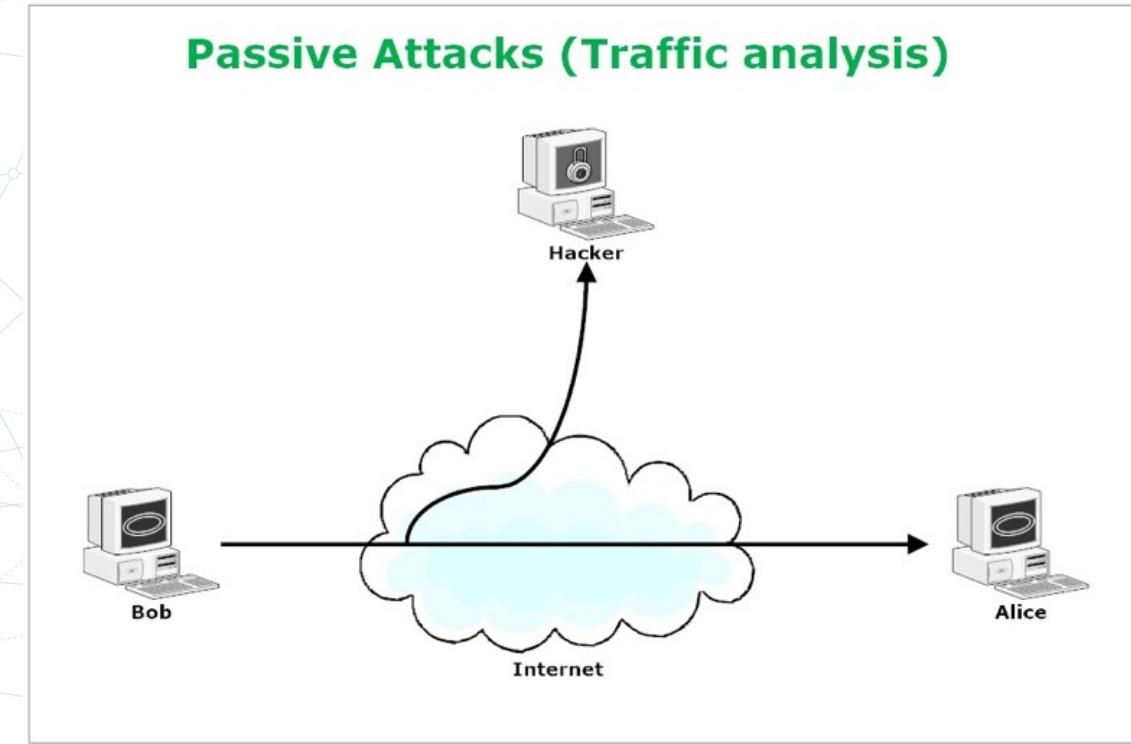


Type of Cyberattacks: Attacker's Behavior

- Active vs. Passive Attacks:
 - An active attack attempts to alter system resources or affect their operation
 - Involve some modification of the data stream or the creation of a false stream
 - A passive attack attempts to learn or make use of information from the system but does not affect system resources
 - Eavesdropping, monitoring, etc.

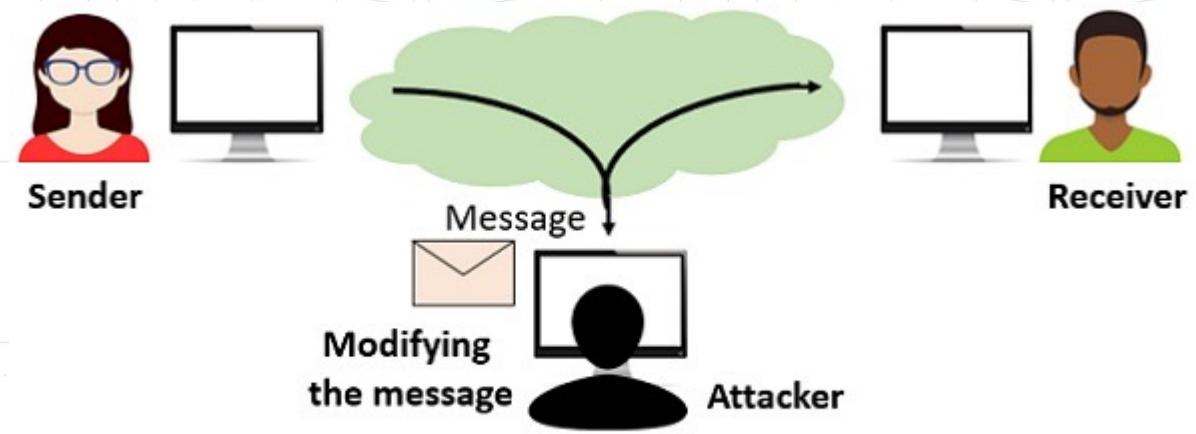
Passive Attacks

- Computer and network surveillance
- Network
 - Wiretapping
 - Fiber tapping
 - Port scan
- Host
 - Keystroke logging
 - Backdoor



Active Attacks

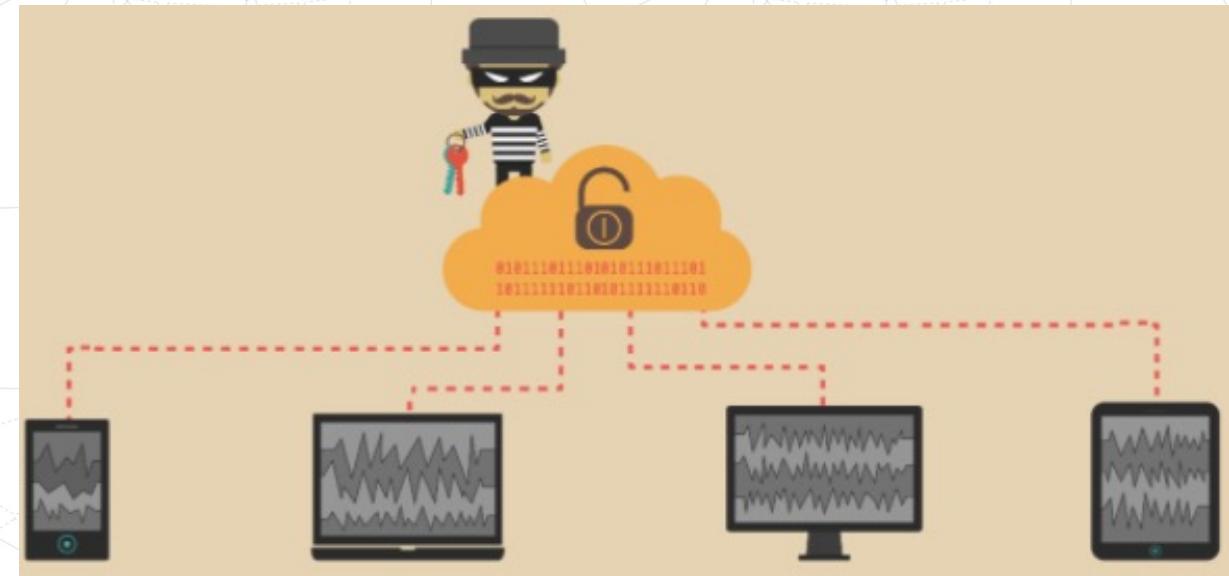
- Denial-of-service (DoS) attack
 - Distributed Denial of service (DDoS) attack
- Spoofing
- Network based:
 - Man-in-the-middle
- Host based:
 - Buffer overflow
 - Format string attack



Active Attack

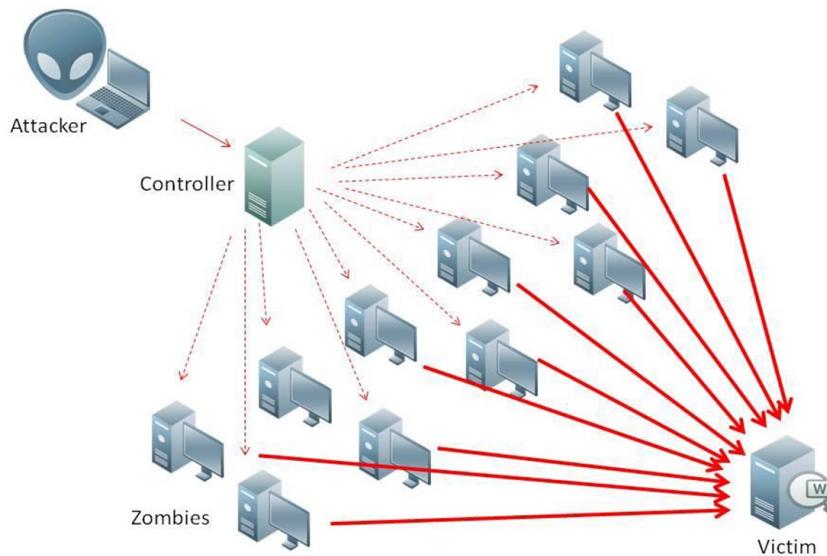
Some Common Attack Types

- Denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks
- Man-in-the-middle (MitM) attack
- Phishing and spear phishing attacks
- Malware attack



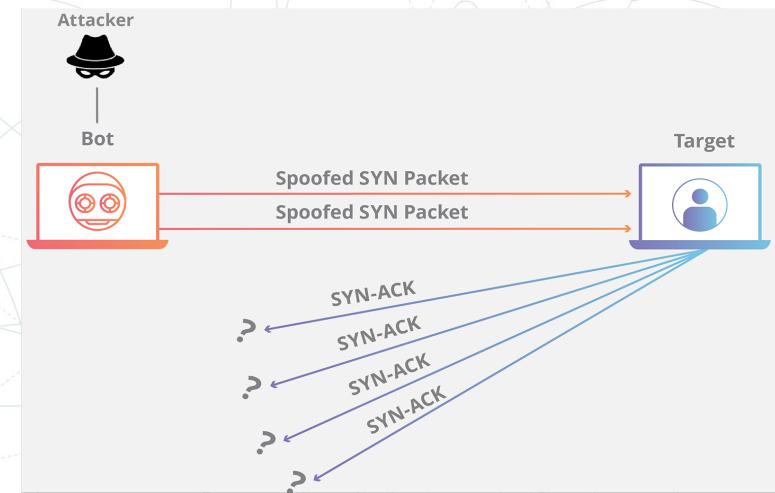
DoS and DDoS

- DoS attack overwhelms a system's resources so that it cannot respond to service requests
- DDoS attack is also an attack on system's resources, but it is launched from a large number of other host machines that are infected by malicious software controlled by the attacker
 - TCP SYN flood attack, etc.



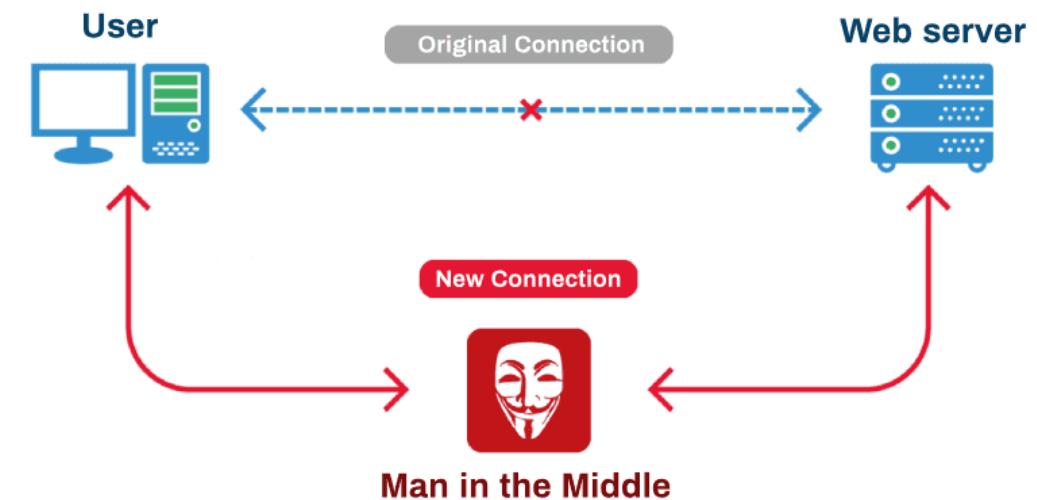
TCP SYN flood attack

- Exploits the use of the buffer space during a Transmission Control Protocol (TCP) session initialization handshake
- The attacker's device floods the target system's small in-process queue with connection requests, but it does not respond when the target system replies to those requests
 - Causes the target system to time out while waiting for the response from the attacker's device
 - Makes the system crash or become unusable when the connection queue fills up

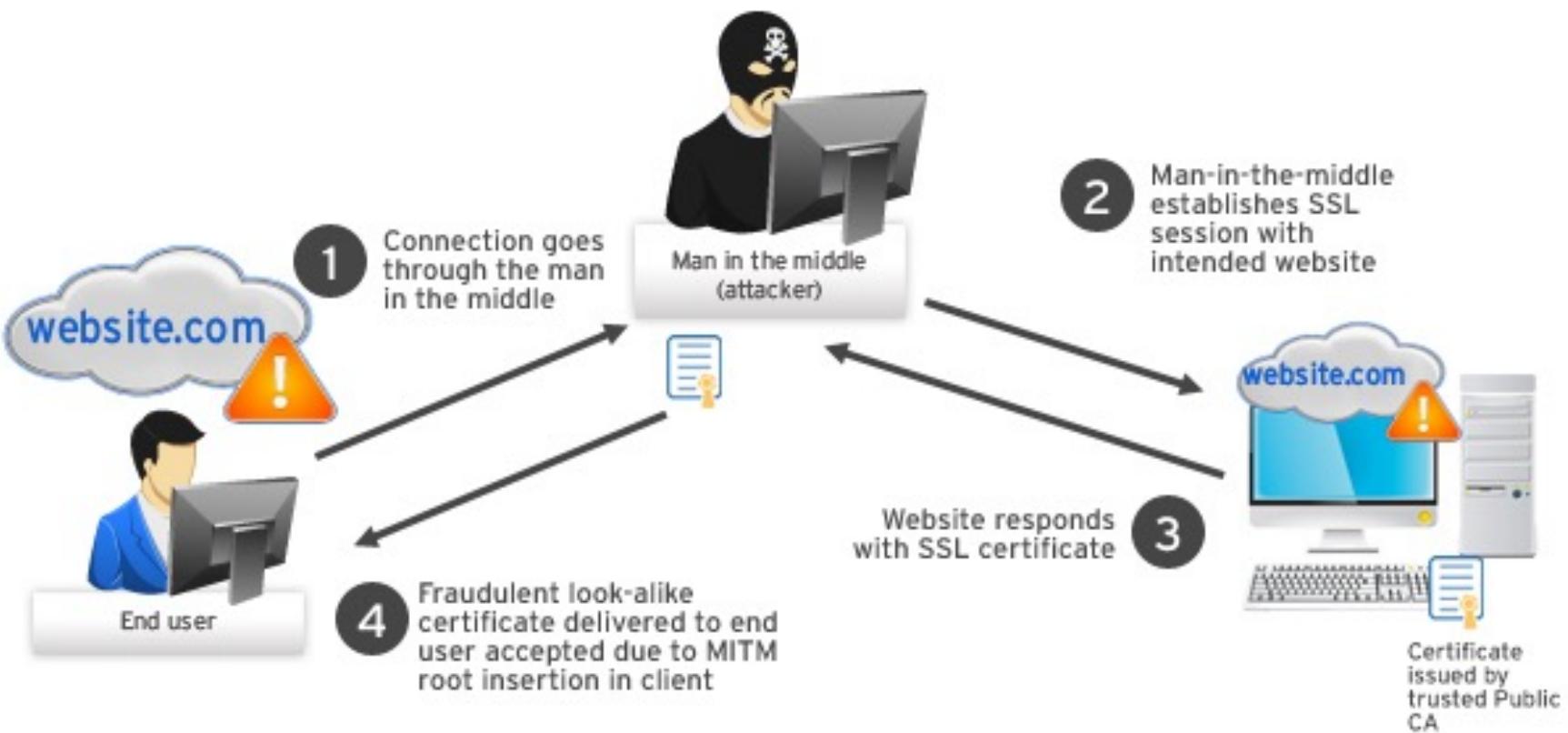


Man-in-the-middle (MitM) attack

- A MitM attack occurs when a hacker inserts itself between the communications of a client and a server
- Some common MitM attacks:
 - Session hijacking
 - IP Spoofing
 - Replay



MitM: How it works

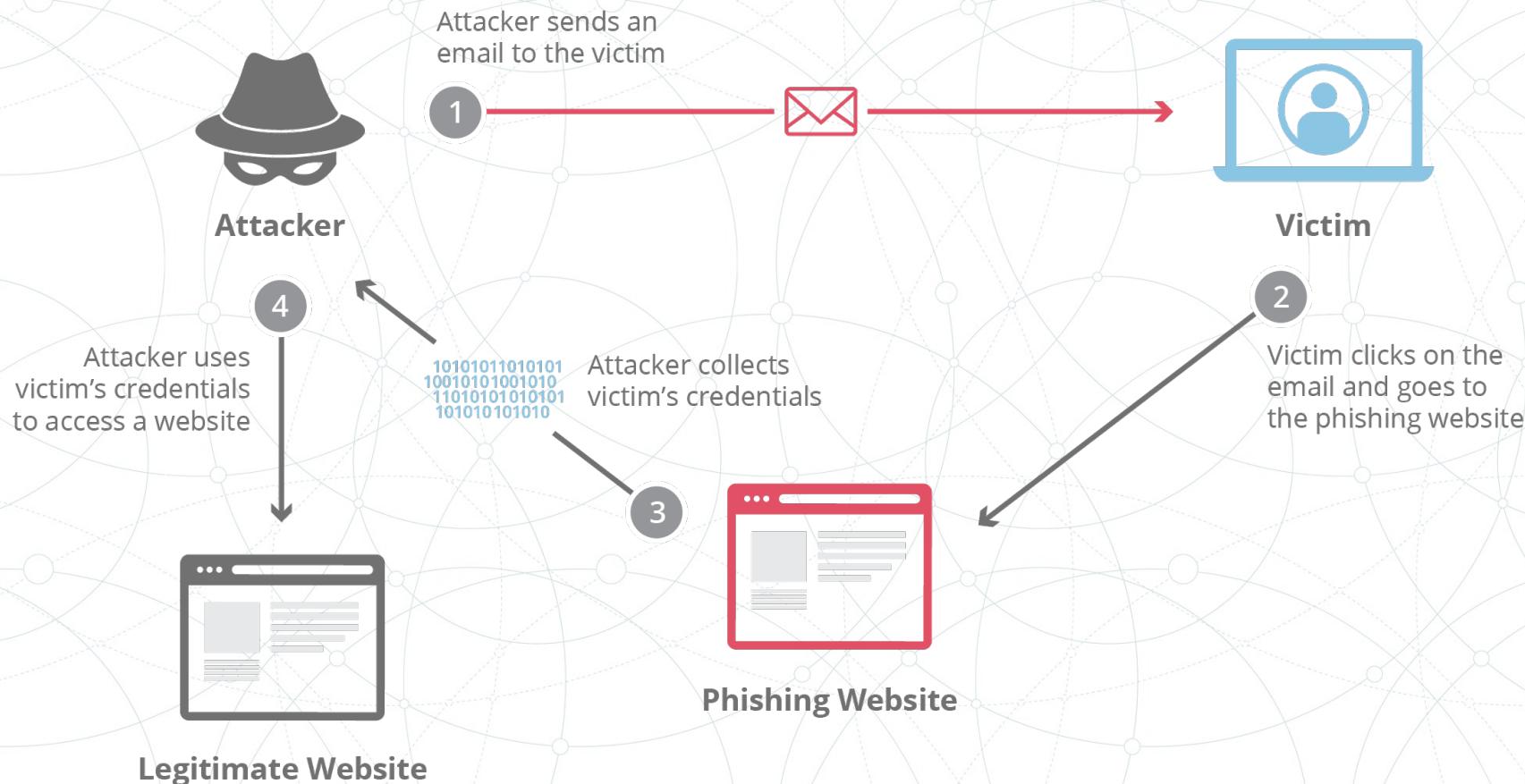


Phishing attacks



- Practice of sending emails which appeared to be from trusted sources with the aim of getting more personal information or appealing users to do something
 - Combines social engineering and technical trickery that could involve an attachment to an email which loaded malware into your device
 - The attacker devices link with an illegitimate website that tricks you into downloading malware or offering over your personal information

Phishing attacks: How it is done



Malware attack

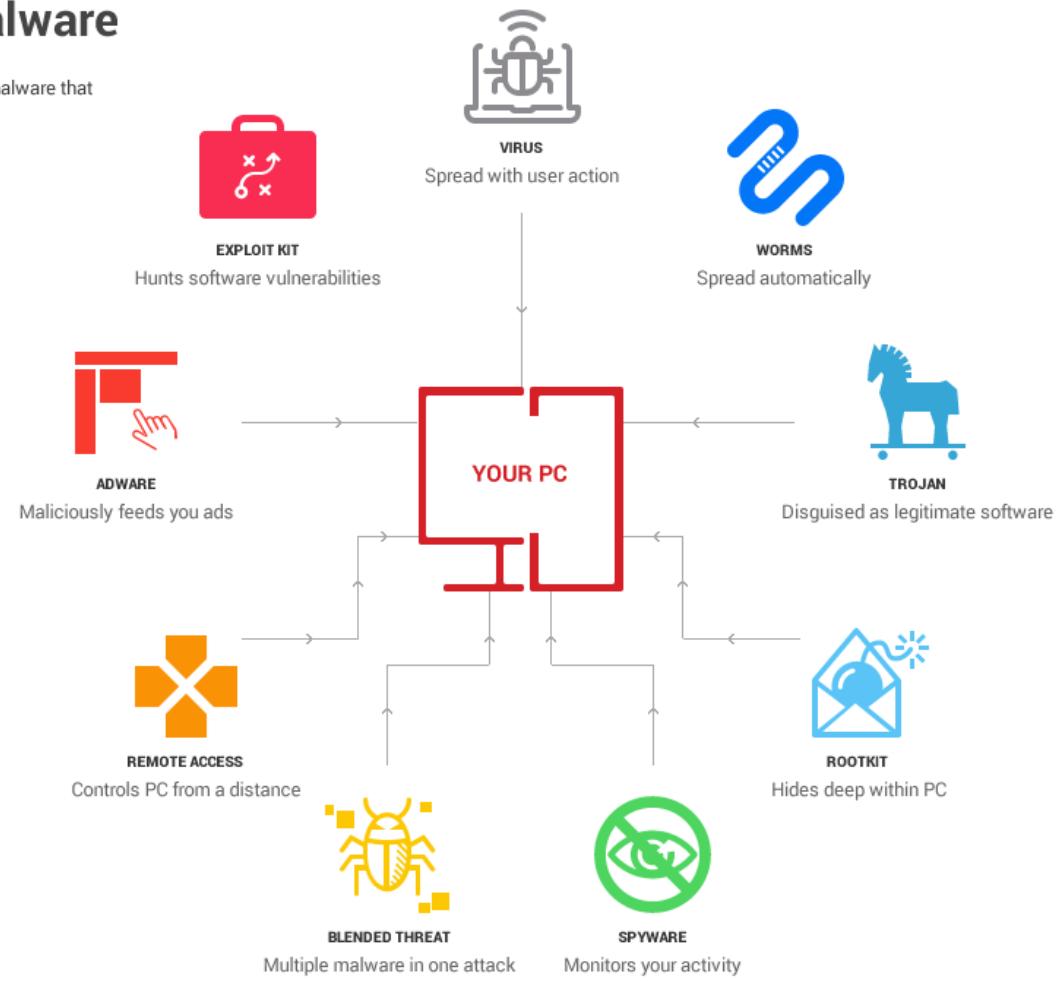
- Malicious software that's put in your system while not your consent
- It will attach itself to legitimate code and propagate; it will lurk in helpful applications or replicate itself across the net
 - Examples: viruses, trojans, worms, ransomware, etc.

Types of Malware

- Virus
- Worms
- Trojan

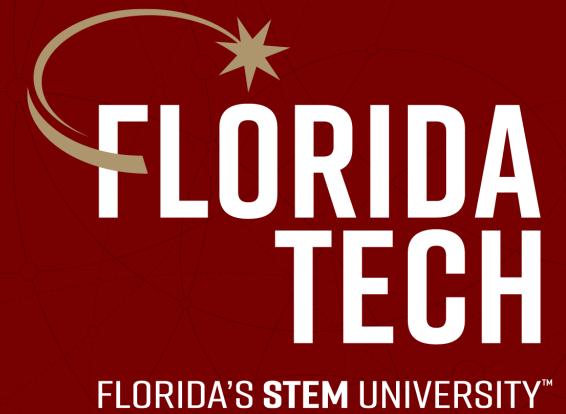
Types of malware

These are the main types of malware that can be found across the web.



Discussion

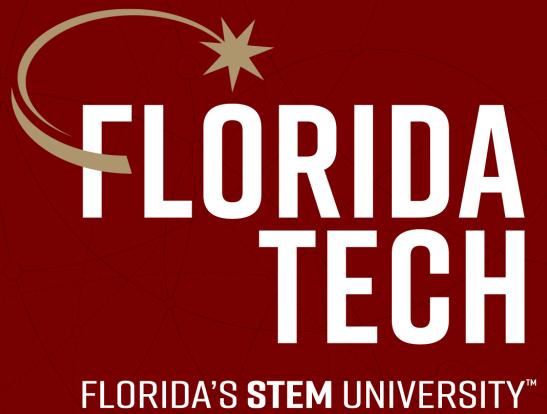
- Which attack type(s) do you think the most used against Wifi / Mobile?
 - DDoS
 - Phishing
 - MiTM
 - Malware



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

2. WiFi (IEEE 802.11)

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

WiFi

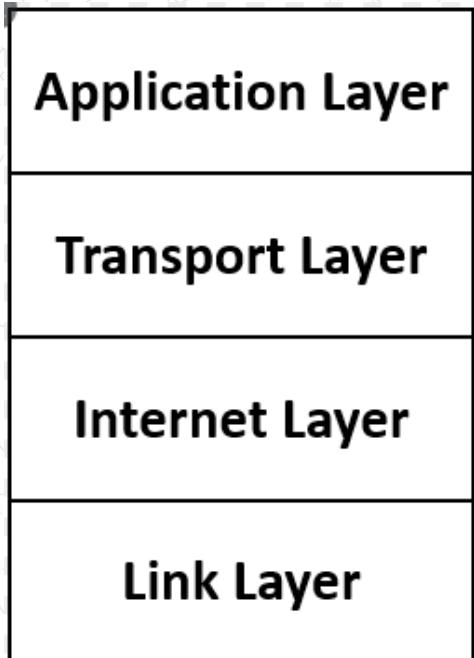
WiFi Security

WEP

WPA/WPA2/WPA3

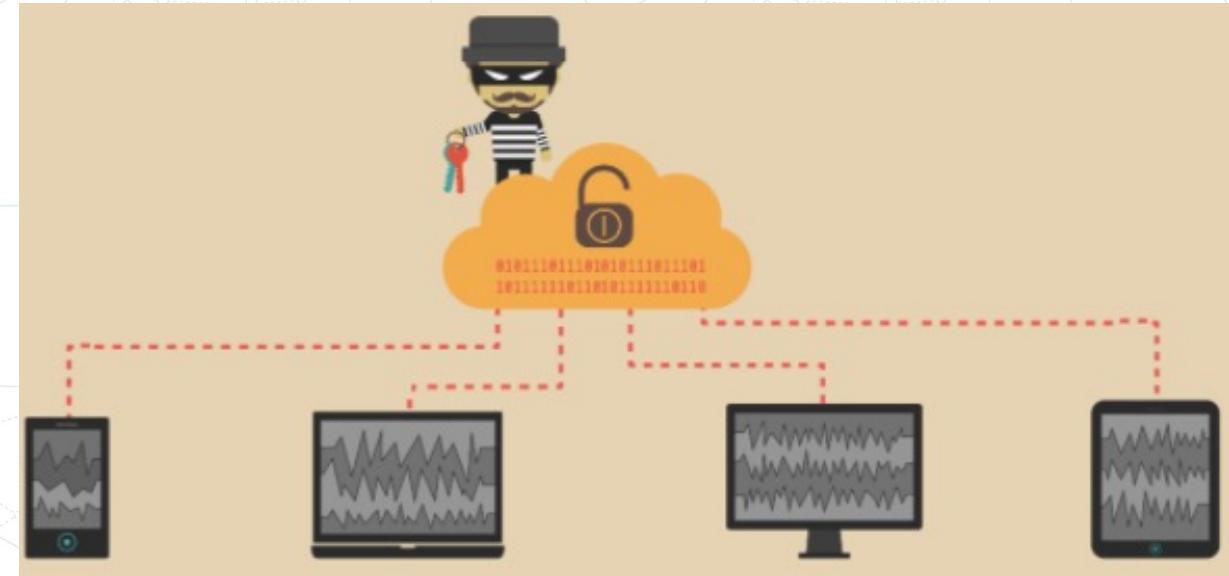
Recall: Network Protocols

- **Application Layers:** End-user applications
- **Transport Layer:** Data transfer from end to end
- **Internet (Network) Layer:** Routing of data from source to destination
- **Link (Physical) Layer:** Physical media carrying the data



Recall: Some Common Attack Types

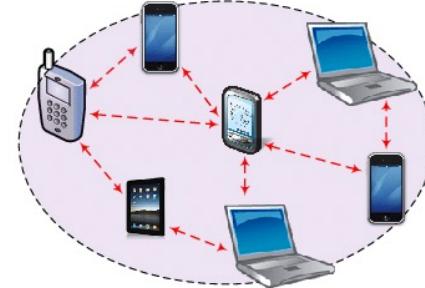
- Denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks
- Man-in-the-middle (MitM) attack
- Phishing and spear phishing attacks
- Malware attack



802.11



Infrastructure-based wireless networks



Wireless ad hoc networks

- IEEE defines the 802.11 -> link layer wireless protocol
- Provides wireless access to wired networks with Access Points (AP)
 - Can be used without an AP which is referred as ad-hoc or IBSS (Independent Basic Service Set) mode
- Three packet categories:
 - Data, management, and control

802.11 Packet Types

- Data packets are used for carrying high-level data (e.g., IP packets)
- Management packets to control the network
 - Attackers are interested in these mostly
 - Beacon and De-authentication are examples of management
- Control packets are used to mediating access to shared medium
 - Examples: RTS (Request to Send) and CTS (Clear to Send)

802.11 Packets

- Three addresses: Source, destination and BSSID (basic service set ID)
 - Where the packets are going, who sent them and what AP to go through
- BSSID identifies the AP and its collected stations
 - Often same MAC address as the wireless interface on the AP

WiFi History

- 802.11 -> 1997
- 802.11a/b -> 2000
- 802.11g -> 2003
- 802.11n -> 2009
- 802.11ac -> 2013
- 802.11ax -> 2017

802.11 (legacy)

- The original version of the standard IEEE 802.11 was released in 1997 and clarified in 1999, but is now obsolete
 - It specified two net bit rates of 1 or 2 megabits per second (Mbit/s) and it specified three alternative physical layer technologies:
 - Diffuse infrared operating at 1 Mbit/s; frequency-hopping spread spectrum (FHSS) operating at 1 Mbit/s or 2 Mbit/s
 - Direct-sequence spread spectrum (DSSS) operating at 1 Mbit/s or 2 Mbit/s
 - The latter two radio technologies used microwave transmission over the Industrial Scientific Medical frequency band at 2.4 GHz

802.11b

- Has a maximum raw data rate of 11 Mbit/s, and uses the same media access method defined in the original standard
- 802.11b products appeared on the market in early 2000, since 802.11b is a direct extension of the modulation technique defined in the original standard
- The dramatic increase in throughput of 802.11b (compared to the original standard) along with simultaneous substantial price reductions led to the rapid acceptance of 802.11b as the definitive wireless LAN technology
- Devices using 802.11b experience interference from other products operating in the 2.4 GHz band
 - E.g., microwave ovens, Bluetooth devices, cordless telephones, and some amateur radio equipment

802.11a

- Provides protocols that allow transmission and reception of data at rates of 1.5 to 54 Mbit/s
 - It has seen widespread worldwide implementation, particularly within the corporate workspace
- While the original amendment is no longer valid, the term *802.11a* is still used by wireless access point (cards and routers) manufacturers to describe interoperability of their systems at 5 GHz, 54 Mbit/s
- The 802.11a standard uses the same data link layer protocol and frame format as the original standard, but an OFDM (Orthogonal frequency-division multiplexing) based air interface (physical layer)

802.11a

- Since the 2.4 GHz band is heavily used to the point of being crowded, using the relatively unused 5 GHz band gives 802.11a a significant advantage
- However, this high carrier frequency also brings a disadvantage: the effective overall range of 802.11a is less than that of 802.11b / g
 - In theory, 802.11a signals are absorbed more readily by walls and other solid objects in their path due to their smaller wavelength, and, as a result, cannot penetrate as far as those of 802.11b
 - In practice, 802.11b typically has a higher range at low speeds (802.11b will reduce speed to 5.5 Mbit/s or even 1 Mbit/s at low signal strengths)
 - 802.11a also suffers from interference, but locally there may be fewer signals to interfere with, resulting in less interference and better throughput

802.11g

- Works in the 2.4 GHz band (like 802.11b) but uses the same OFDM based transmission scheme as 802.11a
 - It operates at a maximum physical layer bit rate of 54 Mbit/s exclusive of forward error correction codes, or about 22 Mbit/s average throughput
 - 802.11g hardware is fully backward compatible with 802.11b hardware, and therefore is encumbered with legacy issues that reduce throughput by ~21% when compared to 802.11a

802.11g

- It was rapidly adopted in the market starting in January 2003, well before ratification, due to the desire for higher data rates as well as to reductions in manufacturing costs
- Details of making b and g work well together occupied much of the lingering technical process; in an 802.11g network, however, activity of an 802.11b participant will reduce the data rate of the overall 802.11g network
- Like 802.11b, 802.11g devices suffer interference from other products operating in the 2.4 GHz band, for example wireless keyboards

802.11n

- 802.11n is an amendment that improves upon the previous 802.11 standards by adding multiple-input multiple-output (MIMO) antennas 802.11n operates on both the 2.4 GHz and the 5 GHz bands
 - Support for 5 GHz bands is optional
 - It operates at a maximum net data rate from 54 Mbit/s to 600 Mbit/s

802.11ac

- IEEE 802.11ac-2013 is an amendment to IEEE 802.11, published in December 2013, that builds on 802.11n
- Changes compared to 802.11n include wider channels (80 or 160 MHz versus 40 MHz) in the 5 GHz band, more spatial streams (up to eight versus four), higher-order modulation (up to 256-QAM vs. 64-QAM), and the addition of Multi-user MIMO (MU-MIMO)
- Vendors have announced plans to release so-called “Wave 2” devices with support for 160 MHz channels, four spatial streams, and MU-MIMO in 2014 and 2015

802.11ax

- In 2017, the first draft of the 802.11ax standard is published, and manufacturers begin to make devices based on the draft specification
- 802.11ax adds OFDMA, 1024-QAM modulation, up to 8 spatial streams and bi-directional MU-MIMO, for data rates up to 1200 Mbps over a single spatial stream in a 160 MHz channel

Renaming WiFi

- 802.11b = Wi-Fi 1
- 802.11a = Wi-Fi 2
- 802.11g = Wi-Fi 3
- 802.11n = Wi-Fi 4
- 802.11ac = Wi-Fi 5
- 802.11ax = Wi-Fi 6

802.11 Security

- Wired Equivalency Protocol (WEP)
- Wi-Fi Protected Access (WPA)

WEP

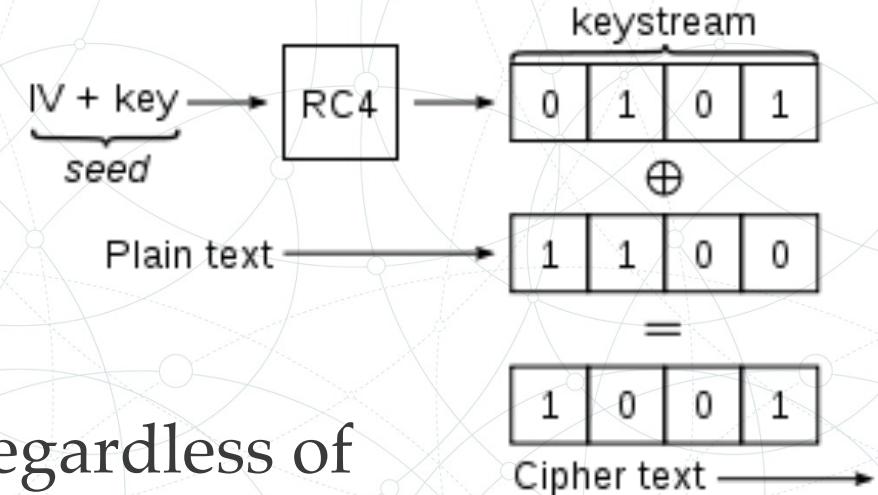
- Introduced in 1997, was the first attempt at wireless protection
- The aim was to add security to wireless networks by encrypting data
 - If wireless data were intercepted, it would be unrecognizable to the interceptors since it had been encrypted
 - However, systems that are authorized on the network would be able to recognize and decrypt the data
 - This is because devices on the network make use of the same encryption algorithm

WEP: Encryption

- WEP initially used a 64-bit key with the RC4 stream encryption algorithm to encrypt data transmitted wirelessly
- Later versions of the protocol added support for 128-bit keys and 256-bit keys for improved security
- WEP uses a 24-bit initialization vector, which resulted in effective key lengths of 40, 104 and 232 bits

WEP: Encryption

- This is a static key, which means all traffic, regardless of device, is encrypted using a single key
- A WEP key allows computers on a network to exchange encoded messages while hiding the messages' contents from intruders
- This key is what is used to connect to a wireless-security-enabled network



WEP: Data integrity:

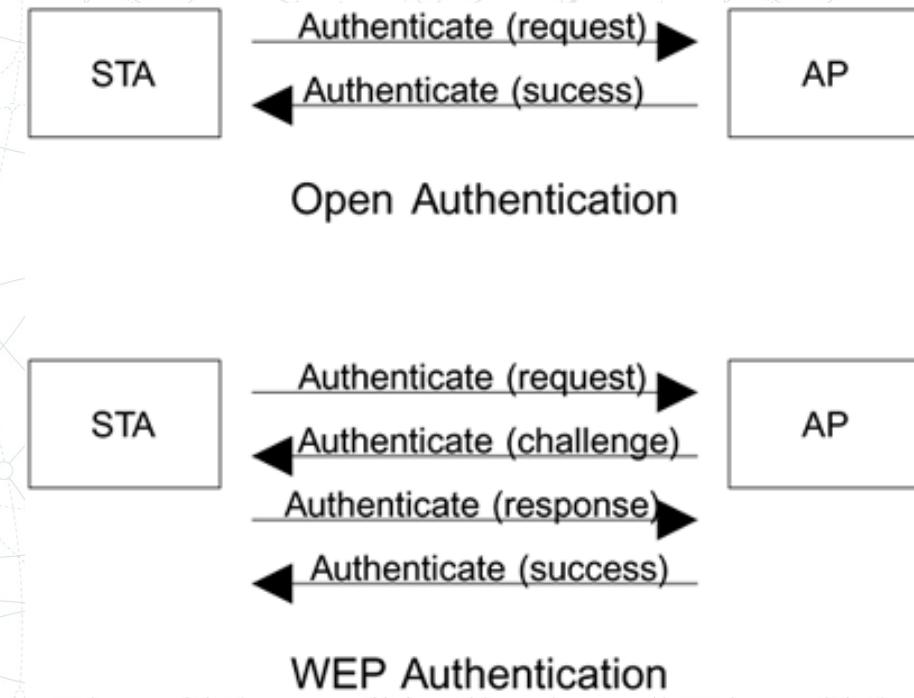
- WEP uses the CRC-32 checksum algorithm to check that transmitted data is unchanged at its destination
- The sender uses the CRC-32 cyclic redundancy check to generate a 32-bit hash value from a sequence of data
- The recipient uses the same check on receipt
- If the two values differ, the recipient can request a retransmission

WEP: Authentication

- WEP authenticates clients when they first connect to the wireless network access point:
 - **Open System Authentication;** Wi-Fi-connected systems can access any WEP network access point, as long as the connected system uses a service set identifier that matches the access point SSID.
 - **Shared Key Authentication;** Wi-Fi-connected systems use a four-step challenge-response algorithm to authenticate.

WEP: Authentication: Shared Key

- A four-step challenge-response handshake:
 - The client sends an authentication request to the AP
 - The AP replies with a clear-text challenge
 - The client encrypts the challenge-text using the configured WEP key and sends it back in 'authentication response'
 - The AP decrypts the response. If this matches the challenge text, the AP sends back a positive reply



WPA

- The Wi-Fi Alliance intended WPA as an intermediate measure to take the place of WEP pending the availability of the full IEEE 802.11i standard
- WPA could be implemented through firmware upgrades on wireless network interface cards designed for WEP that began shipping as far back as 1999
 - However, since the changes required in the wireless access points (APs) were more extensive than those needed on the network cards, most pre-2003 APs could not be upgraded to support WPA

WPA

- The WPA protocol implements the Temporal Key Integrity Protocol (TKIP)
 - WEP used a 64-bit or 128-bit encryption key that must be manually entered on wireless access points and devices and does not change
 - TKIP employs a per-packet key, meaning that it dynamically generates a new 128-bit key for each packet and thus prevents the types of attacks that compromised WEP

WPA

- WPA also includes a Message Integrity Check, which is designed to prevent an attacker from altering and resending data packets
 - This replaces the cyclic redundancy check (CRC) that was used by the WEP standard
- CRC's main flaw was that it did not provide a sufficiently strong data integrity guarantee for the packets it handled
 - Well-tested message authentication codes existed to solve these problems, but they required too much computation to be used on old network cards

WPA

- WPA uses a message integrity check algorithm called TKIP to verify the integrity of the packets
 - TKIP is much stronger than a CRC, but not as strong as the algorithm used in WPA2
- However, despite these improvements, elements of WPA came to be exploited – which led to WPA2

WPA: Pre-shared Key

- Similar to WEP
 - Requires connecting party to provide a key to access to network
- Pre-shared key is between 8-63 ASCII long
- Encryption relies on pairwise master key (PMK)
 - That is computed from the pre-shared key and SSID

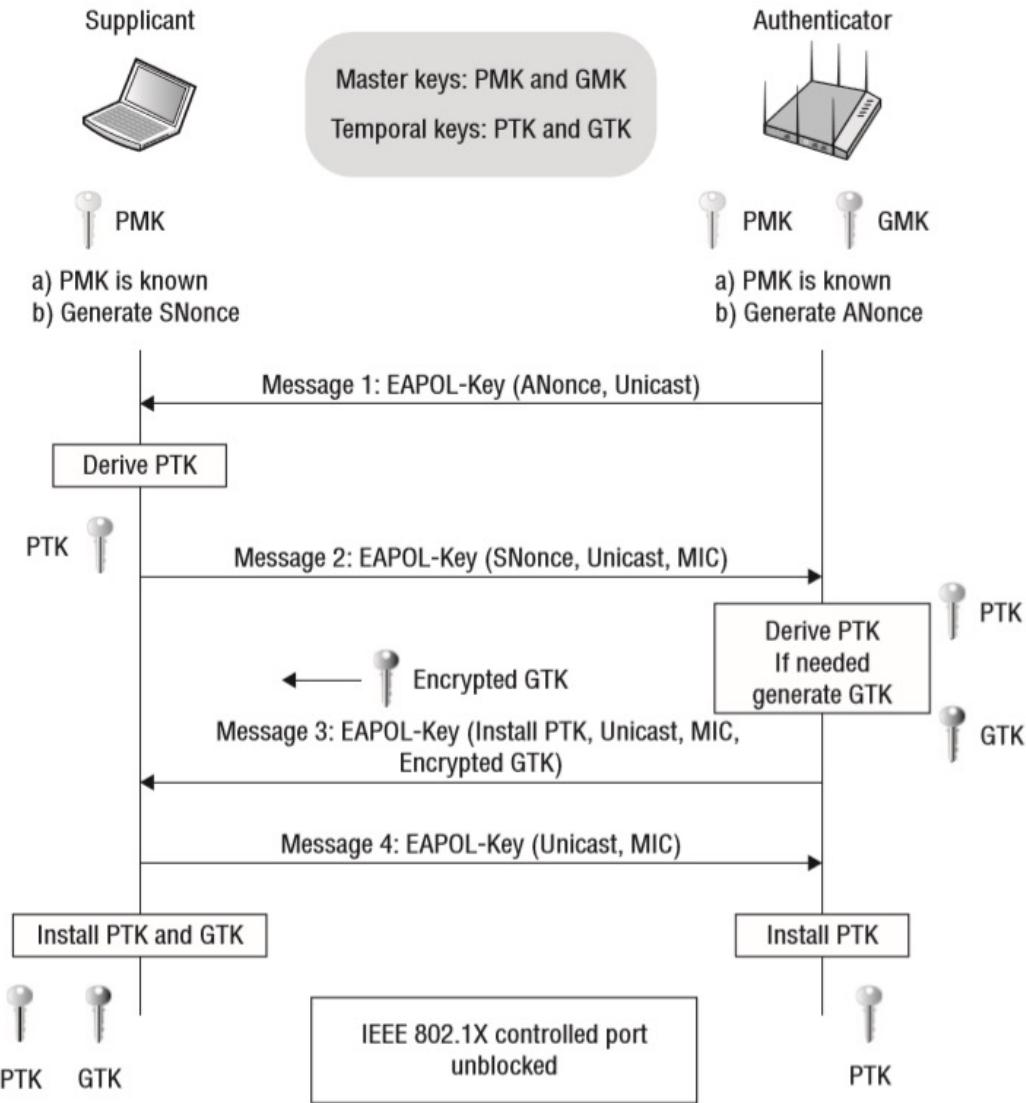
WPA: Pre-shared Key

- Once client has PMK, it will negotiate with AP for PTK (pairwise transient key)
 - These keys are created dynamically every time client connects
 - Function of PMK, random numbers, and MAC addresses
 - To ensure they are unique and non-repeating

WPA: Pre-shared Key

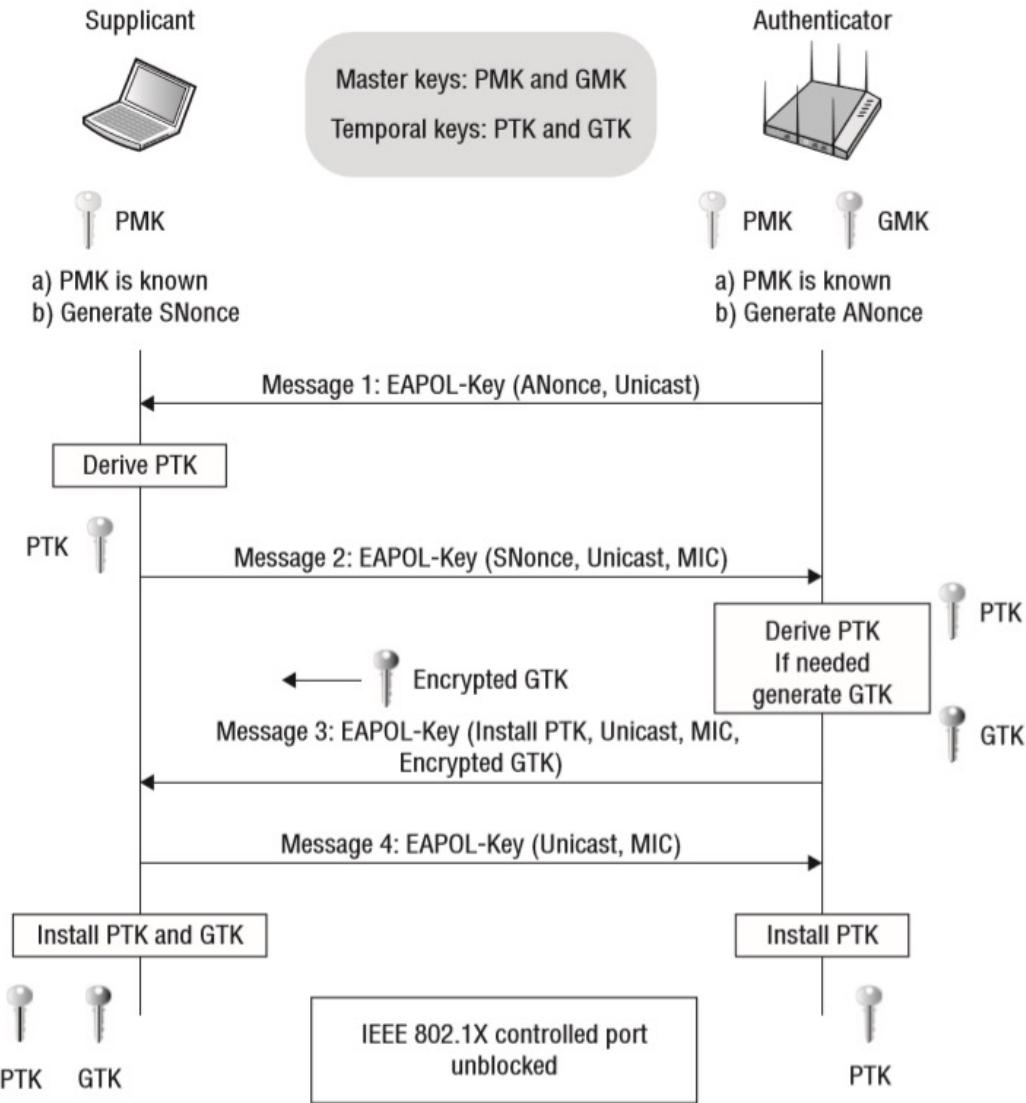
- **Message1:** AP sends EAPOL message with Anonce (random number) to the device to generate PTK
 - Client device knows AP's MAC because its connected to it
 - It has PMK, Snonce and its own MAC address
 - Once it receives Anonce from AP, it has all the inputs to create the PTK

$$PTK = PRF (PMK + Anonce + SNonce + Mac (AA) + Mac (SA))$$



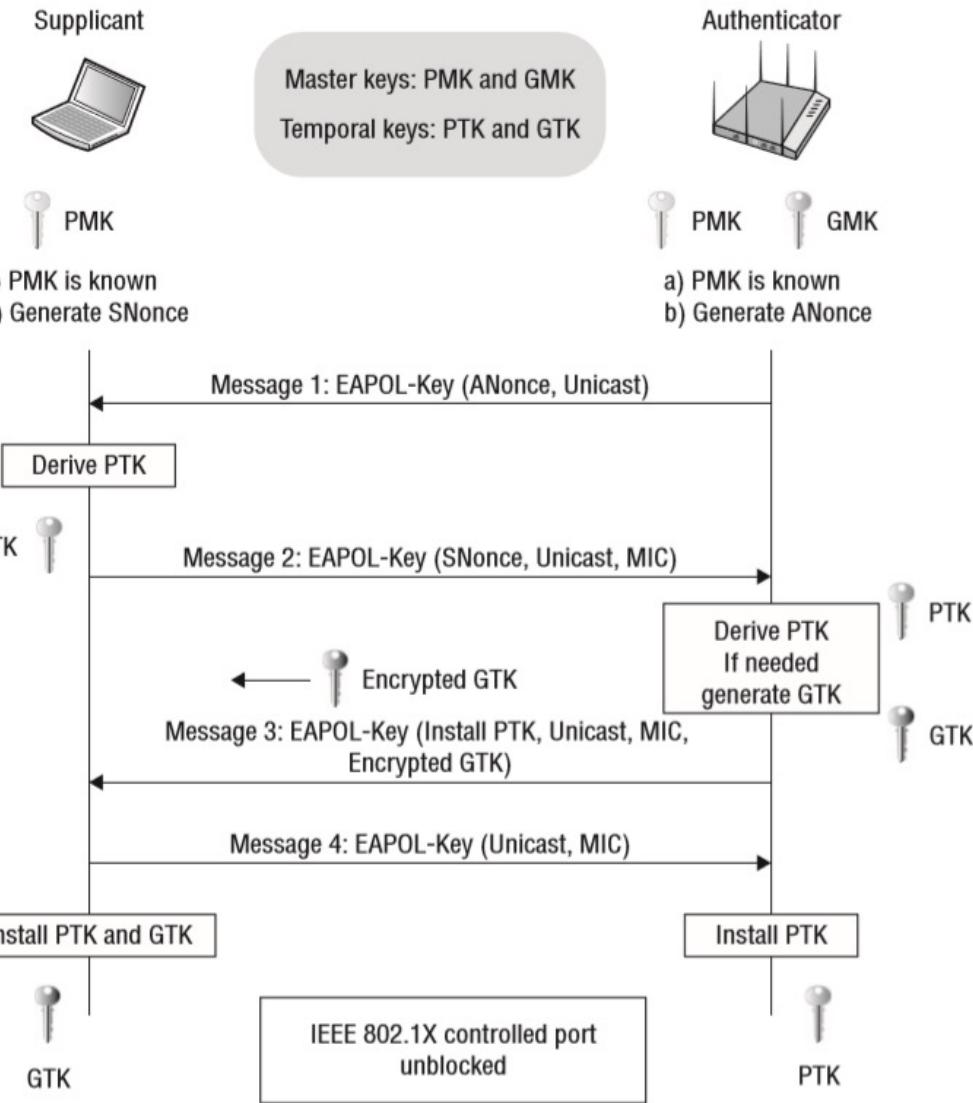
WPA: Pre-shared Key

- **Message2:** Once the device has created its PTK it sends out SNonce which is needed by the access point to generate PTK as well
 - The device sends EAPOL to AP message2 with MIC (message integrity check) to make sure when the access point can verify whether this message corrupted or modified
 - Once SNonce received by the AP it can generate PTK as well for unicast traffic encryption



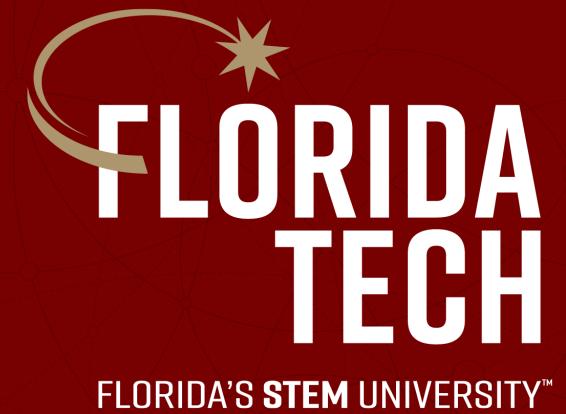
WPA: Pre-shared Key

- **Message3:** EAPOL message3 is sent from AP to client device containing GTK
 - AP creates GTK without the involvement of the client from GMK
- **Message4:** Fourth and last EPOL message will be sent from the client to AP just to confirm that Keys have been installed



WPA: Pre-shared Key

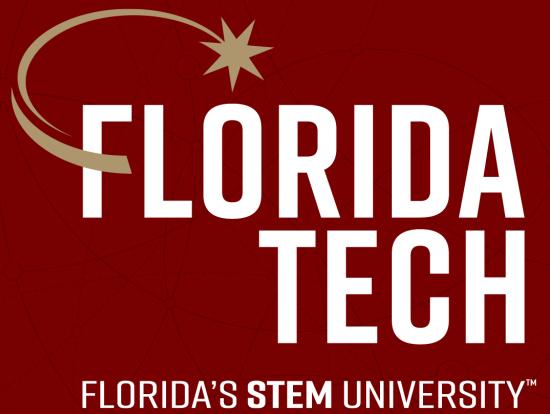
- **4-way handshake Result: Control port unlocked**
 - Once the 4-way handshake is completed successfully virtual control port which blocks all the traffic will be open and now encrypted traffic can flow
 - Now all unicast traffic will be encrypted with PTK and all multicast traffic will be encrypted via GTK which created in the 4-way handshake process



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

3. WPA2 / WPA3

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

WiFi Security

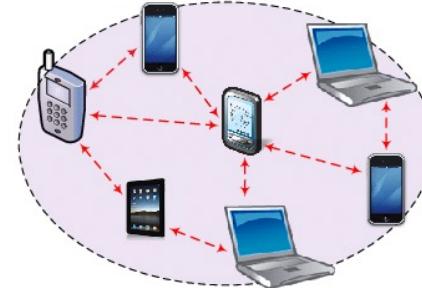
WEP

WPA/WPA2/WPA3

Recall: 802.11



Infrastructure-based wireless networks



Wireless ad hoc networks

- IEEE defines the 802.11 -> link layer wireless protocol
- Provides wireless access to wired networks with Access Points (AP)
 - Can be used without an AP which is referred as ad-hoc or IBSS (Independent Basic Service Set) mode
- Three packet categories:
 - Data, management, and control

Recall: WiFi History

- 802.11 -> 1997
- 802.11a/b -> 2000
- 802.11g -> 2003
- 802.11n -> 2009
- 802.11ac -> 2013
- 802.11ax -> 2017

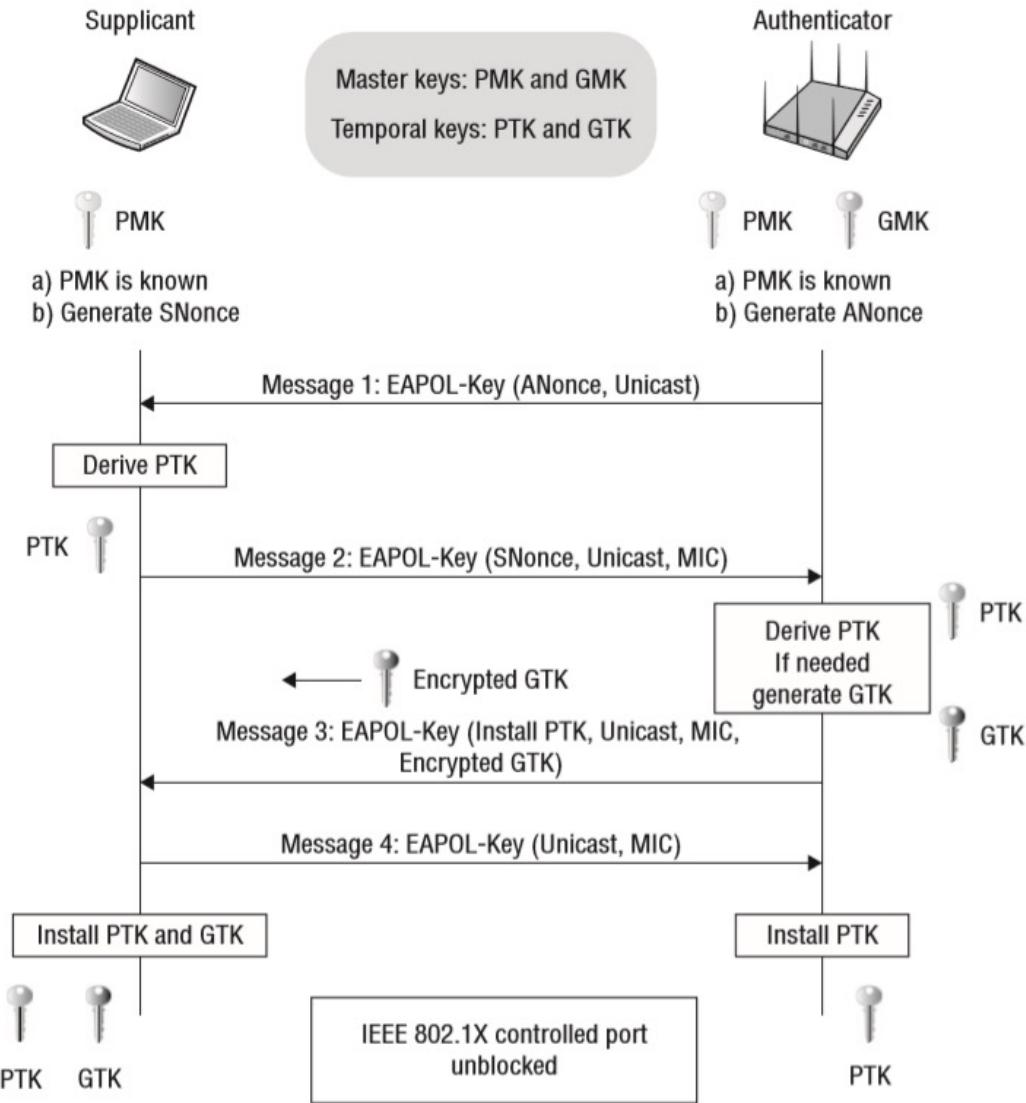
Recall: 802.11 Security

- Wired Equivalency Protocol (WEP)
- Wi-Fi Protected Access (WPA)

Recall: WPA: Pre-shared Key

- **Message1:** AP sends EAPOL message with Anonce (random number) to the device to generate PTK
 - Client device knows AP's MAC because its connected to it
 - It has PMK, Snonce and its own MAC address
 - Once it receives Anonce from AP, it has all the inputs to create the PTK

$$PTK = PRF (PMK + Anonce + SNonce + Mac (AA) + Mac (SA))$$



WPA2

- WPA still uses the RC4 encryption algorithm, and retained other weaknesses from WEP
- WPA2 was introduced in 2004 and was an upgraded version of WPA
- WPA2 is based on the robust security network (RSN) mechanism and operates on two modes:
 - **Personal mode or Pre-shared Key (WPA2-PSK)** – which relies on a shared passcode for access and is usually used in home environments
 - **Enterprise mode (WPA2-EAP)** – as the name suggests, this is more suited to organizational or business use

WPA2

- Both modes use the CCMP (Counter Mode Cipher Block Chaining Message Authentication Code Protocol)
- The CCMP protocol is based on the Advanced Encryption Standard (AES) algorithm, which provides message authenticity and integrity verification
- CCMP is stronger and more reliable than WPA's original TKIP, making it more difficult for attackers to spot patterns

WPA2

- However, WPA2 still has drawbacks
 - For example, it is vulnerable to key reinstallation attacks (KRACK)
 - KRACK exploits a weakness in WPA2, which allows attackers to pose as a clone network and force the victim to connect to a malicious network instead
- This enables the hacker to decrypt a small piece of data that may be aggregated to crack the encryption key
- Yet, WPA2 is still considered sufficiently secure and more secure than WEP or WPA

WPA3

- WPA3 is the third iteration of the Wi-Fi Protected Access protocol
- The Wi-Fi Alliance introduced WPA3 in 2018
- WPA3 devices became widely available in 2019 and are backwards compatible with devices that use the WPA2 protocol
- WPA3 introduced new features for both personal and enterprise use

WPA3: Features

- **Individualized data encryption:** When logging on to a public network, WPA3 signs up a new device through a process other than a shared password
- WPA3 uses a Wi-Fi Device Provisioning Protocol (DPP) system that allows users to use Near Field Communication (NFC) tags or QR codes to allow devices on the network
- In addition, WPA3 security uses 256 encryption rather than the previously used 128-bit encryption

WPA3: Features

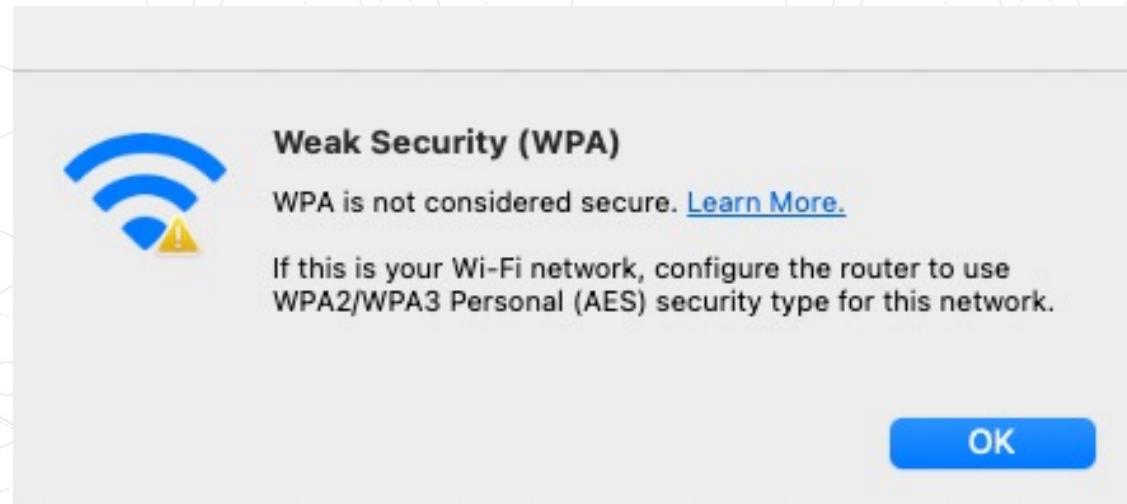
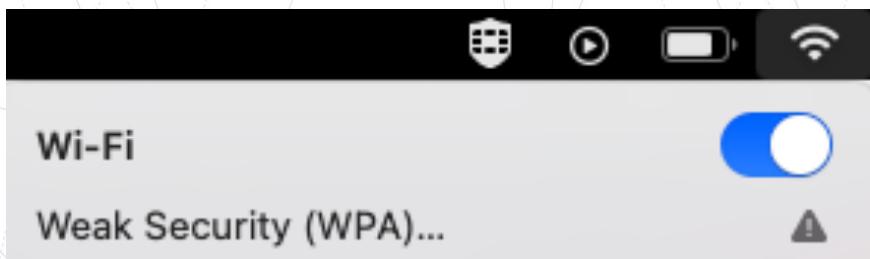
- **Simultaneous Authentication of Equals protocol:**
 - This is used to create a secure handshake, where a network device will connect to a wireless access point, and both devices communicate to verify authentication and connection
 - Even if a user's password is weak, WPA3 provides a more secure handshake using Wi-Fi DPP

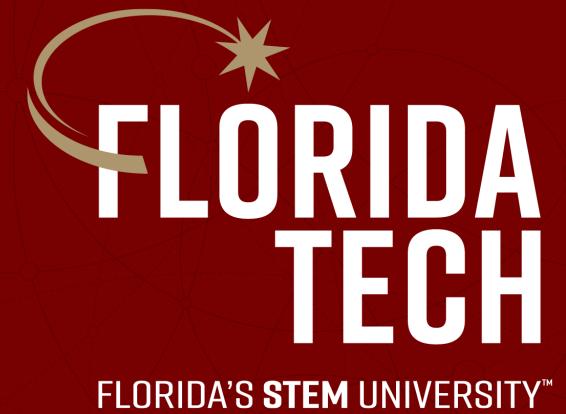
WPA3: Features

- **Stronger brute force attack protection:**
 - WPA3 protects against offline password guesses by allowing a user only one guess, forcing the user to interact with the Wi-Fi device directly, meaning they would have to be physically present every time they want to guess the password
 - WPA2 lacks built-in encryption and privacy in public open networks, making brute force attacks a significant threat

What do you use?

- For MacOS; just check the wireless symbol on top:

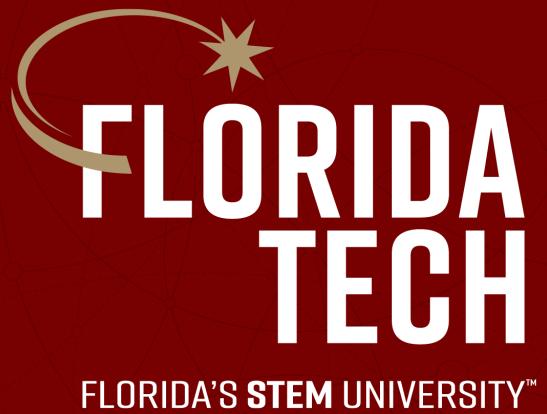




Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

4. Digital Signal Processing

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Definitions

Frequency, Amplitude, Wavelength

Modulations

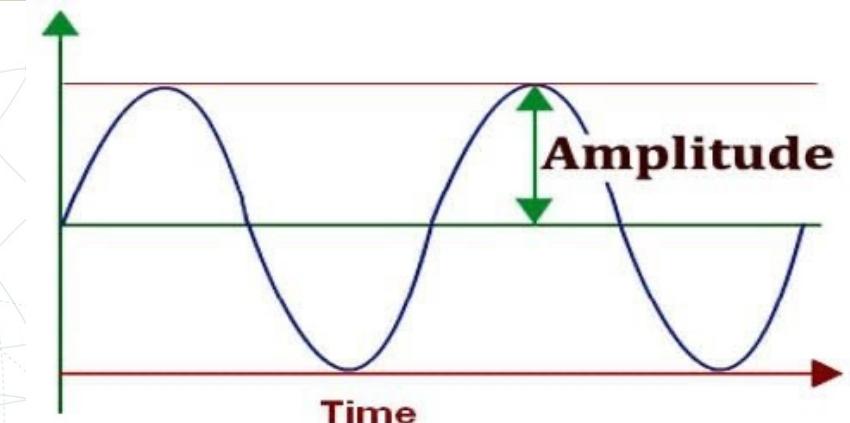
Multiple Access

Amplitude

Measure of change of waveform per signal period

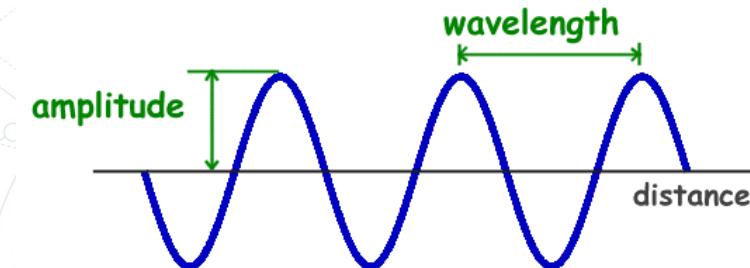
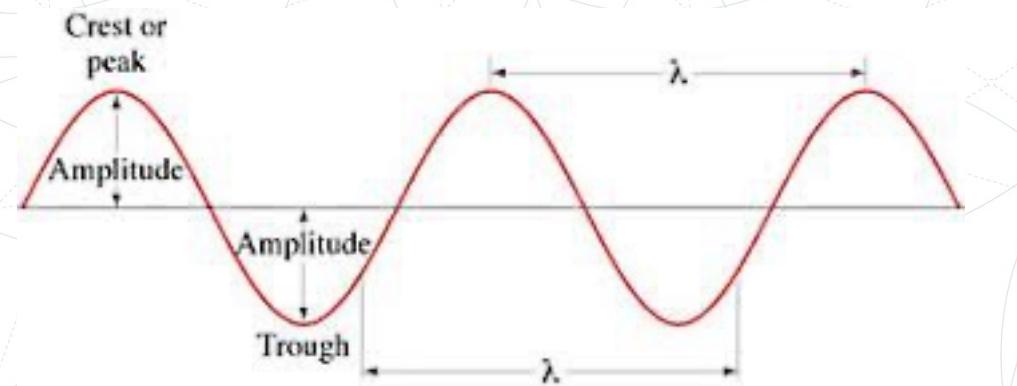
The amplitude is a measure of the strength or intensity of the wave

- For example, when looking at a sound wave, the amplitude will measure the loudness of the sound
- The energy of the wave also varies in direct proportion to the amplitude of the wave



Wavelength

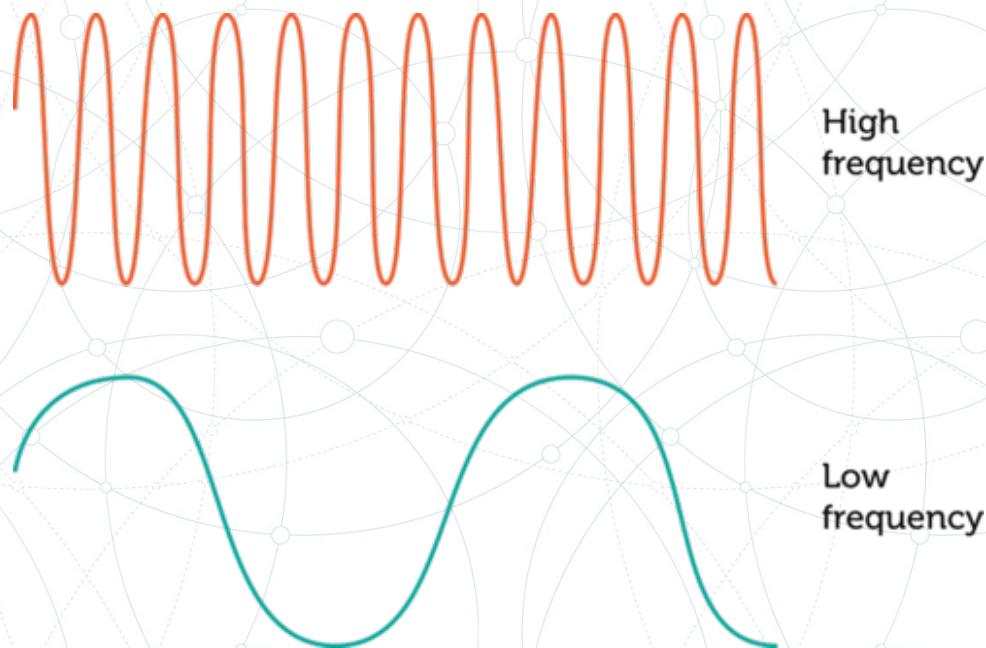
- The wavelength of a wave is the distance between two corresponding points on back-to-back cycles of a wave
- This can be measured between two crests of a wave or two troughs of a wave
- The wavelength is represented in physics by the Greek letter lambda (λ)



Frequency

How often the waveform repeats per second (measured in hertz)

The unit for wave frequency is the **hertz (Hz)**, where 1 hertz equals 1 wave passing a fixed point in 1 second



Period

- The period of the wave is the time between wave crests
 - Measured in time units such as seconds and represented by the upper case "T"
- The period and frequency are closely related to each other
 - Period equals 1 over the frequency and the frequency is equal to one over the period
 - Reciprocals of each other as shown in the following formulas

$$T = 1/f$$

Velocity

- How fast the disturbance of the wave is moving
 - The speed of mechanical waves depends on the medium that the wave is traveling through
 - For example, sound will travel at a different speed in water than in air
 - The velocity of a wave is usually represented by the letter "v."
 - The velocity can be calculated by multiplying the frequency by the wavelength

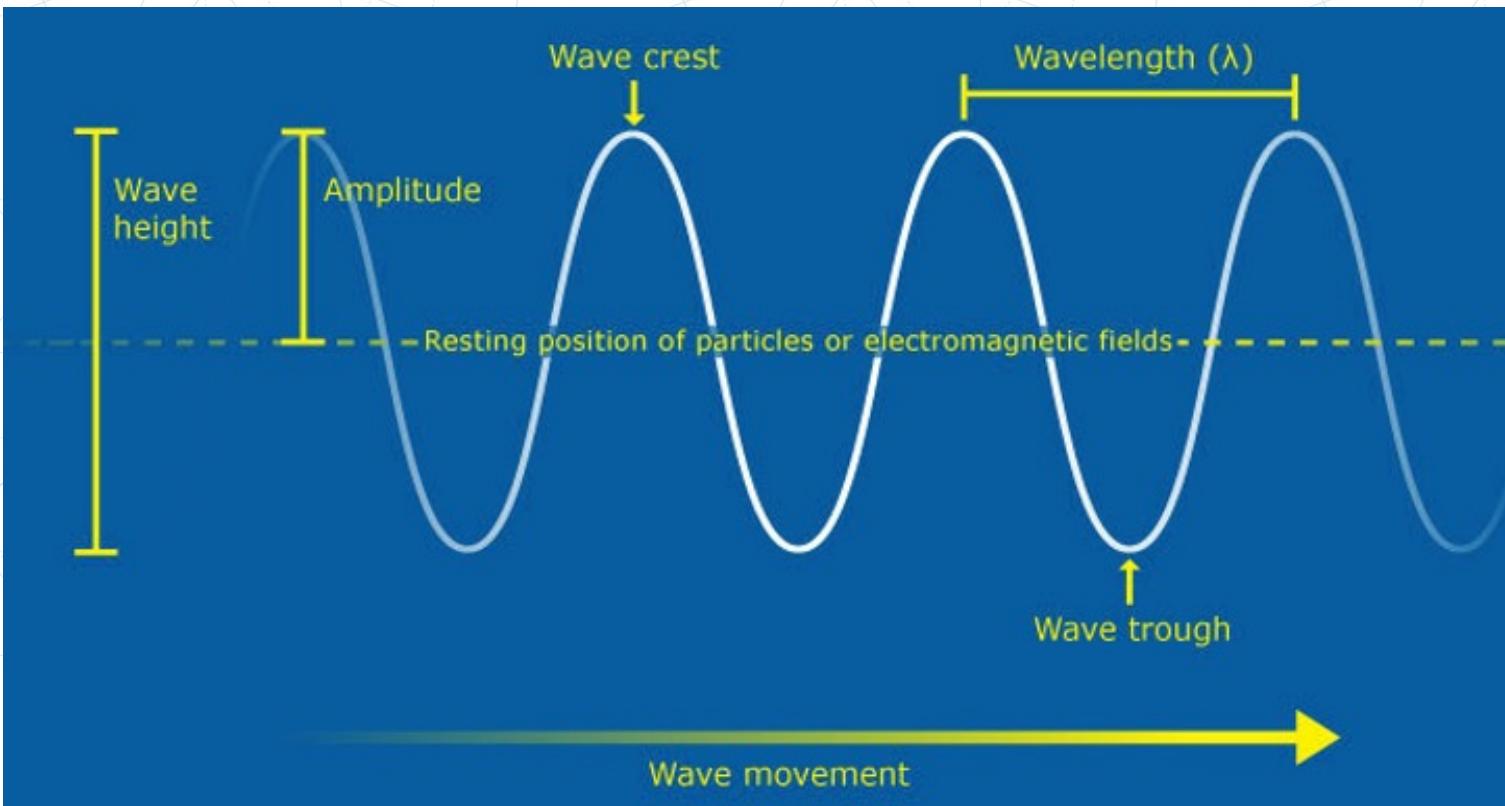
$$v = f * \lambda$$

Sampling Rate

- Average number of samples obtained in 1 second; reduction of continuous signal to discrete signal
- S1 is sampled at $f/2$
S2 is sampled at $f/4$
S2 misses $1/2$ the data

Digital Signal Processing

- DSP: process by which we encode or decode a signal using frequency, sampling, modulation, coding

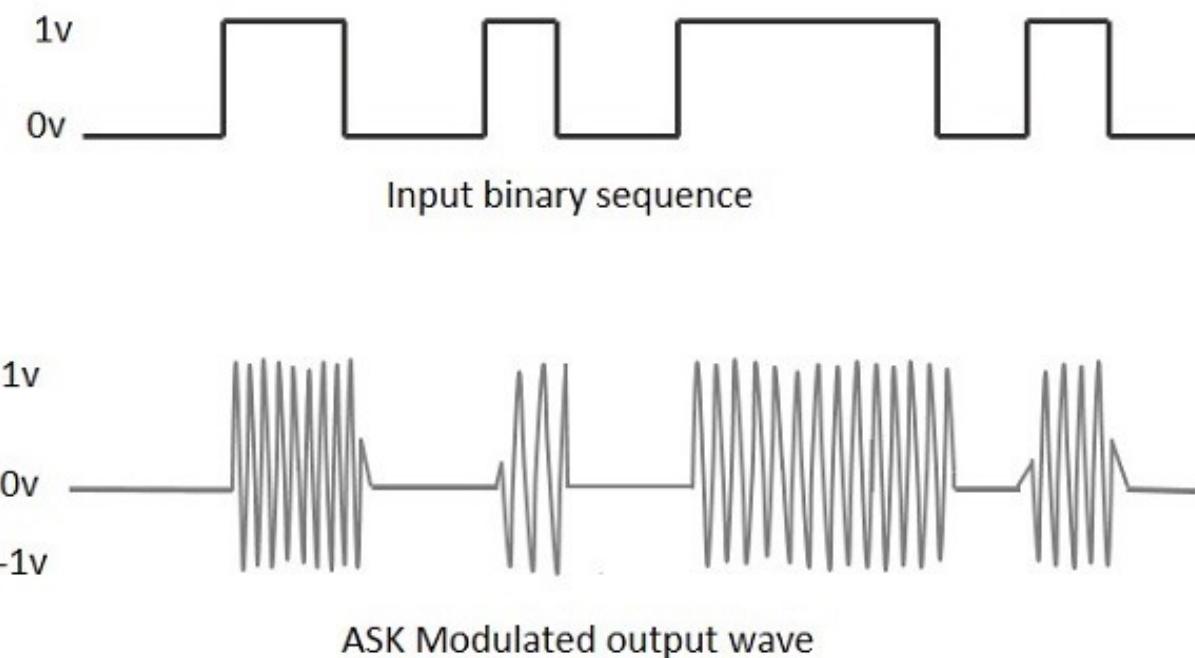


Digital Signal Modulation (Keying)

- Modifying properties of waveform to encode data
- Different potential approaches:
 - Amplitude Shift Keying (ASK)
 - Frequency Shift Keying (FSK)
 - Phase Shift Keying (PSK)
 - Differential BPSK
 - Quadrature Phase Shift Keying

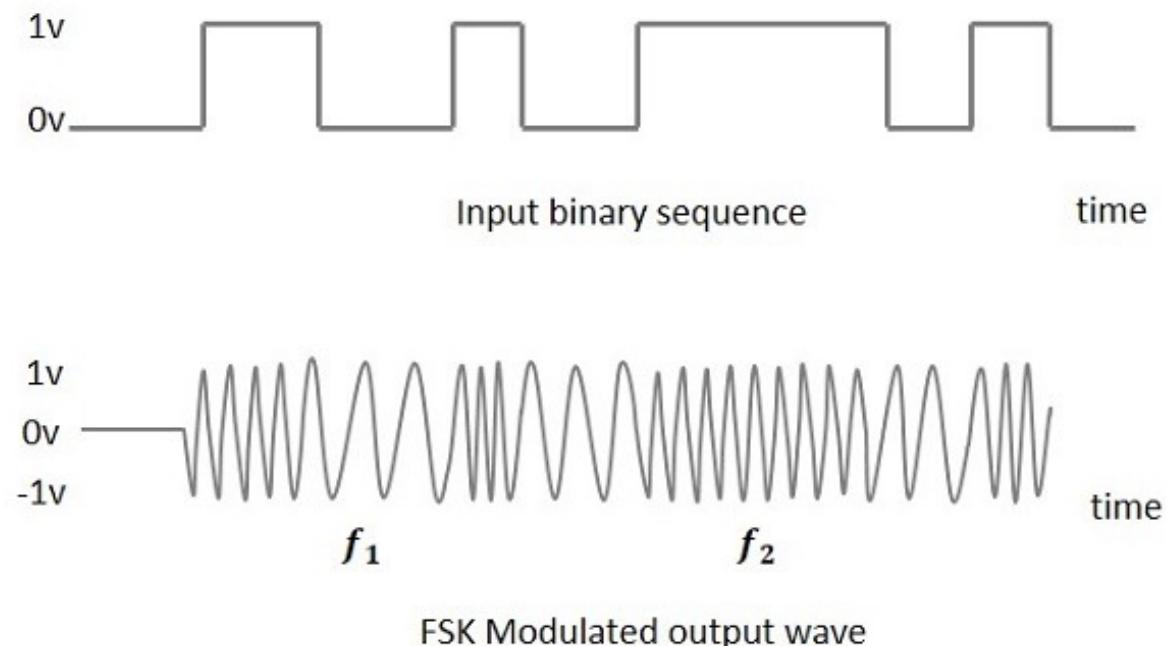
Amplitude Shift Keying (ASK)

- Represents the binary data in the form of variations in the amplitude of a signal
- Any modulated signal has a high frequency carrier
 - The binary signal when ASK modulated, gives a zero value for Low input while it gives the carrier output for High input



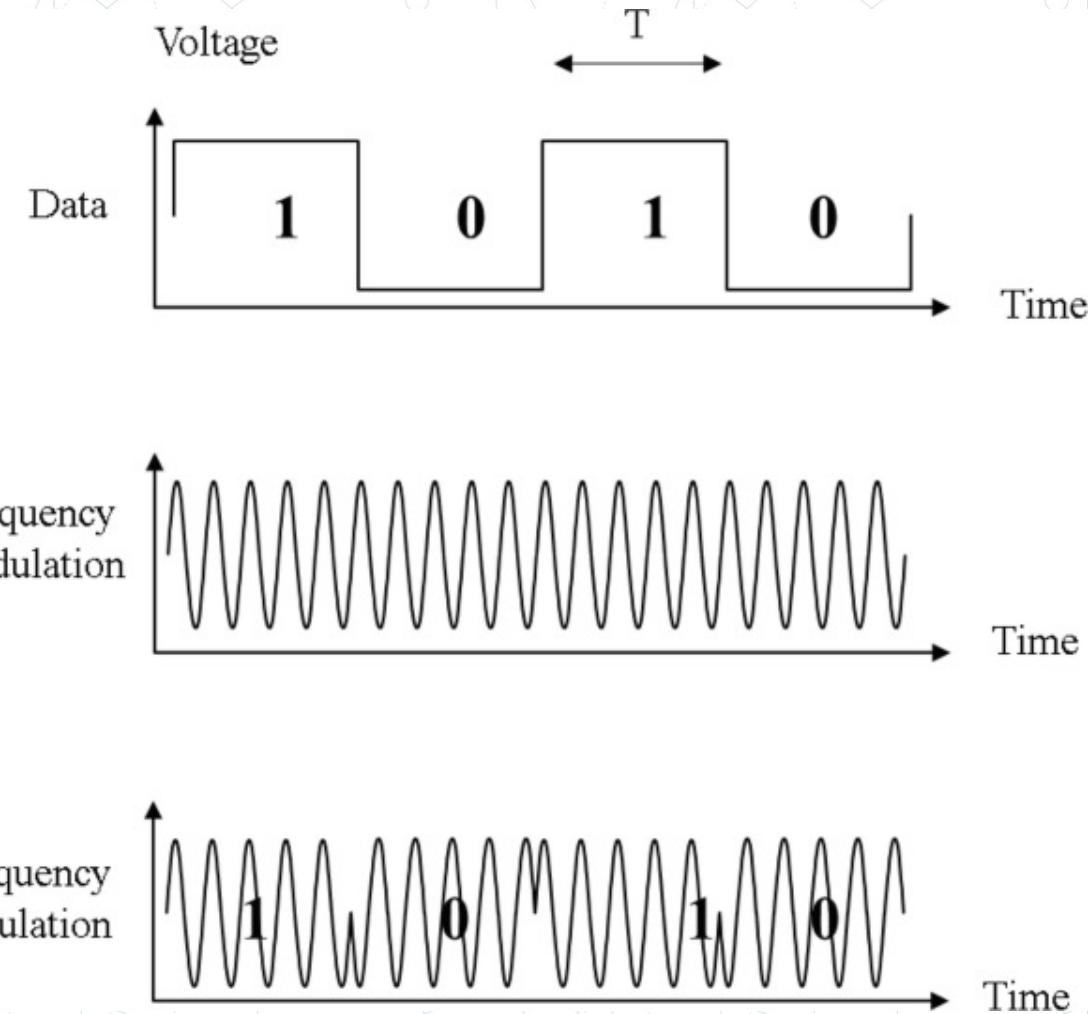
Frequency Shift Keying (FSK)

- The frequency of the carrier signal varies according to the digital signal changes
 - FSK is a scheme of frequency modulation
- The output of a FSK modulated wave is high in frequency for a binary High input and is low in frequency for a binary Low input
 - The binary 1s and 0s are called Mark and Space frequencies



Phase Shift Keying

- The phase of the carrier signal is changed by varying the sine and cosine inputs at a particular time
- PSK technique is widely used for wireless LANs, bio-metric, contactless operations, along with RFID and Bluetooth communications

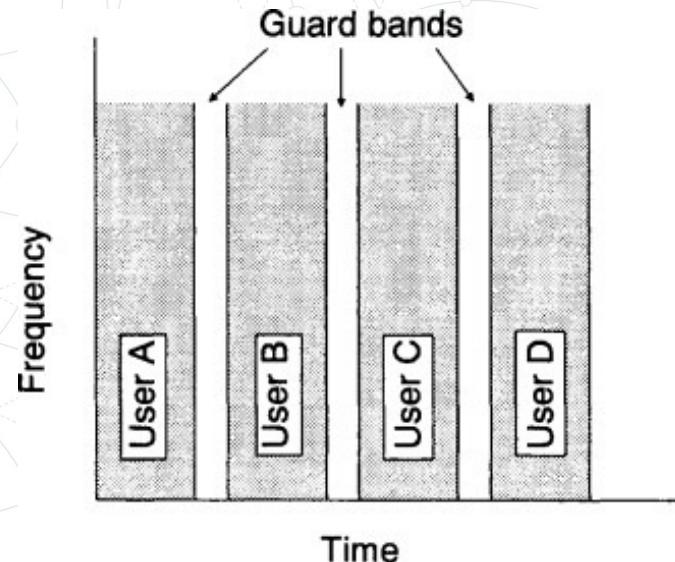


Multiple Access Methods

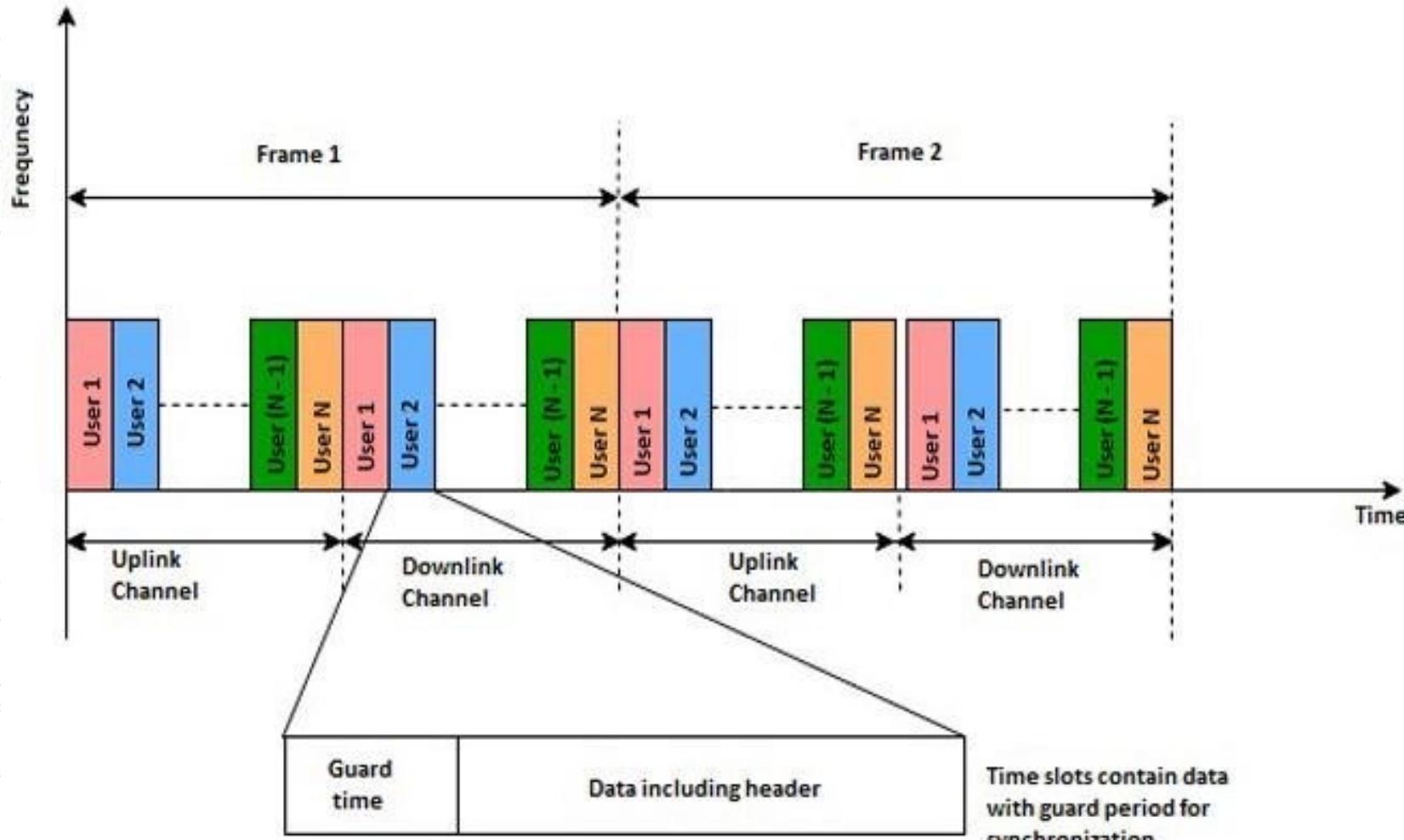
- Time Division Multiple Access
- Frequency Division Multiple Access
- Code Division Multiple Access
- OFDMA

Time Division Multiple Access

- TDMA shares a single carrier frequency with several users
 - Each user makes use of non-overlapping time slots
- Data transmission in TDMA is not continuous, but occurs in bursts
 - Hence handoff process is simpler
- TDMA uses different time slots for transmission and reception thus duplexers are not required
- TDMA has an advantage that it is possible to allocate different numbers of time slots per frame to different users
- Bandwidth can be supplied on demand to different users by concatenating or reassigning time slot based on priority



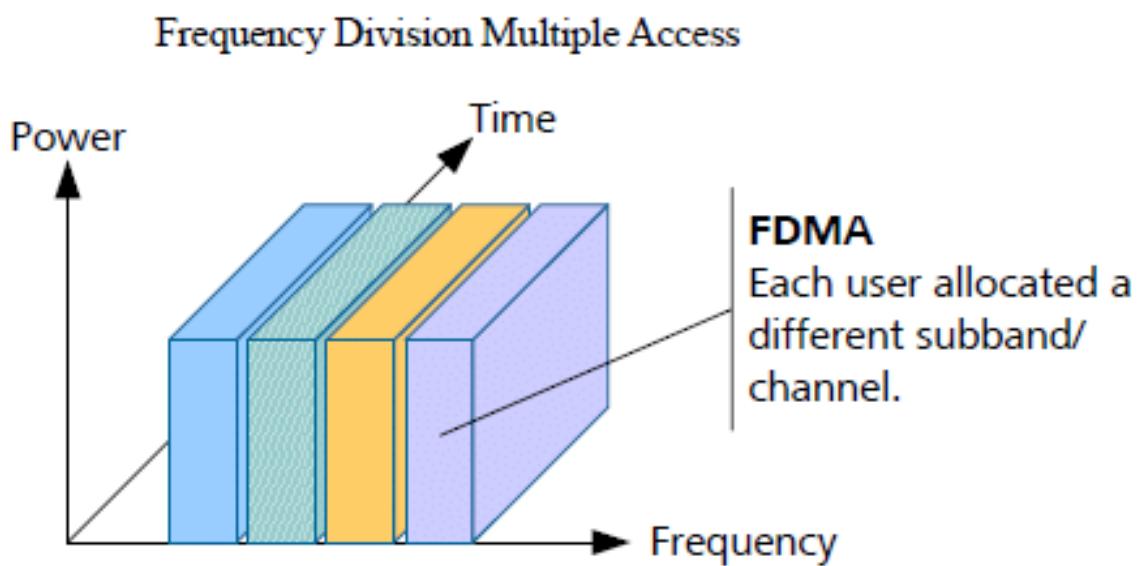
Time Division Multiple Access



- Where continuous transmission is not required, there TDMA is used instead of FDMA

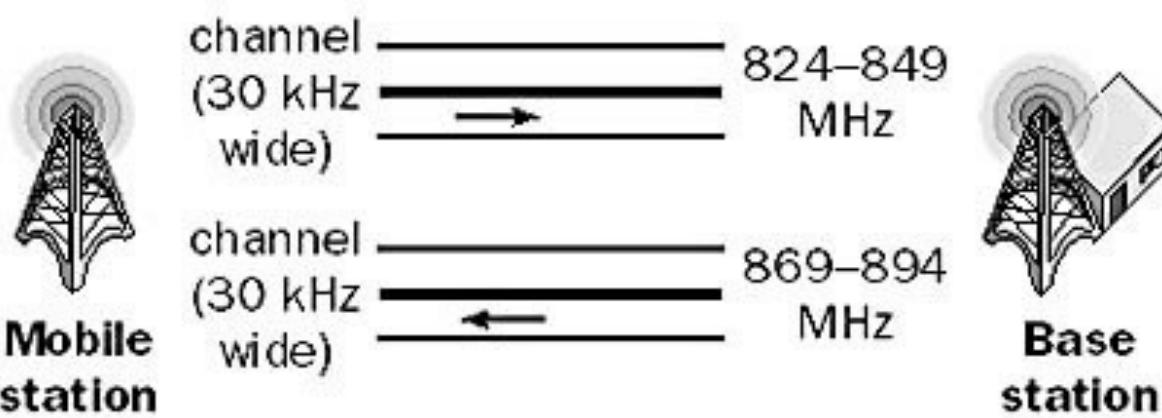
Frequency Division Multiple Access

- FDMA allots a different sub-band of frequency to each different user to access the network
- If FDMA is not in use, the channel is left idle instead of allotting to the other users



Frequency Division Multiple Access

- FDMA is implemented in Narrowband systems
 - It is less complex than TDMA
- The base station BS and mobile station MS, transmit and receive simultaneously and continuously in FDMA



Code Division Multiple Access

- Technique that allows multiple users to simultaneously transmit data signals over a common channel by assigning unique spreading code to each individual user
 - This leads to the increasing of bandwidth used by transmitting stations from a few Hz to a few MHz

Code Division Multiple Access

- CDMA uses the principle of spread spectrum
 - The various signals are modulated after which a single signal is transmitted and is correlated at the receiving end using the spreading function to get the actual data

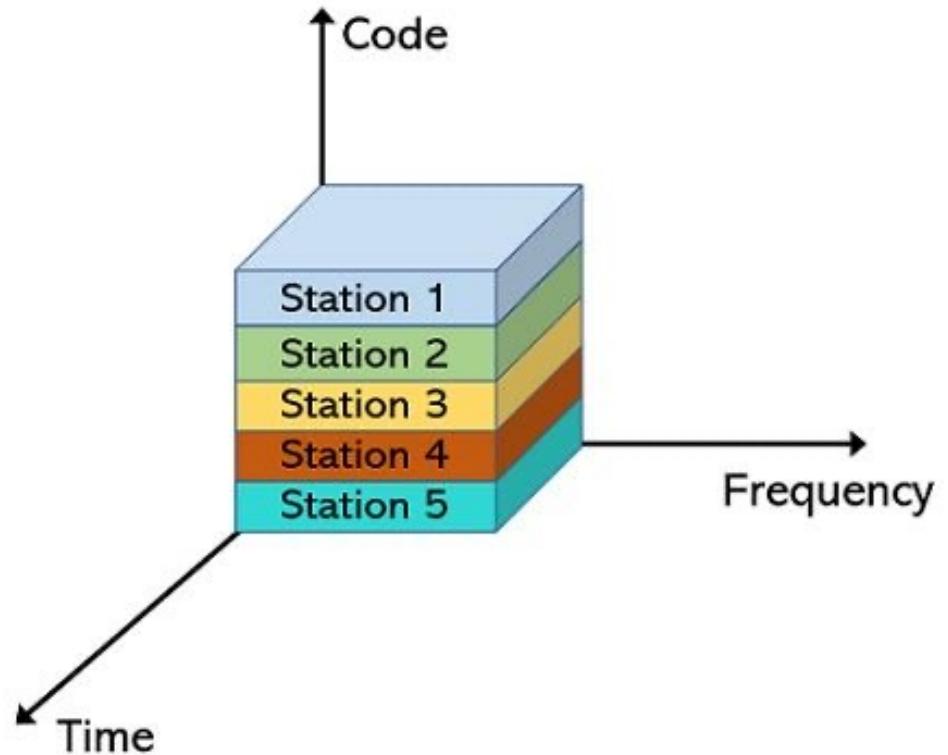


Illustration of CDMA

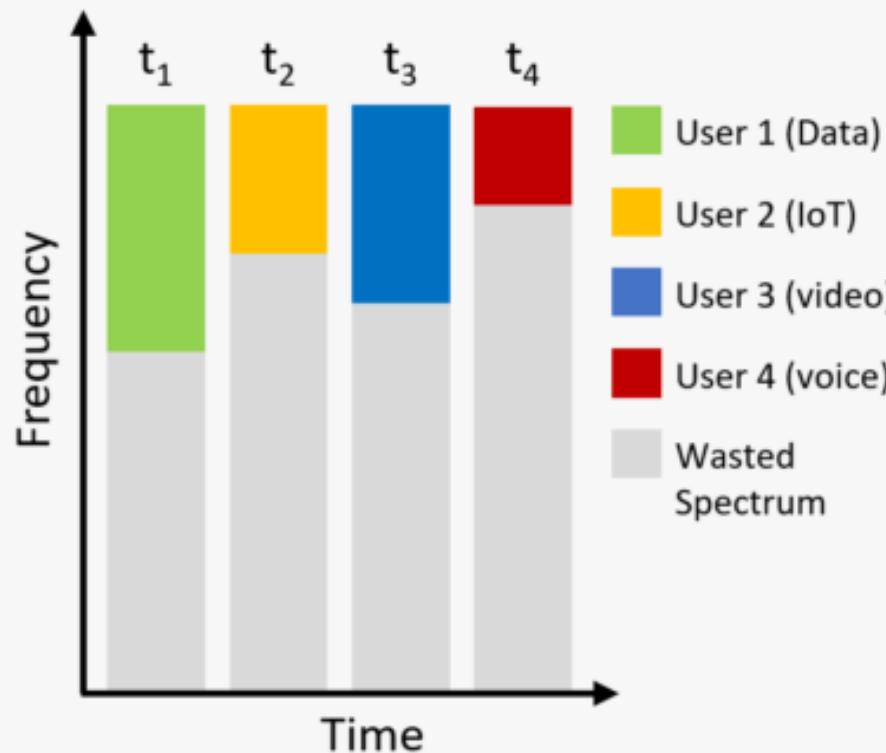
Orthogonal Frequency Division Multiplexing

- Ten 100 kHz channels are better than one 1 MHz Channel
- Frequency band is divided into 256 or more sub-bands
 - Orthogonal -> Peak of one at null of others
- Each carrier is modulated with a BPSK, QPSK, 16-QAM, 64-QAM etc. depending on the noise
- Used in WiFi, LTE, 5G, etc.

OFDM vs OFDMA

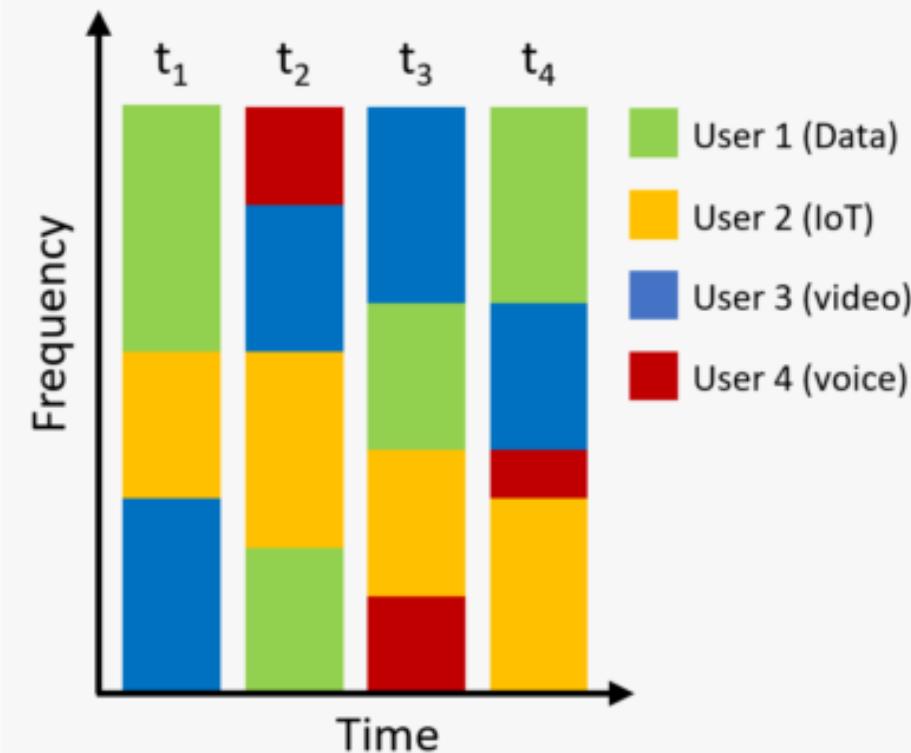
OFDM (Wi-Fi 2-5)

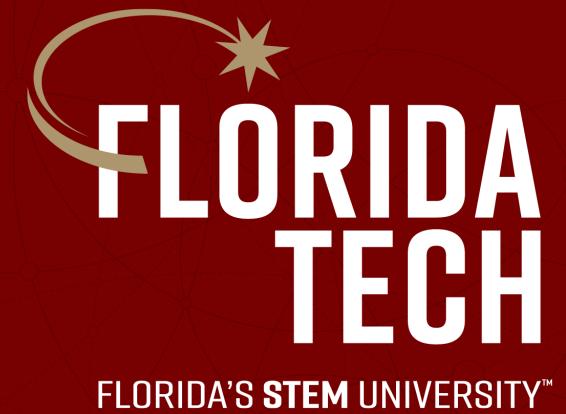
- One user packet per time segment
- Inefficient for small packets
- High voice/video delay



OFDMA (Wi-Fi 6)

- Multiple users packet per time segment
- Highly efficient
- Low voice/video delay

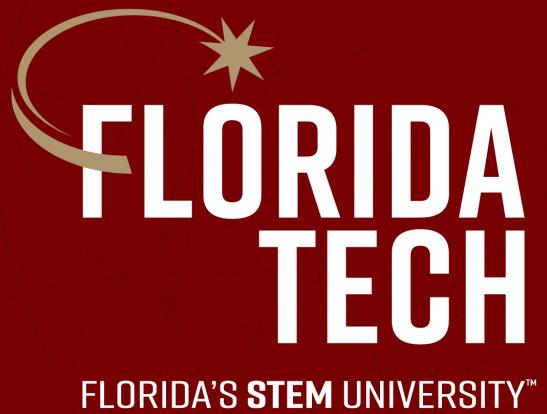




Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

5. WiFi Enumeration

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

802.11 Enumeration

WiFi Sniffer

- A specific type of network analyzer or packet sniffer that is designed to work with wireless networks
- WiFi sniffing can be accomplished with a dedicated piece of electronic equipment or a software application
- Wireless network sniffing is akin to wiretapping a phone line, only without the court order that legalizes the activity

WiFi Sniffer

- There are valid and legal uses for a wireless sniffer tool to be used
 - An example is where a network administrator makes use of one to secure or monitor their network
 - E.g., home users may employ a sniffer to better understand how their network operates
- Unfortunately, a prime reason that WiFi sniffers are used by unscrupulous individuals is to attempt to collect information from, or gain access to, an unsecured network
 - A wireless sniffer allows someone to attack your network from a distance, making it hard to determine if there are attempts to compromise your data

How does WiFi Sniffer Work?

- WiFi sniffers come in two flavors: hardware and software
- They perform the same tasks, though the software route may be more popular in most cases
- You can obtain software WiFi sniffers for Windows, Mac, and mobile operating systems
- In the case of a mobile device, you are in essence turning it into a hardware sniffer when using it with a sniffing application

Hardware WiFi Sniffers

- No need any special level of technical expertise to operate a hardware WiFi sniffer
 - Most devices are small and portable, able to easily fit in your pocket or laptop bag
- You will be alerted by display lights when a wireless network is found within range of your device
- A WiFi sniffer for Android devices uses add-on tools and will offer a similar display when searching for networks
 - Will be able to detect a wireless signal in spite of interference from Bluetooth devices, microwaves or cell phones

Software WiFi Sniffers

- You can download these tools for just about any operating system
 - Some of these tools have advanced features that allow you to do more than just locate the nearest wireless network
- Tools such as Wireshark and Network Miner serve as WiFi sniffers for the Windows platform
- Users can download tools such as aircrack-ng or Zanti to perform WiFi sniffing on an Android device (or Linux/MAC)

KisMAC

- OS X passive scanner
- Has support for GPS
- Outdated

NetSpot

- NetSpot collects every detail about surrounding Wi-Fi networks and presents wireless data as an interactive table
- It lets you troubleshoot and improve your network's coverage, capacity, performance, APs configurations, signal level, interference, noise, etc.
 - Locate your busiest and least occupied channels

NetSpot Example Capture:

| SSID | BSSID | Alias | Channel | Band | Security | Vendor | Mode | Level (SNR) | | Signal | Signal... | Avg | Max | Min | Noise | Noi... | Last seen |
|---------------------|--------------------------|--------------|---------------|----------------------|-------------------|--------------|--|---|--|------------|------------|------------|------------|-----------|------------|-----------------|-----------|
| | | | | | | | | ^ | - | | | | | | | | |
| ATTfHV8Cwa | 02:30:44:21:5C:8F | | 6 | 2.4GHz | WPA2 Personal | 02:30:44 | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -91 | -84 | -94 | - | 0% | 1min 13s ago | |
| ExpressPress#2 | 02:30:44:21:5C... | 6 | 2.4GHz | WPA2 Personal | 02:30:44 | b/g/n | <div style="width: 100%; background-color: red;">red</div> | <div style="width: 92%; background-color: #ccc;">92</div> | <div style="width: 8%; background-color: #ccc;">8%</div> | -91 | -82 | -94 | -94 | 6% | now | | |
| BLINK-4SFM | AC:CC:FC:AF:76:... | | 6 | 2.4GHz | Open | AC:CC:FC | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -47 | -24 | -65 | - | 0% | 7h 2min 48s... | |
| MySpectrumWiFiE4-2G | 98:F7:81:B8:C7:E5 | | 6 | 2.4GHz | WPA2 Personal | ARRIS | ac | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -82 | -75 | -85 | - | 0% | 59s ago | |
| Poncho | E4:F7:5B:13:FC:20 | | 1 | 2.4GHz | WPA2 Personal | ARRIS | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -91 | -91 | -93 | - | 0% | 6h 50min 11s... | |
| Drasha33 | 2C:00:AB:40:2D:... | | 6 | 2.4GHz | WPA2 Personal | ARRIS | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -92 | -89 | -94 | - | 0% | 59s ago | |
| MySpectrumWiFi3A-2G | 84:BB:69:CC:B... | 6 | 2.4GHz | WPA2 Personal | ARRIS | ac | <div style="width: 100%;">-</div> | <div style="width: 94%; background-color: #ccc;">94</div> | <div style="width: 6%; background-color: #ccc;">6%</div> | -92 | -90 | -94 | -94 | 6% | now | | |
| The_Travers | 10:93:97:19:D4:70 | | 1 | 2.4GHz | WPA2 Personal | ARRIS | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -92 | -91 | -93 | - | 0% | 6h 47min 48s... | |
| ATTfHV8Cwa | B0:DA:F9:7B:8... | 1 | 2.4GHz | WPA2 Personal | ARRIS | b/g/n | <div style="width: 100%; background-color: red;">red</div> | <div style="width: 92%; background-color: #ccc;">92</div> | <div style="width: 8%; background-color: #ccc;">8%</div> | -89 | -87 | -93 | -94 | 6% | now | | |
| ATTQH4bXYs_2.4 | FC:AE:34:A4:4F:... | | 1 | 2.4GHz | WPA2 Personal | ARRIS | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -93 | -93 | -93 | - | 0% | 7h 3min 6s... | |
| chittum2014 | AC:B3:13:A7:C8:10 | | 1 | 2.4GHz | WPA2 Personal | ARRIS | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -93 | -92 | -95 | - | 0% | 1min 27s ago | |
| MySpectrumWiFi73-2G | 7C:DB:98:E7:F5... | 11,-1 | 2.4GHz | WPA2 Personal | ASKEY | ac | <div style="width: 100%; background-color: red;">red</div> | <div style="width: 73%; background-color: #ccc;">73</div> | <div style="width: 27%; background-color: #ccc;">27%</div> | -72 | -58 | -75 | -93 | 7% | now | | |
| MySpectrumWiFi73-5G | 7C:DB:98:E7:F5... | 161 | 5GHz | WPA2 Personal | ASKEY | ac | <div style="width: 100%; background-color: red;">red</div> | <div style="width: 79%; background-color: #ccc;">79</div> | <div style="width: 21%; background-color: #ccc;">21%</div> | -77 | -71 | -81 | -94 | 6% | now | | |
| SpectrumSetup-55 | F4:69:42:8A:C7:53 | | 11,-1 | 2.4GHz | WPA2 Personal | ASKEY | ac | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -89 | -88 | -90 | - | 0% | 6h 47min 49s... | |
| Amandas network | F4:69:42:25:BD:45 | | 6,+1 | 2.4GHz | WPA2 Personal | ASKEY | ac | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -91 | -89 | -91 | - | 0% | 6h 48min 3s... | |
| BLINK-76VS | 4C:53:FD:37:EA:78 | | 6 | 2.4GHz | Open | Amazon | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -63 | -63 | -63 | - | 0% | 7h 4min 43s... | |
| NTGR_VMB_9395471872 | A4:11:62:4B:E3:66 | | 8 | 2.4GHz | WPA2 Personal | Arlo | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -92 | -86 | -93 | - | 0% | 6h 51min 7s... | |
| SpectrumSetup-07 | 88:DE:7C:C5:85:... | | 6 | 2.4GHz | WPA2 Personal | Askey | ax | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -88 | -86 | -91 | - | 0% | 46s ago | |
| myBuick | BA:9F:09:44:20:... | | 1 | 2.4GHz | WPA2 Personal | BA:9F:09 | g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -85 | -85 | -85 | - | 0% | 6h 48min 3s... | |
| Hotspot7918 | 00:54:AF:52:79:18 | | 8 | 2.4GHz | WPA/WPA2 Personal | Continental | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -82 | -77 | -88 | - | 0% | 7h 12s ago | |
| Hotspot5631 | 00:54:AF:7B:56:31 | | 4 | 2.4GHz | WPA/WPA2 Personal | Continental | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -93 | -93 | -93 | - | 0% | 7h 18min 12s... | |
| Hotspot2026 | 00:54:AF:72:20:26 | | 7 | 2.4GHz | WPA/WPA2 Personal | Continental | b/g/n | <div style="width: 100%;">-</div> | <div style="width: 0%; background-color: #ccc;">-</div> | 0% | -82 | -82 | -82 | - | 0% | 7h 2min 34s... | |

NetSpot Example Capture:

| | | | | | | | | | | | | | | | |
|--|--------------------|------|--------|--------------------|----------|-------|--|-----|-----|-----|-----|-----|-----|----|-----------------|
| <input type="checkbox"/>  Caballete0071 | EA:9F:80:21:D4:60 | 8 | 2.4GHz | WPA2/WPA3 Personal | EA:9F:80 | ax | <div style="width: 50%;"></div> | - | 0% | -93 | -93 | -93 | - | 0% | 7h 41min 8s... |
| <input type="checkbox"/>  919 wifi | CC:F4:11:77:6D:... | 11 | 2.4GHz | WPA2 Personal | Google | b/g/n | <div style="width: 10%; background-color: red;"></div> | -87 | 13% | -86 | -80 | -91 | -94 | 6% | now |
| <input type="checkbox"/>  919 wifi | CC:F4:11:8F:09:... | 1 | 2.4GHz | WPA2 Personal | Google | b/g/n | <div style="width: 5%; background-color: red;"></div> | -91 | 9% | -89 | -80 | -91 | -93 | 7% | now |
| <input type="checkbox"/>  919 wifi | B0:E4:D5:0A:3... | 1 | 2.4GHz | WPA2 Personal | Google | b/g/n | <div style="width: 10%; background-color: red;"></div> | -86 | 14% | -81 | -70 | -87 | -94 | 6% | now |
| <input type="checkbox"/>  919 wifi | CC:F4:11:8F:09:DD | 149 | 5GHz | WPA2 Personal | Google | ac | <div style="width: 5%;"></div> | - | 0% | -92 | -91 | -94 | - | 0% | 19min 23s a... |
| <input type="checkbox"/>  Private Access V3 | 94:A6:7E:43:C6:7B | 7 | 2.4GHz | WPA2 Personal | NETGEAR | ax | <div style="width: 5%;"></div> | - | 0% | -90 | -85 | -93 | - | 0% | 14s ago |
| <input type="checkbox"/>  Private Access V3 | 3C:37:86:C8:0... | 7 | 2.4GHz | WPA2 Personal | NETGEAR | ax | <div style="width: 10%; background-color: red;"></div> | -87 | 13% | -89 | -84 | -92 | -94 | 6% | now |
| <input type="checkbox"/>  ATTfHV8Cwa_2GEXT | 94:A6:7E:2D:F0:41 | 1 | 2.4GHz | WPA2 Personal | NETGEAR | b/g/n | <div style="width: 5%;"></div> | - | 0% | -92 | -91 | -92 | - | 0% | 6h 55min 51s... |
| <input type="checkbox"/>  NETGEAR93 | 14:59:C0:C1:19:BD | 2 | 2.4GHz | WPA2 Personal | NETGEAR | ac | <div style="width: 5%;"></div> | - | 0% | -92 | -92 | -92 | - | 0% | 7h 41min 50s... |
| <input type="checkbox"/>  SpectrumSetup-FE | 4C:19:5D:3C:86:... | 6 | 2.4GHz | WPA2 Personal | Sagemcom | ax | <div style="width: 5%;"></div> | - | 0% | -82 | -76 | -90 | - | 0% | 14s ago |
| <input type="checkbox"/>  SpectrumSetup-FE | 4C:19:5D:3C:8... | 44 | 5GHz | WPA2 Personal | Sagemcom | ax | <div style="width: 10%; background-color: red;"></div> | -89 | 11% | -88 | -84 | -91 | -91 | 9% | now |
| <input type="checkbox"/>  SpectrumSetup-90 | 4C:19:5D:0E:97:... | 6 | 2.4GHz | WPA2 Personal | Sagemcom | ax | <div style="width: 10%; background-color: red;"></div> | -92 | 8% | -91 | -90 | -94 | -93 | 7% | now |
| <input type="checkbox"/>  SpectrumSetup-A3 | 58:2F:F7:EE:93:A9 | 6 | 2.4GHz | WPA2 Personal | Sagemcom | ax | <div style="width: 5%;"></div> | - | 0% | -92 | -90 | -93 | - | 0% | 7h 5s ago |
| <input type="checkbox"/>  MySpectrumWiFic8-2G | A8:9A:93:9E:2... | 6,-1 | 2.4GHz | WPA2 Personal | Sagemcom | b/g/n | <div style="width: 10%; background-color: red;"></div> | -93 | 7% | -93 | -92 | -94 | -94 | 6% | now |
| <input type="checkbox"/>  dashcam | 58:70:C6:CF:63:B2 | 5 | 2.4GHz | WPA2 Personal | Shanghai | b/g/n | <div style="width: 5%;"></div> | - | 0% | -70 | -70 | -70 | - | 0% | 6h 58min ago |
| <input type="checkbox"/>  tl61974e | 00:25:F0:61:97:4E | 11 | 2.4GHz | WPA2 Personal | Suga | g/n | <div style="width: 5%;"></div> | - | 0% | -92 | -91 | -92 | - | 0% | 7h 29min 1s... |
| <input type="checkbox"/>  TP-Link_73B0 | CC:32:E5:64:73:... | 2 | 2.4GHz | WPA/WPA2 Personal | TP-LINK | b/g/n | <div style="width: 5%;"></div> | - | 0% | -91 | -89 | -94 | - | 0% | 15min 39s a... |
| <input type="checkbox"/>  ARCNETH5G | 68:D7:9A:2A:60:... | 6 | 2.4GHz | WPA2 Personal | Ubiquiti | b/g/n | <div style="width: 5%;"></div> | - | 0% | -86 | -86 | -89 | - | 0% | 9min 53s ago |



NetSpot Surveys

- Wi-Fi surveys are the key feature of NetSpot
 - You run a survey by walking, marking your position on the map, giving NetSpot a few seconds to collect data samples, watching Wi-Fi networks being detected and visualized
 - 15+ heatmap coverage graphs are available with powerful customizable reports

NetSpot Passive Surveys

- Signal-to-noise ratio
- Signal level
- Quantity of access points
- Noise level
- Signal-to-interference ratio
- Frequency band coverage
- PHY mode coverage

NetSpot Active Surveys

- Throughput testing with Iperf 3 or custom speed test servers
- Upload and download speed
- Wireless transmit rate
- Iperf upload, download and jitter

NetSpot Survey Example:

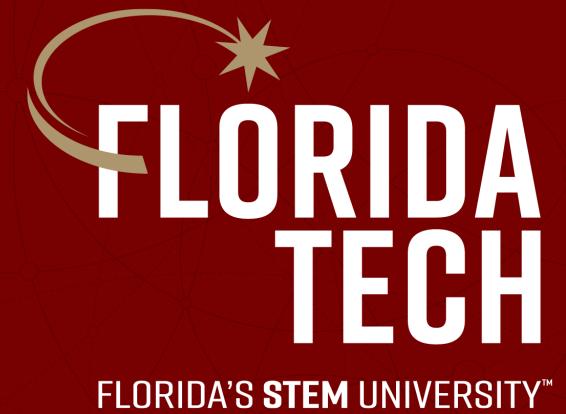


Airodump-ng

- Airodump-ng is used for packet capture, capturing raw 802.11 frames
- It is particularly suitable for collecting WEP IVs (Initialization Vector) or WPA handshakes for the intent of using them with aircrack-ng
- If you have a GPS receiver connected to the computer, airodump-ng is capable of logging the coordinates of the found access points

Kismet

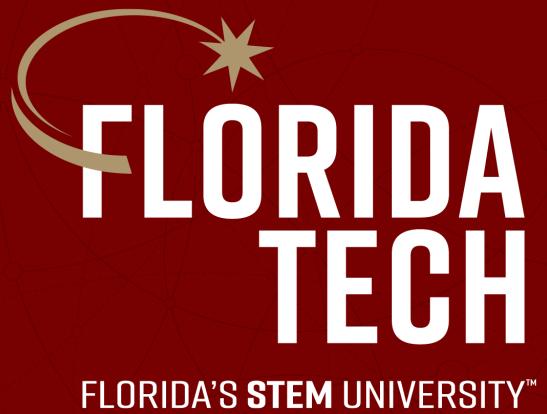
- A wireless network and device detector, sniffer, wardriving tool, and WIDS (wireless intrusion detection) framework
- Kismet works with Wi-Fi interfaces, Bluetooth interfaces, some SDR (software defined radio)
- Kismet works on Linux, OSX, and Windows 10
 - On Linux it works with most Wi-Fi cards, Bluetooth interfaces, and other hardware devices
 - On OSX it works with the built-in Wi-Fi interfaces, and on Windows 10 it will work with remote captures



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

6. WiFi Hacking

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Security through Obscurity

WiFi Di-association / De-authentication

Security Through Obscurity

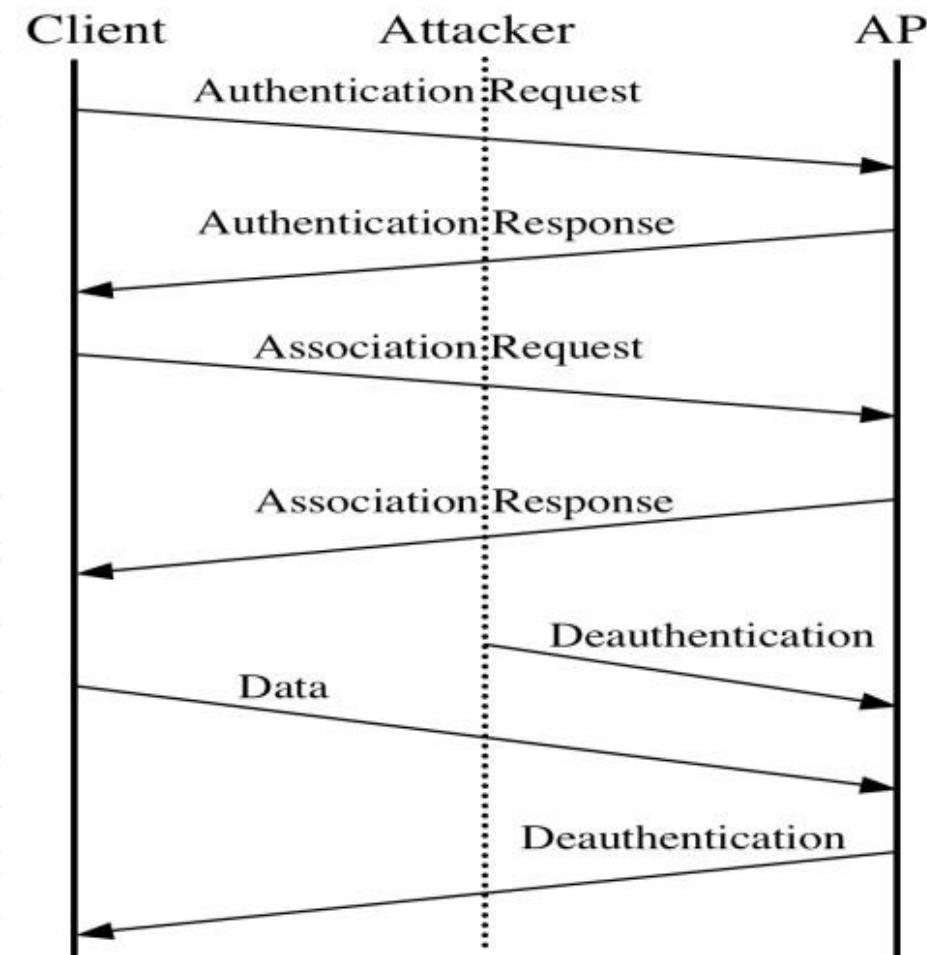
- Wireless network can operate in hidden or non-broadcasting mode
 - They don't include their SSID (network name) in beacon packets, and don't respond to broadcast probe requests
 - SSID is not (cannot be) a secret since it is included in many packets coming from legitimate clients (not just beacon packets)
 - You need to know SSID associated with which AP

Security Through Obscurity

- Passive sniffers can easily take advantage of this behavior
 - If you sniff the network, you will get the SSID whenever someone joins the network
 - You may even force user's hand
 - How?

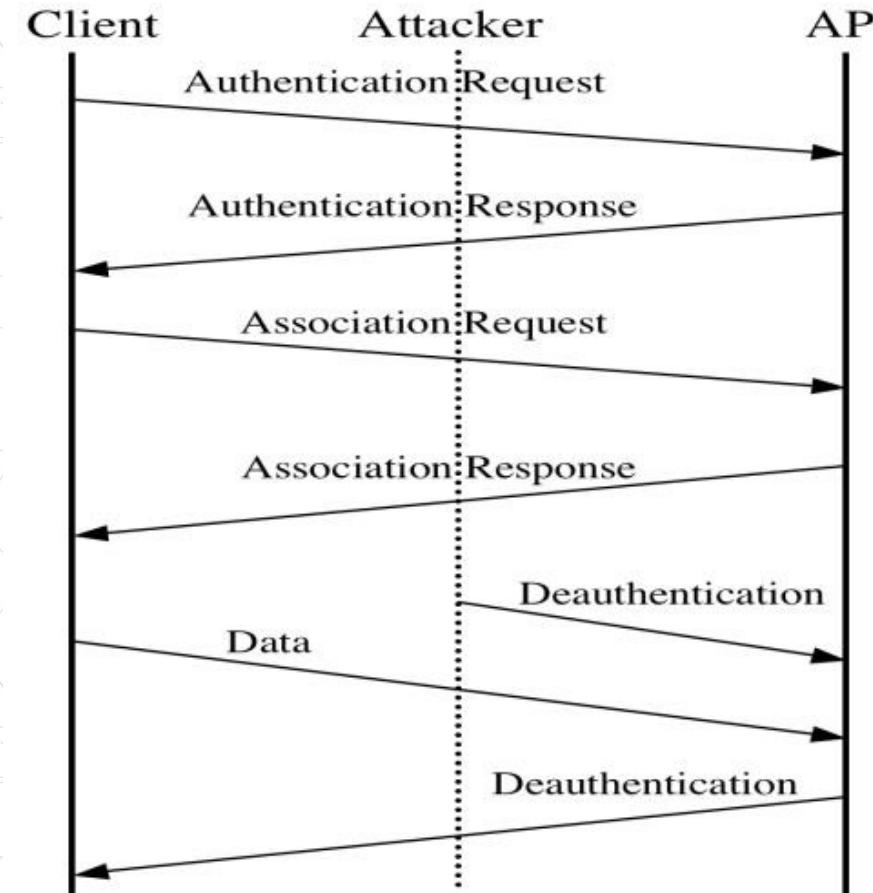
De-authenticating user

- Management frames in 802.11 are not authenticated
 - Send a packet to user that looks like coming from AP
 - The user can't tell the difference
 - Wireless driver will reconnect immediately
 - Reassociation request with the SSID in it will be sent



The De-authentication Frame

- A type of packet defined in the IEEE 802.11 WiFi standard
 - It has been part of the standard since the beginning and still plays an important role
- It's used to terminate a WiFi connection
 - It can be sent by either the AP or the station to let the other side know that the connection is closed

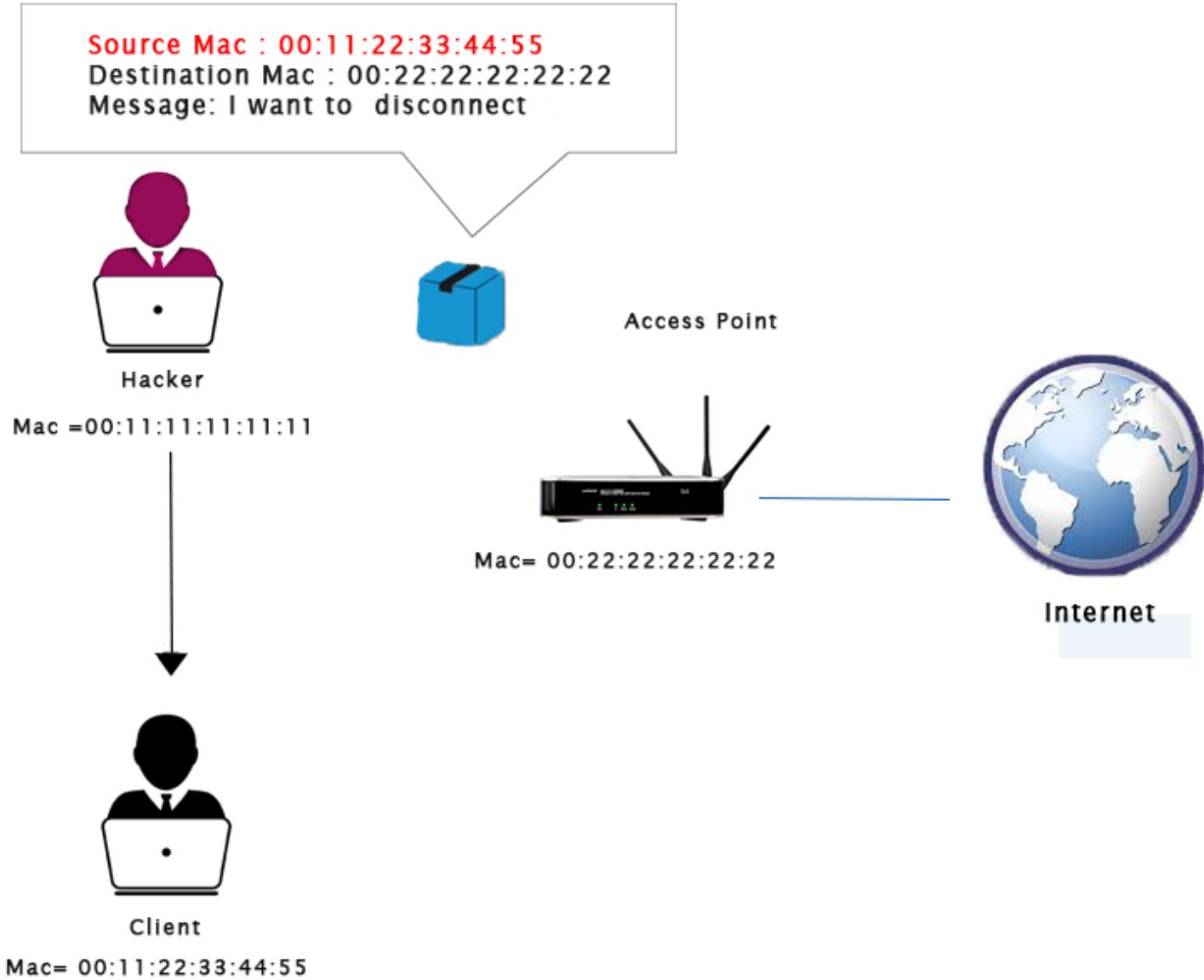


The Deauthentication Frame

- The station might send a deauthentication frame to the access point because it's switching to another WiFi network
 - Or the access point might send a deauthentication frame to the station because the router has to restart
 - Deauthentication works both ways, and there are plenty of reasons why they are sent - you can find a complete list of reasons below

The Deauthentication Frame

- But one crucial attribute of the deauthentication frame is that it's not a request; it's a notification, and it can not be refused



The Deauthentication Frame Example

```
+ Frame 348: 26 bytes on wire (208 bits), 26 bytes captured (208 bits)
+ 802.11 radio information
- IEEE 802.11 Deauthentication, Flags: .....c
  Type/Subtype: Deauthentication (0x000c)
+ Frame Control Field: 0xc000
  .000 0000 0011 0000 = Duration: 48 microseconds
  Receiver address: Cisco_58:e6:1a (00:1b:d4:58:e6:1a)
  Destination address: Cisco_58:e6:1a (00:1b:d4:58:e6:1a)
  Transmitter address: Cisco_af:47:4f (64:a0:e7:af:47:4f)
  Source address: Cisco_af:47:4f (64:a0:e7:af:47:4f)
  BSS Id: Cisco_af:47:4f (64:a0:e7:af:47:4f)
  Fragment number: 0
  Sequence number: 3679
- IEEE 802.11 wireless LAN management frame
  - Fixed parameters (2 bytes)
    Reason code: Unspecified reason (0x0001)
```

Di-association vs. De-authentication

- In the case of a regular home router, you both authenticate and associate to the same AP
 - And if you disconnect, you both deauthenticate and disassociate to the same AP
- But in a larger network made out of multiple APs, you might disassociate from one AP and associate to a new one while staying authenticated to the same network

How can De-auth be Exploited?

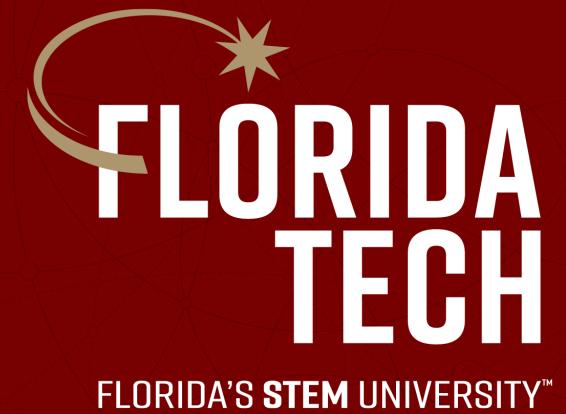
- Deauthentication frames are very simple in their structure
 - You basically only need a sender or receiver MAC address
 - And you can obtain such by simply scanning for WiFi devices nearby
- Thus, it's very easy to spoof a deauth packet
 - And keep in mind that if the target receives it, it has to drop its connection

How can De-auth be Exploited?

- The target can reconnect immediately and it can do that quite fast, maybe without the user noticing that the connection was ever dropped
- But if these deauth packets are sent continuously, it results in a denial of service attack, and network access is blocked for the entirety of the attack
- Luckily this was addressed, and we now have protected management frames!
 - This feature allows packets like deauthentication frames to be safe against spoofing

Protected Management Frames

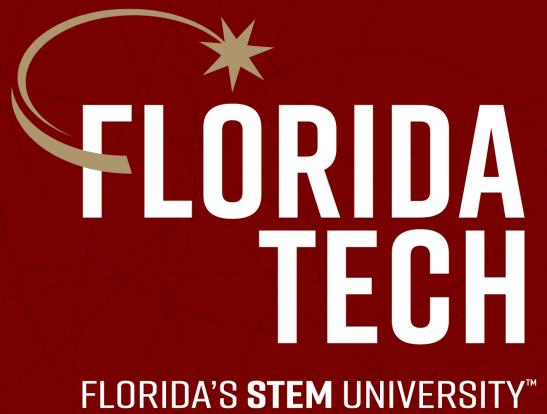
- PMF provide protection for unicast and multicast management action frames
 - Unicast management action frames are protected from both eavesdropping and forging, and multicast management action frames are protected from forging
 - PMF is required for all new certified devices
 - However, may not be implemented in all devices out there



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

7. WEP Vulnerabilities

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

WEP Security Analysis

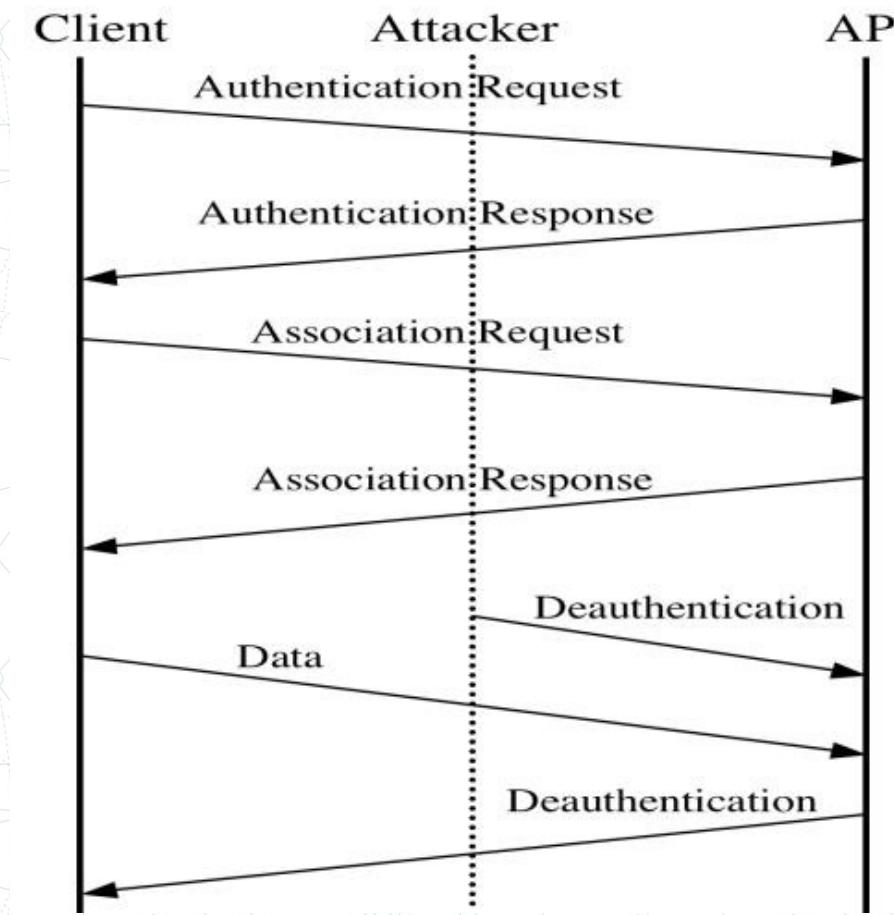
MAC Filtering

Recall: Security Through Obscurity

- Passive sniffers can easily take advantage of this behavior
 - If you sniff the network, you will get the SSID whenever someone joins the network
 - You may even force user's hand
 - How?

Recall: De-authenticating user

- Management frames in 802.11 are not authenticated
 - Send a packet to user that looks like coming from AP
 - The user can't tell the difference
 - Wireless driver will reconnect immediately
 - Reassociation request with the SSID in it will be sent



Recall: Protected Management Frames

- PMF provide protection for unicast and multicast management action frames
 - Unicast management action frames are protected from both eavesdropping and forging, and multicast management action frames are protected from forging
 - PMF is required for all new certified devices
 - However, may not be implemented in all devices out there

Countermeasure for Deauthenticating User

- 802.11w amendment (2012 update) to support protection of both deauthenticate and disassociation frames
 - Using a message integrity check to identify spoofed frames
- Available only when WPA2 is used
- Newer OS support but some APs do not



Review WEP

- Weak Encryption Protocol
 - Authentication
 - Access control
 - Replay prevention
 - Message modification detection
 - Message privacy
 - Key protection

WEP: Authentication

- The basic requirements for authentication in wireless LANs are:
 - Robust method of proving identity that cannot be spoofed
 - Method of preserving identity over subsequent transactions that cannot be transferred
 - Mutual authentication
 - Keys are independent from encryption's (and other purposes) keys

WEP: Authentication

- As a reminder, WEP authentication relies on a challenge-response mechanism
 - First, the AP sends a random string of numbers
 - Second, the mobile device encrypts the string and sends it back
 - Third, the AP decrypts the string and compares to the original string
 - It can then choose to accept the device and send a success message
- The key used for this process is the same WEP key used for encryption, thus breaking rule 4
 - Need independent keys

WEP: Authentication

- The operation does not authenticate the AP to the mobile device because a rogue AP can pretend it was able to check the encrypted string and send a success message without ever knowing the key
 - Hence rule 3 is broken
 - Rule 2 is broken because there is no token provided to validate subsequent transactions, making the whole authentication process rather futile

WEP: Authentication

- During authentication the AP sends a random string of 128 bytes
 - The way in which this "random" string is generated is not defined
 - One would hope at least that it was different for each authentication attempt
- The mobile station encrypts the string and sends it back
- WEP encryption involves generating a sequence of pseudorandom bytes called the key stream and XORing it with the plaintext
- Thus, anyone watching this transaction now has the plaintext challenge and the encrypted response

WEP: Authentication

- Therefore, simply by XORing the two together, the enemy has a copy of the RC4 random bytes
- Authentication: $P \oplus R = C$ (Plaintext XOR Randombytes = Ciphertext)
- And remember that XORing twice gets you back to the original value (that's decryption):
 $\text{If } P \oplus R = C \text{ then } C \oplus R = P$
- By the same argument, XORing the ciphertext with the plaintext gives you the random key stream:
 $\text{If } P \oplus R = C \text{ then } C \oplus P = R$

WEP: Authentication

- The attacker now knows the key stream corresponding to a given IV value
- Now the attacker simply requests authentication, waits for the challenge text, XORs with the previously captured key stream, and returns the result with the previously captured IV
- To check the result, the AP appends the IV (chosen by the attacker) to the secret key and generates the RC4 random key stream

WEP: Authentication

- These will be the same bytes that the attacker worked out because the key and IV are the same as last time
- Therefore, when the access point decrypts the message by XORing with the RC4 key stream, it matches
- The attacker is "authenticated" without ever knowing the secret key

WEP: Authentication

- Although an attacker can get authenticated in this way, can't communicate because frames are encrypted with WEP
 - Therefore, need to break WEP encryption as well
- The enemy needs a sample of matching plaintext and ciphertext
 - The WEP authentication method provides a 128-byte sample free of charge
 - Worse, it is a sample of the first 128 bytes of the key stream, which is the most vulnerable to attack
 - Assists the enemy to attack the encryption keys

Access Control

- Access control is the process of allowing or denying a mobile device to communicate with the network
 - It is often confused with authentication
 - All that authentication does is to establish who you are; it does not follow that, because you are authenticated, you should be allowed access
- In general, access is usually controlled by having a list of allowed devices
 - It may also be done by allowing access to anyone who can prove he has possession of a certificate or some other electronic pass

Access Control

- IEEE 802.11 does not define how access control is implemented
 - However, identification of devices is only done by MAC address, so there is an implication that a list of acceptable MAC addresses exists somewhere
 - Many systems implement a simple scheme whereby a list of allowed MAC addresses can be entered into the access point, even when you are operating without WEP
 - However, given the ease with which MAC addresses can be forged, this cannot be considered as a serious security mechanism

MAC Filtering

- Most APs allow MAC filtering
 - Trusted MAC Addresses to talk to
 - Rest ignored
- To beat it, you steal MAC address from someone already in the network:
 - Run passive scanner to get the address
 - The preferred way is to wait user to disconnect from the network
 - Or you kick them
 - DoS (deauthenticate) attack
 - Attempt to share MAC

MAC Filtering

The screenshot shows the D-Link DIR-808L router's web-based management interface. The top navigation bar includes links for **DIR-808L //**, **SETUP**, **ADVANCED** (which is currently selected), **TOOLS**, **STATUS**, and **SUPPORT**. The left sidebar contains links for **VIRTUAL SERVER**, **PORT FORWARDING**, **APPLICATION RULES**, **QOS ENGINE**, **NETWORK FILTER**, **ACCESS CONTROL**, **WEBSITE FILTER**, **INBOUND FILTER**, **FIREWALL SETTINGS**, **ROUTING**, **ADVANCED WIRELESS**, **WI-FI PROTECTED SETUP**, **ADVANCED NETWORK**, **GUEST ZONE**, **IPV6 FIREWALL**, and **IPV6 ROUTING**.

The main content area has a title **MAC ADDRESS FILTER**. A descriptive text block explains that the MAC (Media Access Controller) Address filter option is used to control network access based on the MAC Address of the network adapter. It states that a MAC address is a unique ID assigned by the manufacturer of the network adapter. This feature can be configured to ALLOW or DENY network/Internet access.

Below this text are two buttons: **Save Settings** and **Don't Save Settings**.

A section titled **24 -- MAC FILTERING RULES** contains the instruction **Configure MAC Filtering below:**. A checkbox labeled **Turn MAC Filtering OFF** is checked. A tooltip for this checkbox provides two options: **Turn MAC Filtering ON and ALLOW computers listed to access the network** and **Turn MAC Filtering ON and DENY computers listed to access the network**.

The main configuration area displays a table with 8 rows, each consisting of a MAC address input field, a double-left arrow button (<<), a "Computer Name" dropdown menu, and a "Clear" button. All rows are currently empty.

Helpful Hints...

Create a list of MAC addresses that you would either like to allow or deny access to your network.

Computers that have obtained an IP address from the router's DHCP server will be in the DHCP Client List. Select a device from the drop down menu, then click the arrow to add that device's MAC address to the list.

Click the **Clear** button to remove the MAC address from the MAC Filtering list.

More...

Beating MAC Filtering

- Change your MAC on Linux:

```
sudo ifconfig wlan0 down
```

```
sudo ifconfig wlan0 hw ether 00:11:22:33:44:55
```

```
sudo ifconfig wlan0 up
```

- IDS (Intrusion Detection System) may detect sharing MAC
 - But can't detect if the attacker waiting for user to disconnect
 - Thus, not much of additional security

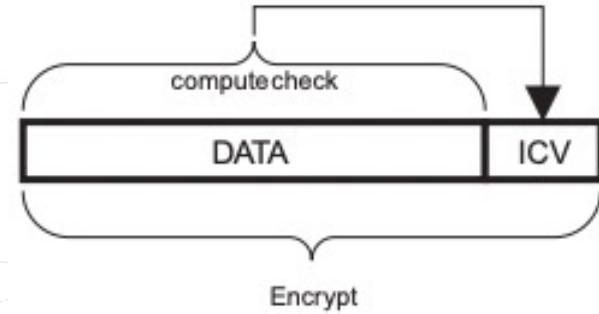
Access Control

- If you can't trust the MAC address, the only thing left to WEP is the encryption key
 - If the mobile station doesn't know the correct WEP key, then the frames it sends will produce an error when decrypted
 - Therefore, the frames will be discarded and, effectively, the device is denied access

Replay Prevention

- WEP has no protection against replay at all
 - It was just not considered in the design
 - There is a sequence number in the MAC frame that must increase monotonically
 - However, it is not included in the WEP protection so it is easy to modify the sequence number to be valid without messing with the encrypted portion on the frame
- Replay protection is not broken in WEP; it simply doesn't exist

Message Modification Detection



- To prevent tampering, WEP includes a checkfield called the integrity check value (ICV)
 - Compute a check value or CRC (cyclic redundancy check) over all the data to be encrypted, append the check value to the end of the data, and then encrypt the whole lot
- If someone changes a bit in the ciphertext, the decrypted data will not have the same check word and the modification will be detected
 - Because the ICV is encrypted, you cannot go back and correct its value to compensate for the other changes you have made
 - It is only intended to provide protection to the ciphertext

Message Modification Detection

- If an attacker already knows the keys, he can modify the data and recompute the ICV before re-encrypting and forwarding the frame
- So use of the ICV protects the ciphertext from tampering?
 - Not quite
- ICV is called a linear method
 - You can predict which bits in the ICV (32-bits) will be changed if you change a single bit in the message

Message Modification Detection

- Let's suppose the message is 8,000 bits (1,000 bytes) and you flip bit position 5244
 - You can then compute which bits in the ICV will be changed as a result
 - It is typically not a single bit but a combination of bits that will change
- You don't need to know the actual value of the plaintext; you just need to know that if you flip the value of a certain bit in the data, you can keep the ICV valid by also flipping a certain combination of its bits
 - Because WEP works by XORing the data to get the ciphertext, bit flipping survives the encryption process
 - Flipping a bit in the plaintext always flips the same bit in the ciphertext, and vice versa

Message Privacy

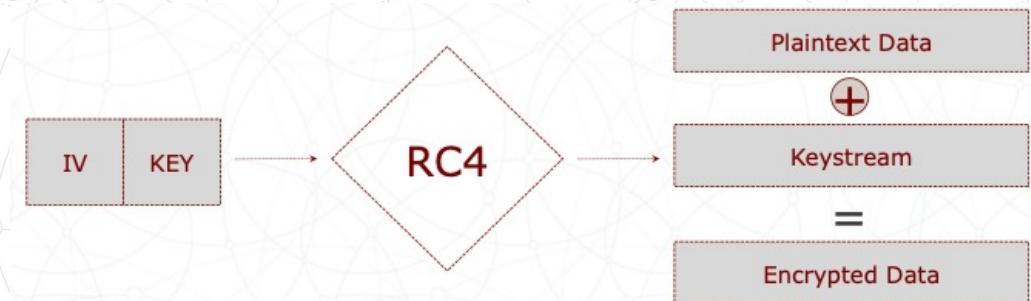
- Attacking the encryption method of WEP
 - If the encryption method holds up, then the attacker is very limited in what he can do
 - So far, it's just watching shadows or throwing rocks at the window; but if the encryption can be breached, the attacker is inside the house
- There are two main objectives in attacking the encryption: decode a message or get the keys
- The ultimate success is to get the keys
 - Once an attacker has the keys, he is free to explore and look for the valuables
- Possession of the keys doesn't automatically mean access to confidential information because there are other layers of security inside, such as server passwords and operating system protections
 - However, the issue of network access is put aside

Message Privacy

- If an attacker can get the keys, he can probably go undetected, which is important to buy the time to find useful information
 - If an attack is detected, the WEP keys can be changed, putting the attacker back to square one
- The next best thing to getting the keys is to be able to get the plaintext
 - If you can get the plaintext in a reasonably fast and reliable way, you have access to a range of other types of attacks using message modification and replay
 - That information can also be used as a stepping-stone to getting the keys
- There are three weaknesses in the way RC4 is used in WEP:
 - IV reuse
 - RC4 weak keys
 - Direct key attack

WEP Keys

- Protects eavesdropping by preventing repetition of RC4 Key



- IV is 24 bits; so total IVs = $2^{24} = 16,777,216$
- Probability of IV Repetition after 5,000 frames = 50 %

IV Reuse

- Instead of using a fixed secret key, the secret key is appended to a 24-bit IV value and then the combined IV / secret is used as the encryption key
- The value of the IV is sent in the frame so the receiving device can perform the decryption
- One purpose of the IV is to ensure that two identical messages don't produce the same ciphertext

IV Reuse

- Let's suppose for a moment that there was no IV and only the secret key is used for encryption
 - For every frame, the RC4 algorithm is initialized with the key value prior to the start of the pseudorandom key stream generation
 - But if the key were to remain fixed, the RC4 algorithm would be initialized to the same state every time
 - Therefore, the key stream produced would be the same sequence of bytes for every frame
 - This is disastrous because, if the attacker can figure out what that key stream is, he can decode every frame simply by XORing the frame with the known sequence
 - He doesn't need to know the key

IV Reuse

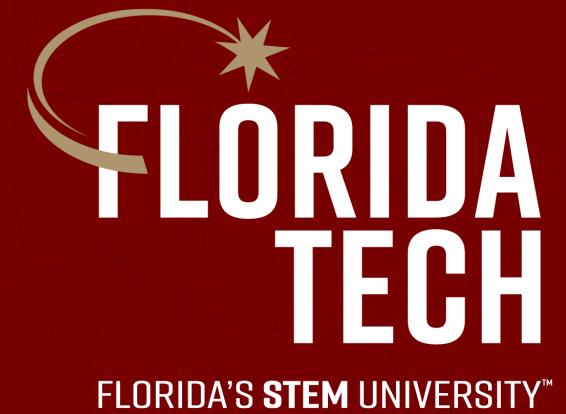
- By adding the IV value to the key each time, RC4 is initialized to a different state for every frame and so the key stream is different for each encryption
 - Let's review that statement because there is an implicit assumption: The IV value is different for every frame
 - If the IV is a constant value, you are no better off than in the static key case
 - The constant IV is useless and using a different IV for every frame is a good idea
 - There are a limited number of possible IVs, so it is acceptable to use a different IV for most frames but eventually start reusing IVs that have been used in the past
 - The simple answer is that this is not acceptable, but it is precisely what WEP does

IV Reuse

- In reality a collision is likely much sooner because there may be many devices transmitting, each incrementing a separate IV value and using it with the same key
- Implementation errors can compound the problem
 - Wi-Fi LAN manufacturer may initialize the IV counter to 0 when the system is started up
- Imagine that ten users come into work and start up their laptops
 - Depending on who does what, the IV counter of some will get ahead of others, but there will be a rich harvest of IV collisions to be had by an observer

Direct Key Attacks

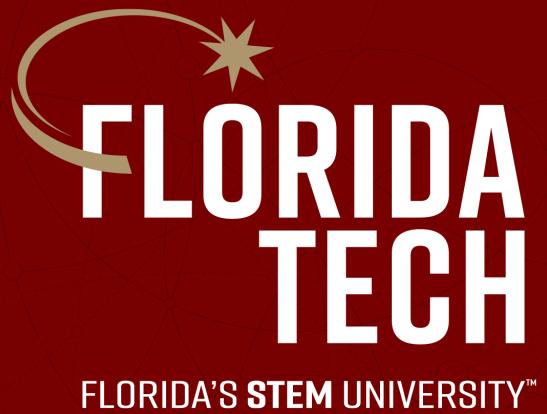
- The method can be tuned to attack each secret key byte in turn so eventually the entire secret key can be extracted
 - Note that increasing the key size from 40 bits to 104 bits means that it takes 2.5 times longer to extract the key (linear, not exponential)
- All the previous weaknesses of WEP pale into insignificance compared to this attack
 - Remember that extracting the keys is the ultimate goal of an attacker, and here is a method that directly extracts the keys in linear time
 - This attack blew apart the remnants of WEP security
 - Because it used a fairly mechanical approach, it was feasible to create a script tool that would do the job unattended
- Within months, some "helpful" person invested their time into generating a cracker tool



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

8. WPA Security

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

WPS

WPA / WPA2 Brute Force

Recall: Review WEP

- Weak Encryption Protocol
 - Authentication
 - Access control
 - Replay prevention
 - Message modification detection
 - Message privacy
 - Key protection

Reaver Brute-force Attack

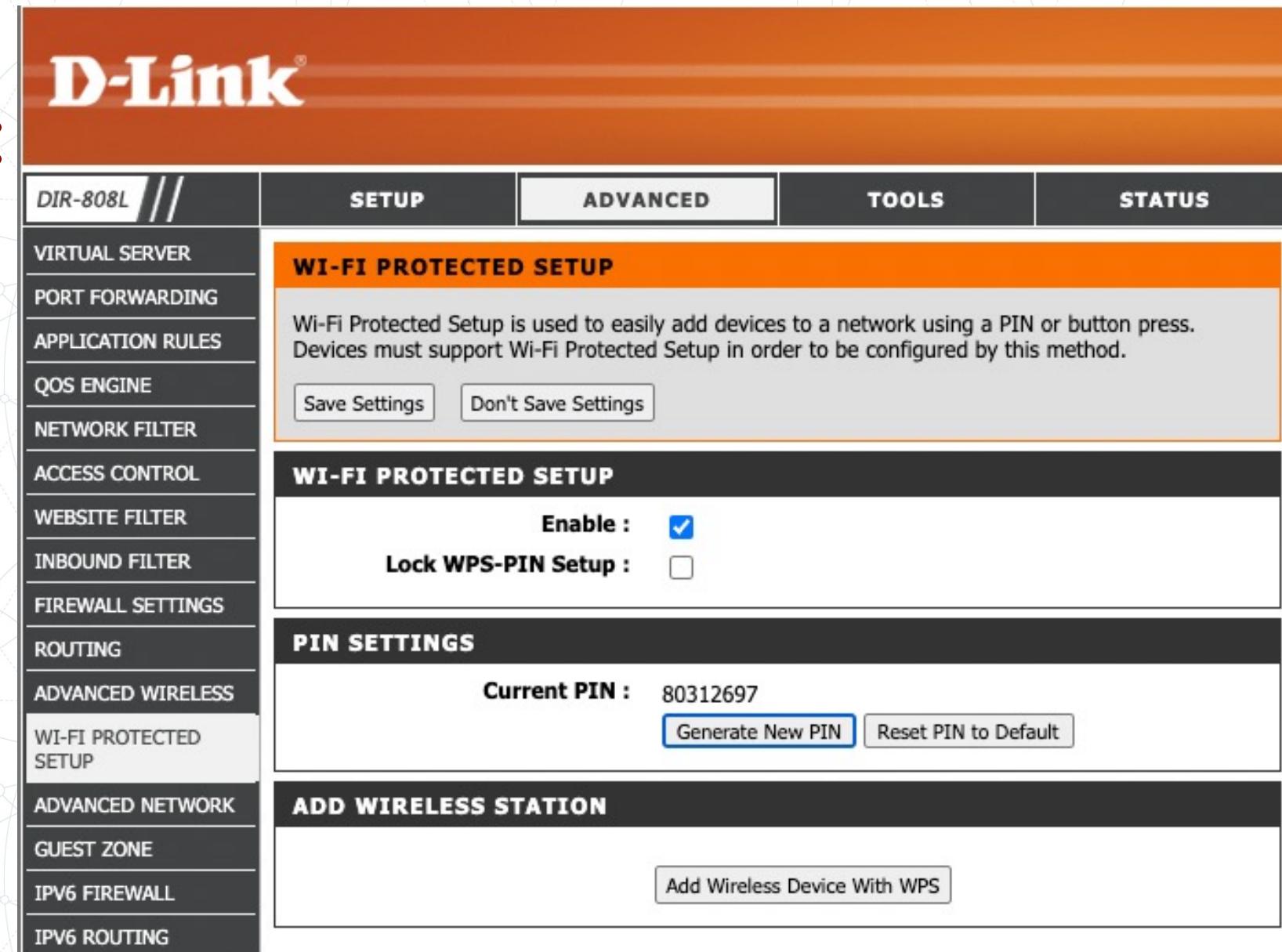
- Was a radical new weapon for Wi-Fi hacking when it was presented in 2011
 - Now obsolete against most routers
- One of the first practical attacks against WPA- and WPA2-encrypted networks, it totally ignored the type of encryption a network used, exploiting poor design choices in the WPS protocol
- Reaver allowed a hacker to sit within range of a network and brute-force the WPS PIN, spilling all the credentials for the router
 - Worse, the 8-digit-long PIN could be guessed in two separate halves, allowing for the attack to take significantly shorter than working against the full length of the PIN

WPS (Wi-Fi Protected Setup)

- It is a network security standard created by the WiFi Alliance in 2006 as an alternative to the regular way of adding devices to the network
 - Instead of requiring the user to insert the SSID (WiFi network name) passkey, it relies on various other methods, such as a PIN, NFC, or Push button to greatly simplify the device pairing process
 - This means that the WPS's reason of existence is simplicity and user friendliness, but it was also a reaction to the independent development of similar solutions by the major manufacturers (the WPS remains non-proprietary and universal)
 - Was created out of necessity, but has been proven to be problematic on the long run

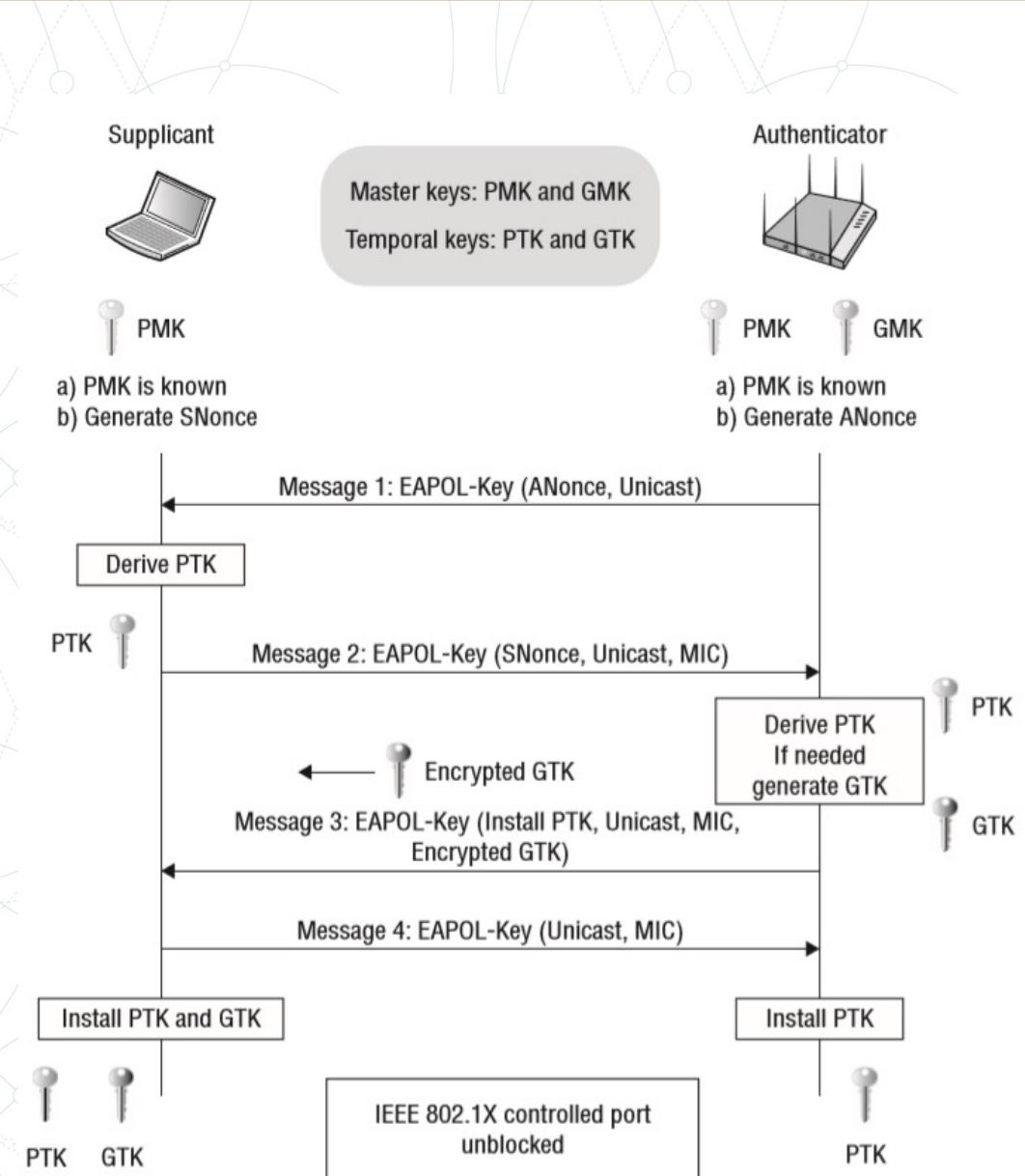
WPS Settings:

- Solution:
 - Disable the WPS
 - To time-out the pairing process in case a brute-force attack was detected



Recall: WPA Handshake

- 4-way handshake
- PTK is derived on run-time
- PMK is pre-installed



WPA Brute Force

- To authenticate with a WPA / WPA2-Personal AP, a station must complete a four-way handshake, securely providing the PSK (Pre-shared Key) without disclosing it in plaintext
- To perform the handshake, both the AP and the station must generate a nonce (a number used only once) to share with one another
 - The AP and the station then both feed the nonce and the pre-shared key (PSK) into a pseudo-random function which generates a pairwise transient key (PTK)
 - The created PTK is unique to both the AP and the station and is used to encrypt communications between the devices, completing the authentication

WPA Brute Force

- However, authentication to an AP is conducted through management frames; meaning, if an attacker can capture the four-way handshake, they will have access to all factors which generated the PTK
 - Isolating the wireless password as the missing variable
- To crack the password, loop this pseudo-random function (with the same nonce's) and a list of possible password as input for the pre-shared key
 - Starting with dictionary list (dictionary attack)
 - Once the transient key generated matches the one from the captured traffic, the password is correct

<https://www.wikihow.com/Hack-WPA/WPA2-Wi-Fi-with-Kali-Linux>

WPA Brute Force

- How much would it take to brute force all?
 - Quite long for regular devices
 - Unless it is in dictionary, your chances are slim
- Even better for attackers is to use “Rainbow Table”
 - Pre-computing hashing in a lookup table
 - As a The ESSID is used as a salt in the encryption process, this speeds up the cracking process for common or reused network names

<http://lastbit.com/pswcalc.asp>

Password length: 8

Speed: 500,000 passwords per second

Number of computers: 1

chars in lower case common punctuation
 chars in upper case full ASCII
 digits

Calculate!

Brute Force Attack will take up to **58 years**
You should have bought a **password manager!** :-)

WPA Brute Force

- No difference between cracking WPA or WPA2 networks
 - The authentication methodology is basically the same between them
 - Thus, the techniques you use are identical
- This can be done either actively or passively
 - “Actively” means you will accelerate the process by deauthenticating an existing wireless client
 - “Passively” means you simply wait for a wireless client to authenticate to the WPA / WPA2 network

Recovering WPA Keys from Clients

- If you have physical access to the device
 - Android, MacOS, Win, etc. all store the WiFi passwords in the file system
 - Mostly do not require root access
 - Not really our concern though

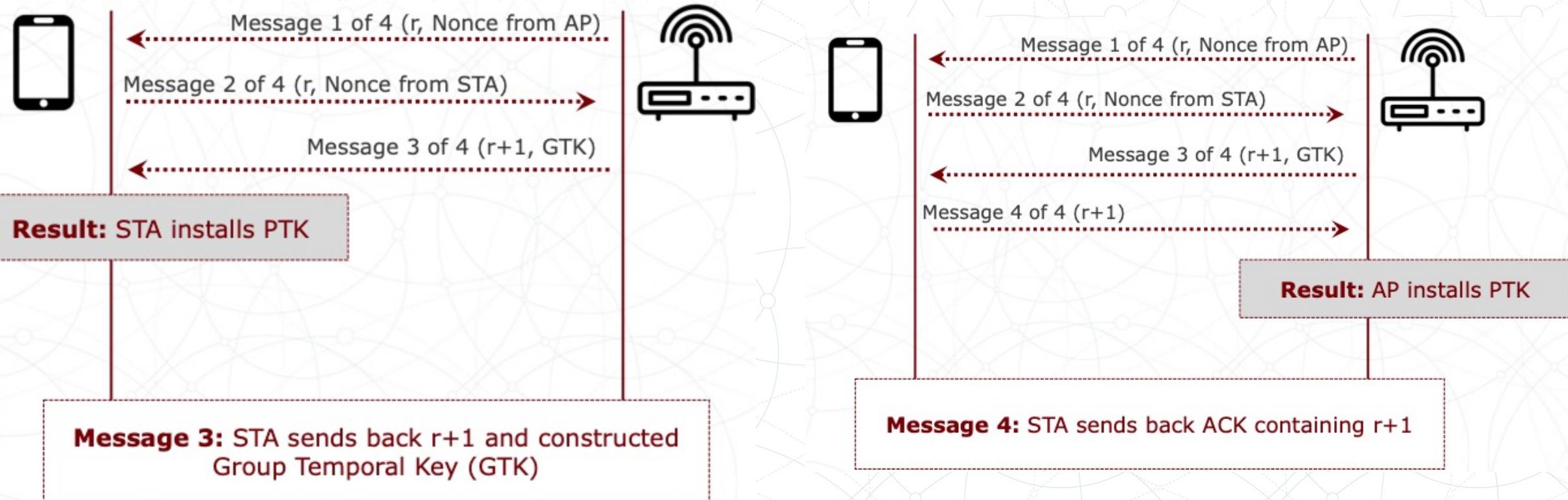
Decrypting the Traffic

- Every user has a unique PTK (pairwise transient key)
- Attacker obtains PMK but not PTK for each user
- Need to capture handshake for that specific user to get PTK
 - Force client to disconnect
 - Then watch / capture re-connection

KRACK (Key Reinstallation attack)

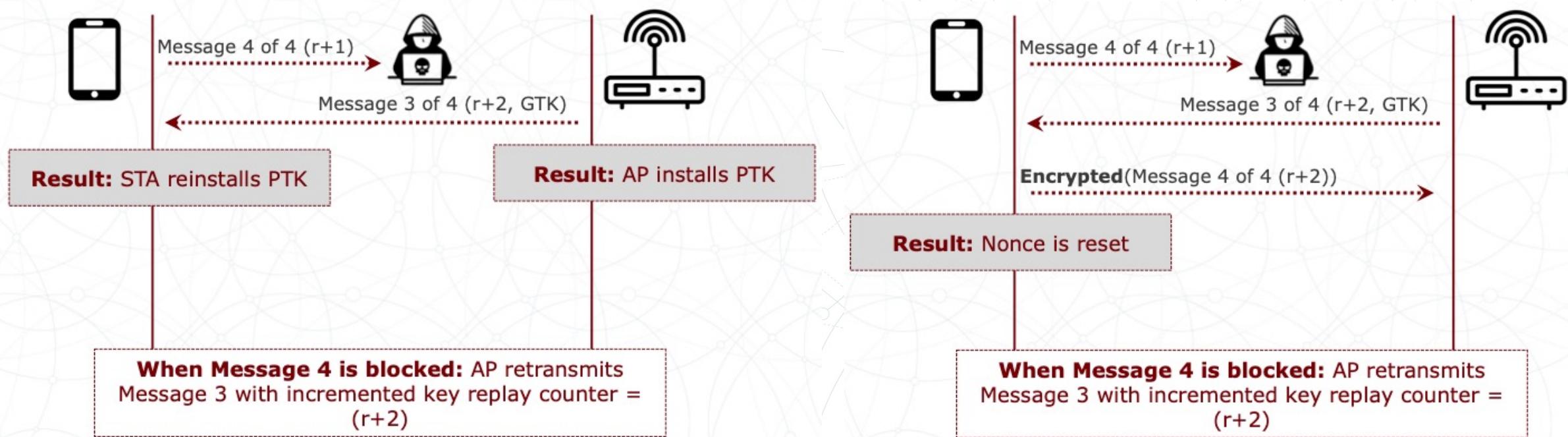
- High Level Idea: attack targets WPA2 four-way handshake protocol flaw specification that allows for third message to be sent repeatedly without changing encryption key
- Discovered in 2016;
 - 14 years after WPA2 was certified by WiFi Alliance
- Attack Discussed in: Vanhoef, Mathy, and Frank Piessens. "Key reinstallation attacks: Forcing nonce reuse in WPA2." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

KRACK (Key Reinstallation attack)



KRACK (Key Reinstallation attack)

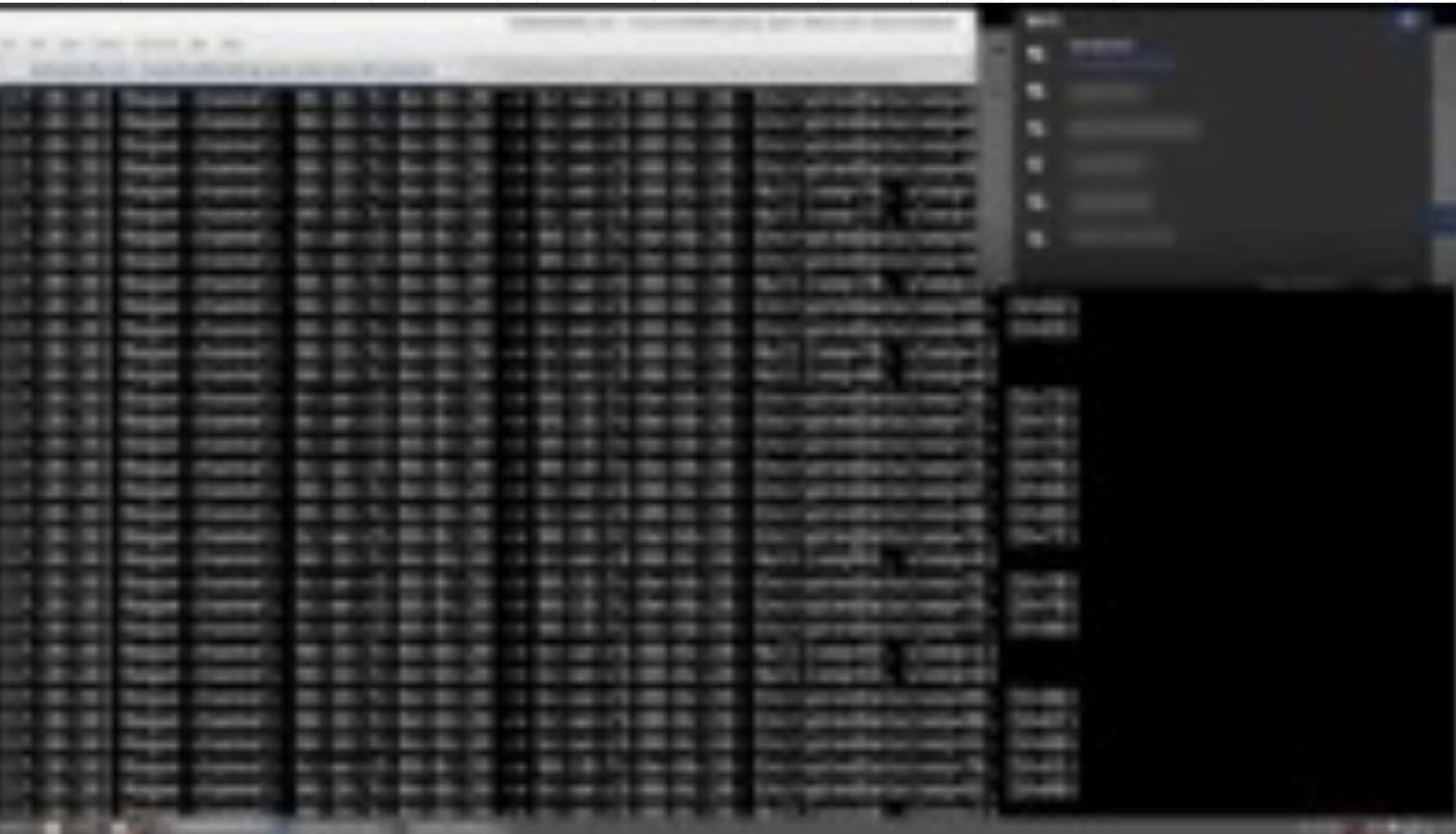
- Attacker blocks message 4
 - Re-transmit message 3 with increasing 'r'



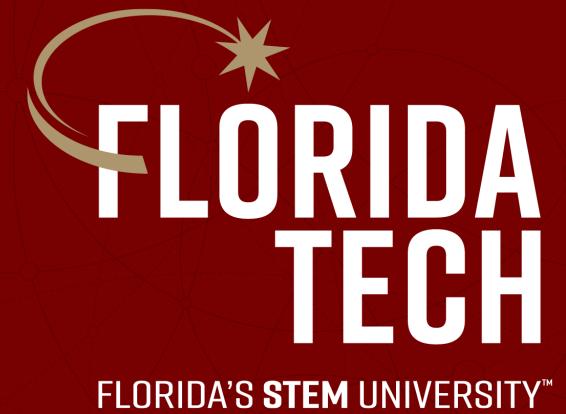
KRACK: Example Scenario

- KRACK allow an adversary to decrypt a TCP packet, learn the sequence number, and hijack the TCP stream to inject arbitrary data
 - Without knowing the password of WiFi
 - This enables one of the most common attacks over Wi-Fi networks: injecting malicious data into an unencrypted HTTP connection

KRACK: Explained



<https://www.youtube.com/watch?v=Oh4WURZoR98&t=1s>



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

9. Bluetooth Explained

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Bluetooth

Basics

Standards

Device Discovery

Connection Establishment

Protocol Overview

Recall: Reaver Brute-force Attack

- Was a radical new weapon for Wi-Fi hacking when it was presented in 2011
 - Now obsolete against most routers
- One of the first practical attacks against WPA- and WPA2-encrypted networks, it totally ignored the type of encryption a network used, exploiting poor design choices in the WPS protocol
- Reaver allowed a hacker to sit within range of a network and brute-force the WPS PIN, spilling all the credentials for the router
 - Worse, the 8-digit-long PIN could be guessed in two separate halves, allowing for the attack to take significantly shorter than working against the full length of the PIN

Recall: WPA Brute Force

- However, authentication to an AP is conducted through management frames; meaning, if an attacker can capture the four-way handshake, they will have access to all factors which generated the PTK
 - Isolating the wireless password as the missing variable
- To crack the password, loop this pseudo-random function (with the same nonce's) and a list of possible password as input for the pre-shared key
 - Starting with dictionary list (dictionary attack)
 - Once the transient key generated matches the one from the captured traffic, the password is correct

<https://www.wikihow.com/Hack-WPA/WPA2-Wi-Fi-with-Kali-Linux>

Recall: WPA Brute Force

- How much would it take to brute force all?
 - Quite long for regular devices
 - Unless it is in dictionary, your chances are slim
- Even better for attackers is to use “Rainbow Table”
 - Pre-computing hashing in a lookup table
 - As a The ESSID is used as a salt in the encryption process, this speeds up the cracking process for common or reused network names

<http://lastbit.com/pswcalc.asp>

The screenshot shows a web-based password cracking calculator. The input fields are set to: Password length: 8, Speed: 500,000 passwords per second, and Number of computers: 1. Under 'Character Set' checkboxes, 'chars in lower case', 'chars in upper case', and 'digits' are checked, while 'common punctuation' and 'full ASCII' are unchecked. A 'Calculate!' button is present. Below the input fields, a yellow banner displays the calculated result: 'Brute Force Attack will take up to **58 years**' and 'You should have bought a **password manager!** :-)'.

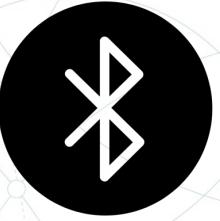
Recall: Decrypting the Traffic

- Every user has a unique PTK (pairwise transient key)
- Attacker obtains PMK but not PTK for each user
- Need to capture handshake for that specific user to get PTK
 - Force client to disconnect
 - Then watch / capture re-connection

Recall: KRACK: Example Scenario

- KRACK allow an adversary to decrypt a TCP packet, learn the sequence number, and hijack the TCP stream to inject arbitrary data
 - Without knowing the password of WiFi
 - This enables one of the most common attacks over Wi-Fi networks: injecting malicious data into an unencrypted HTTP connection

Bluetooth Basics



- Short-range, personal-area network protocol used for cable replacement (headphones, computer peripherals, IoT devices)
- Managed by the Bluetooth Special Interest Group (SIG)
- Standardized in IEEE 802.15.1 protocol
- Bluetooth 1 (1998)
 - Original Specification 802.15.1-2002

Bluetooth Basics

- Defines 79 channels across the 2.4-GHz ISM band, each channel occupying 1-MHz of spectrum
- Devices hop across these channels at a rate of 1600 times a second (every 625 microseconds)
- This channel-hopping technique is Frequency Hopping Spread Spectrum (FHSS), and the user can achieve a rate of 3 Mbps of bandwidth across 100 meters
 - FHSS provides robustness against noisy channels by rapidly changing frequencies
 - Later revisions of the standard have added support for adaptive hopping, which allows noisy channels to be detected and avoided all together

Device 1 and 2 form a piconet; they are channel hopping in step with each other.

Device 1 (master)

| | | | | | | | | | | | |
|---|---|---|---|---|---|----|---|---|----|---|----|
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |

Device 2 (slave)

Device 3

| | | | | | | | | | | | |
|---|---|---|----|---|---|---|---|----|---|---|---|
| 6 | 4 | 5 | 10 | 1 | 2 | 6 | 3 | 11 | 8 | 9 | 7 |
|---|---|---|----|---|---|---|---|----|---|---|---|

Device 3 is not part of the piconet; it is unaware of the channel-hopping sequence in use by the other devices.

Bluetooth Standards

- Bluetooth 2 (2005)
 - Added Enhanced Data Rate (EDR) to 3Mb/s
 - Added reduced power consumption
- Bluetooth 3 (2009)
 - Added AMP (Alternative MAC/PHY)
 - Offered Optional high speed transp. (HS)
- Bluetooth 4 (2010)
 - Bluetooth Low Energy (LE)
 - Added Power consumption for low cost/small size
- Bluetooth 5 (2016)
 - Added new functionality for IoT
 - Added Asynchronous Connection Less services

Bluetooth Devices

- Every device implementing Bluetooth has a high resolution 24-bit clock (referred to as CLKN in the specification)
 - This clock is used to keep the frequency hopping synchronized, as well as schedule other events
- In order to participate in a piconet, the piconet master's BD_ADDR (a 48-bit MAC address) and clock must be known
 - Bluetooth device clocks increment at a rate of one every 312.5 microseconds

Bluetooth: Device Discovery

- Assume that a device is already interacting in a piconet (hopping along with its peers) and that it is also discoverable
 - Which means that it wants to be found by other devices not already in its piconet
 - Yet, that it must be able to temporarily quit hopping along with its piconet peers;
 - Listen for any devices that are potentially looking for it,
 - Respond to those requests, and
 - Catch back up with the other devices in the piconet

Device 1 and 2 form a piconet; they are channel hopping in step with each other.

Device 1 (master)

| | | | | | | | | | | | |
|---|---|---|---|---|---|----|---|---|----|---|----|
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |

Device 2 (slave)

Device 3 is not part of the piconet; it is unaware of the channel-hopping sequence in use by the other devices.

Device 3

| | | | | | | | | | | | |
|---|---|---|----|---|---|---|---|----|---|---|---|
| 6 | 4 | 5 | 10 | 1 | 2 | 6 | 3 | 11 | 8 | 9 | 7 |
|---|---|---|----|---|---|---|---|----|---|---|---|

Bluetooth: Device Discovery

- Devices that periodically check for other devices looking for them are said to be “discoverable”
 - Many devices aren’t discoverable by default and must have this feature specifically enabled, usually for a brief period of time



Bluetooth: Device Discovery

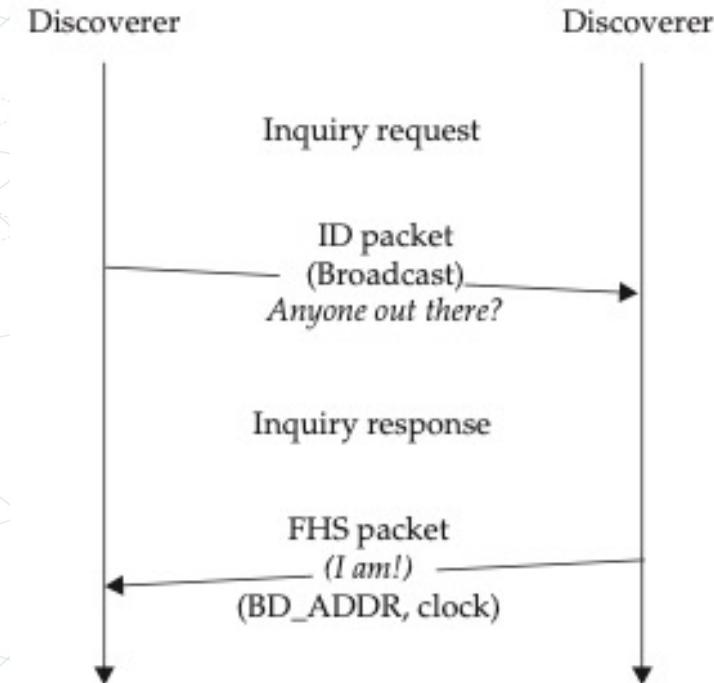
- Technically, discoverable devices are devices that enter the inquiry scan substate
 - These devices respond to inquiry requests
- On the other end of this frequency-hopping dance is the device doing the discovery
 - This device has no knowledge of its potential peer channels at the moment
 - Thus, it must transmit discovery requests (ID packets) into the air in a (mostly) random pattern, hoping to cross paths with a device on the same channel at the same time

Bluetooth: Device Discovery

- Even assuming that the discoverable device sees this request, how is it supposed to respond?
 - It needs to transmit a response, but can't be sure what channel its discovering buddy wandered off to
 - Therefore, it will start responding on a lot of channels, on the assumption that its discovering device will see one of the responses
- The protocol has a few optimizations to help devices find each other, and there is an upper-bound on the time it takes for this entire exchange to happen (10.24 seconds), but the process still seems remarkably difficult
 - If you've ever wondered what your computer was doing when it was looking for your cell phone or Bluetooth mouse the first time, this is it

Bluetooth: Device Discovery

- A device is said to be nondiscoverable if it simply ignores (or doesn't look for) inquiry requests
 - The only way to establish a connection to one of these nondiscoverable devices is to determine its Bluetooth device address (BD_ADDR) through some other means
 - Once the discoverer has the BD_ADDR and clock of the discoveree, it can then attempt to initiate a connection

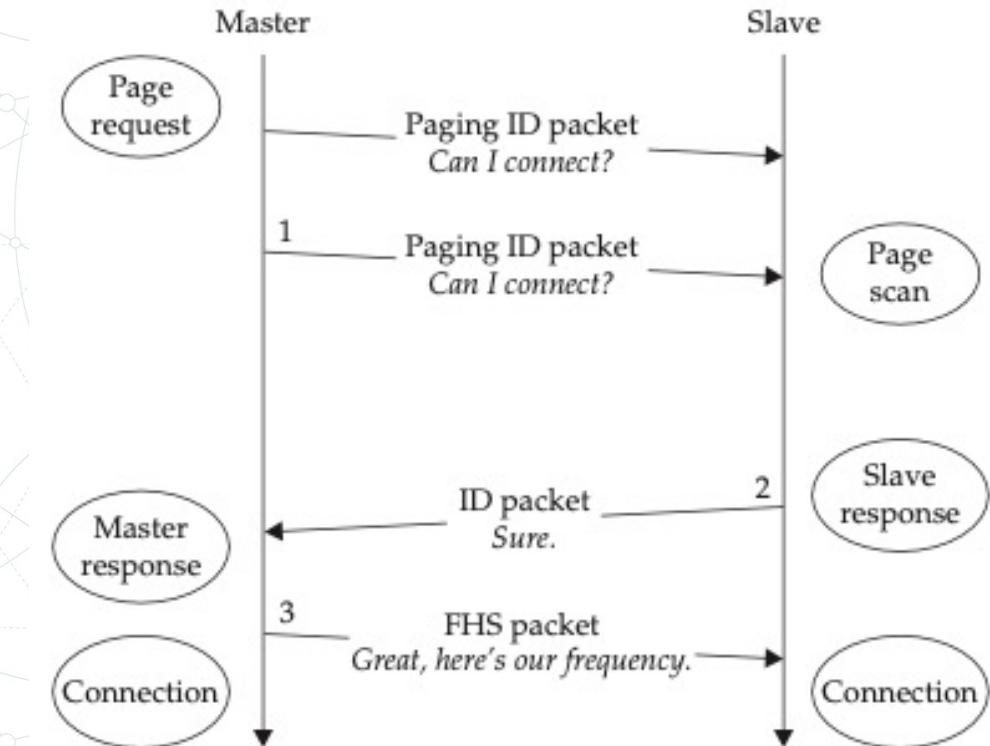


Bluetooth: Connection Establishment

- When a device wishes to establish a connection to another device, it must “page” it
 - This consists of transmitting a page request on the channel it thinks the target device is currently on
 - The transmitting device may not know the target channel for a number of reasons that include power savings, clock drift due to too much time passing since the last communications, and so on
- Devices that accept connection requests (pages) are said to be in page-scan mode, because they will periodically pause their current operation (such as relaying a real-time audio stream) to check to see if any other devices are interested in talking to them

Bluetooth: Connection Establishment

- The diagram covers this in some detail
 - The most important thing to remember about “paging” or connection establishment is that in order to establish a connection you must know the target’s BD_ADDR, and that device must be interested in accepting connections

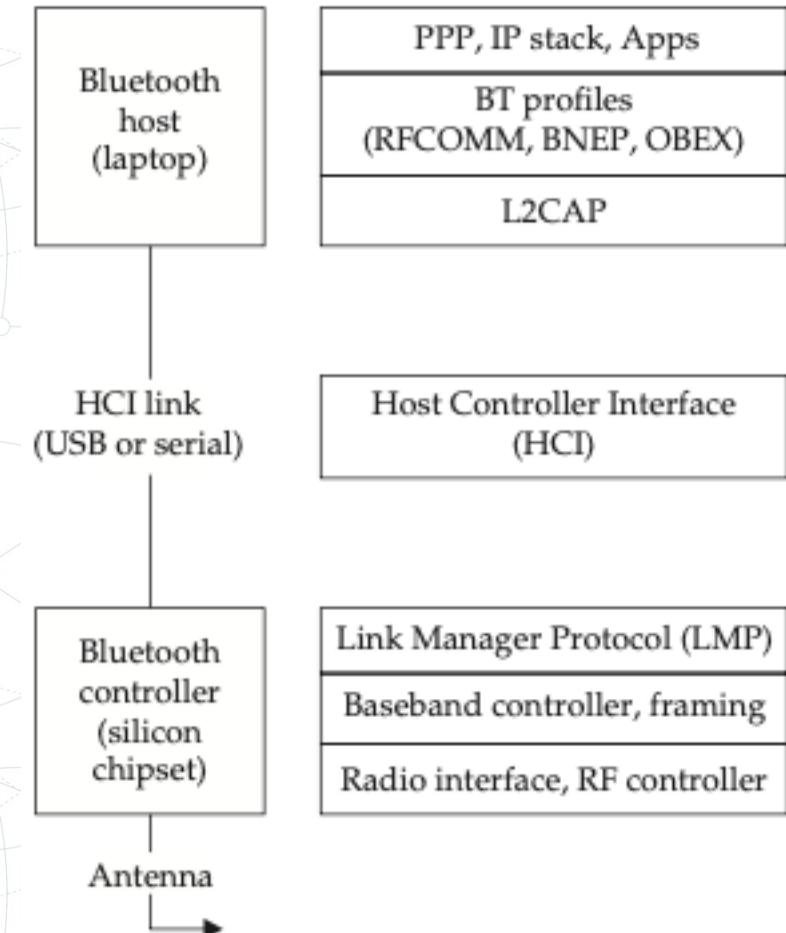


Bluetooth Protocol Overview

- A number of protocols are used within a Bluetooth network
 - They can generally be broken up into two classes: those spoken by the Bluetooth controller and those spoken by the Bluetooth host
 - For the sake of our class, the Bluetooth host is the laptop that you are trying to run attacks from
 - The Bluetooth controller is sitting on the other end of your USB port, interpreting commands from the host

Bluetooth Protocol Overview

- The organization of layers in the Bluetooth stack and where each layer is typically implemented:
 - The controller is responsible for frequency hopping, baseband encapsulation, and returning the appropriate results back to the host
 - The host is responsible for higher-layer protocols
 - The Host Controller Interface (HCI) link is used as the interface between the Bluetooth host (your laptop) and the Bluetooth controller (the chipset in your Bluetooth dongle)



Bluetooth Protocol Overview

- When dealing with Bluetooth, keep this host/controller model in mind
- As hackers, the thing we most desire over a device is control
 - The separation of power in the model means that we are very much at the mercy of the Bluetooth controller
 - No matter how much we want to tell the Bluetooth controller “Stick to channel 6 and blast the following packet out forever,” unless we can map this request into a series of HCI requests (or find some other way to do it), we can’t
 - We just don’t have that much control over the radio

Bluetooth Protocol Stack

- RFCOMM: transport protocol emulates serial over BT (uses such as file transfer) [Like TCP]
- L2CAP: datagram based transport protocol for message-based, unreliable [Like UDP]
- HCI: specs for communicating between chipset and host software
- LMP: handles negotiation, encryption, authentication, and pairing
- BASEBAND: handles over-the-air characteristics (transmission rate, channel)

Radio Frequency Communications (RFCOMM)

- RFCOMM is the transport protocol used by Bluetooth devices that need reliable streams- based transport, analogous to TCP
 - The RFCOMM protocol is commonly used to emulate serial ports, send commands to phones, and to transport files over the Object Exchange (OBEX) protocol
- Similar to TCP, RFCOMM has the notion of ports
 - Instead of 65,536 ports, however, RFCOMM has ports 1 to 30
 - In RFCOMM terminology, these ports are called channels

Radio Frequency Communications (RFCOMM)

- RFCOMM is the simplest of the Bluetooth protocols to wrap your head around
 - It is also the highest level and most universally available to developers on restrictive platforms, such as mobile phones
 - RFCOMM is implemented on top of the L2CAP protocol

Logical Link Control and Adaptation Protocol (L2CAP)

- L2CAP is a datagram-based protocol, which is used mostly as a transport to higher-layer protocols such as RFCOMM and others
 - An application-level programmer can use L2CAP as a transport as well, and when used in this case, L2CAP has semantics similar to that of UDP (messaged based, not reliable, etc.)
- L2CAP has a set of ports (independent from RFCOMM ports) and all ports are odd
 - Ports in the range 1–4,095 are reserved / well-known, applications between 4,097 and 32,765
- Think of L2CAP as straddling the line between IP and UDP
 - Usually L2CAP is used to carry higher-level data packets; however, on some platforms, an application programmer can make use of it directly

Host Controller Interface (HCI)

- HCI is a protocol that has no analogy in an 802.11 or Ethernet-based network
- As mentioned previously, the Bluetooth standard specifies an interface for controlling a Bluetooth chipset (controller)
 - HCI is this interface
 - This technique means that much of the userland tools related to managing Bluetooth connections need no modification at all, even when a completely different Bluetooth chipset (controller) is used

Controller Protocol Stack

- The following protocols are handled by the Bluetooth controller (chipset)
 - Asynchronous Connectionless Link (ACL)
 - Synchronous Connection Oriented (SCO)
 - Link Manager Protocol (LMP)
- Unless utilizing specialized hardware, manipulation of these low-level protocols is outside the capability of users

Baseband

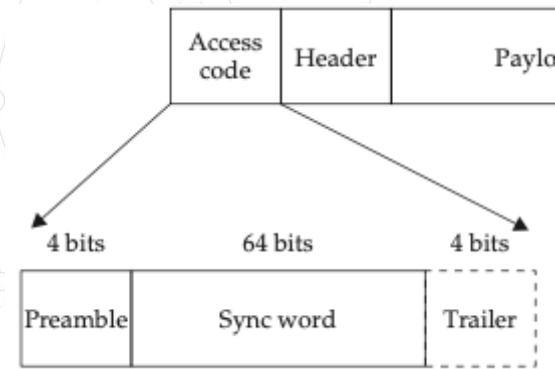
- The Bluetooth baseband specifies the over-the-air characteristics (such as the transmission rate) and the final layer of framing for a packet
- Unlike 802.11, where receiving all the packets on a channel is trivial, actually getting a packet with in-tact baseband headers out of the controller and into the host is difficult
 - Even more difficult is handing the controller an arbitrary buffer and having it push this out to the air as a packet

Baseband



- The organization of every Bluetooth packet at the lowest level:
- Access Codes
 - The first field of every Bluetooth packet is the access code
 - When a Bluetooth controller receives a packet, the first thing it does is examine the access code to determine what to do

Baseband: Access Codes



- The bulk of an access code is taken up by a 64-bit sync word
 - This sync word is key to understanding how Bluetooth device addresses are used to establish a connection within a piconet
- The sync word is a 64-bit expansion of the lower 24-bits of the BD_ADDR that a device wishes to communicate with
 - Conceptually, you can think of the sync word expansion function as a hash, which has the very simple job of mapping 24 bits into a 64-bit space, although it is not designed to be cryptographically hard to reverse

Baseband: Access Codes

- At any given point in time, a Bluetooth controller will be interested in only a handful of sync words
 - Any packets received by the controller with sync words that aren't interesting won't be passed through the HCI link
 - These packets are assumed to be for another piconet
 - At any given time, a particular Bluetooth controller will concern itself with three different types of sync words

Baseband: Access Codes

- First sync word that corresponds with the local device's own BD_ADDR
 - Access codes of this type are called DACs and are used to handle paging requests
- Derived From the BD_ADDR of the piconet's master: CACs
 - Packets with a CAC are used to carry application-level data, and the Bluetooth controller will need to examine the Logical Transport Address (LT_ADDR, the piconet-specific address) field of the header to determine if this packet is meant for the recipient, requiring further processing
- IAC: used to indicate that a device is trying to discover other devices

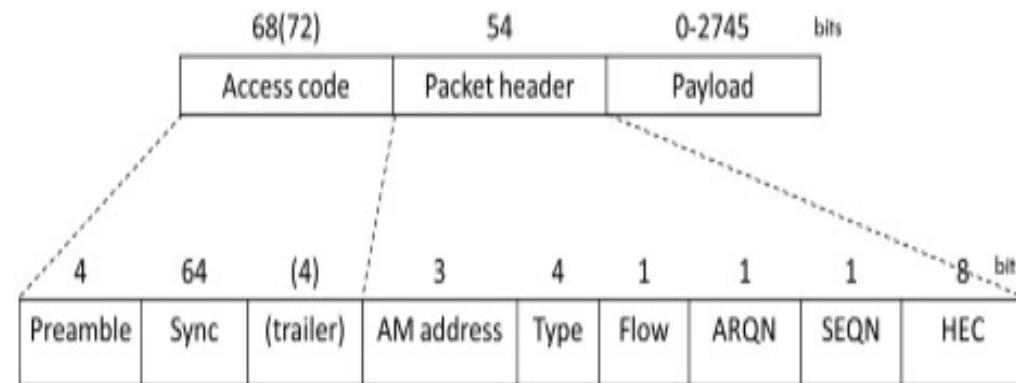
| Sync Word Derived From | Used For | Name Given |
|-------------------------------|--|---------------------------|
| Destination BD_ADDR | Channel signaling (paging requests) | Device Access Code (DAC) |
| Master's BD_ADDR | Data transport | Channel Access Code (CAC) |
| Reserved 0x9E8B00-0x9E8B3F | Inquiry (Device Discovery) | Inquiry Access Code (IAC) |

Header Field

- Most of these fields aren't of concern to us

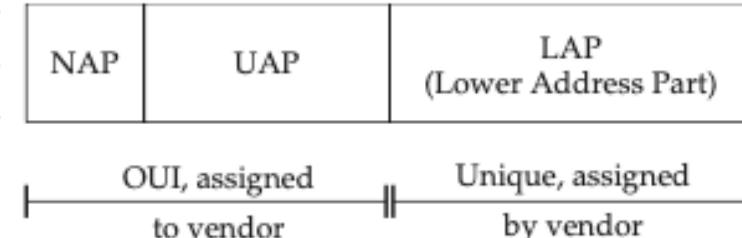
unless we are implementing our own Bluetooth controller in software:

- AM Address (or LT_ADDR); Logical Transport Address
- Type; the type of packet being used, indicating the data type (ACL or SCO)
- Flow; a simple flow-control feature
 - When set to 1 (known as GO), the receiver has sufficient buffering space; 0 implies the opposite
- ARQN or sequence bit (SEQN); for positive acknowledgment of packet delivery and sequence numbering
- HEC Header Error Check; An integrity check is performed over the entire packet

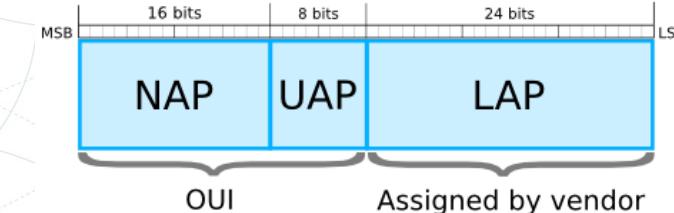


Bluetooth Device Addresses (BD_ADDR)

- Bluetooth devices come with a 6-byte 802-compliant MAC address, similar to that of Ethernet and 802.11 devices
- In Bluetooth, these devices have a little more structure to them and are rarely transmitted over the air
 - As outlined previously, the lower 24 bits of a BD_ADDR is expanded into a 64-bit sync word, which is, in turn, transmitted in the access code of a Bluetooth baseband packet

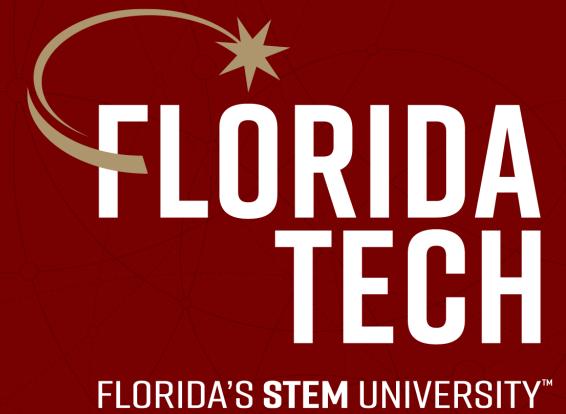


11:22:33:44:55:66



BD_ADDR

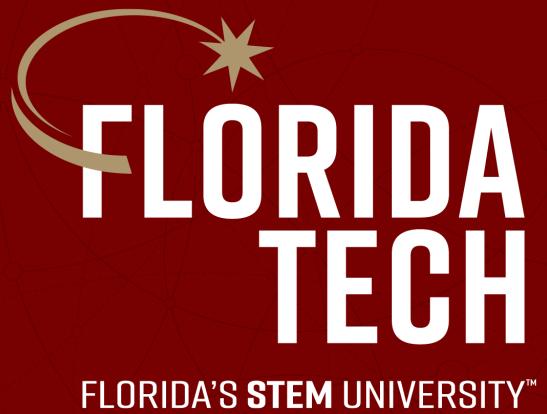
- A BD_ADDR is composed of three distinct parts
 - NAP; The Nonsignificant Address Part consists of the first 16 bits of the OUI (organizationally unique identifier) portion of the BD_ADDR
 - This part is called nonsignificant because these 16 bits are not used for any frequency hopping or other Bluetooth derivation functions
 - UAP; The Upper Address Part composes the last 8 bits of the OUI in the BD_ADDR
 - LAP; The Lower Address Part is 24 bits and is used to uniquely identify a Bluetooth device



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

10. Bluetooth Security

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Bluetooth

Encryption and Authentication

Traditional Pairing

Secure Simple Pairing

Recall: Bluetooth Basics

- Defines 79 channels across the 2.4-GHz ISM band, each channel occupying 1-MHz of spectrum
- Devices hop across these channels at a rate of 1600 times a second (every 625 microseconds)
- This channel-hopping technique is Frequency Hopping Spread Spectrum (FHSS), and the user can achieve a rate of 3 Mbps of bandwidth across 100 meters
 - FHSS provides robustness against noisy channels by rapidly changing frequencies
 - Later revisions of the standard have added support for adaptive hopping, which allows noisy channels to be detected and avoided all together

Device 1 and 2 form a piconet; they are channel hopping in step with each other.

Device 1 (master)

| | | | | | | | | | | | |
|---|---|---|---|---|---|----|---|---|----|---|----|
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |

Device 2 (slave)

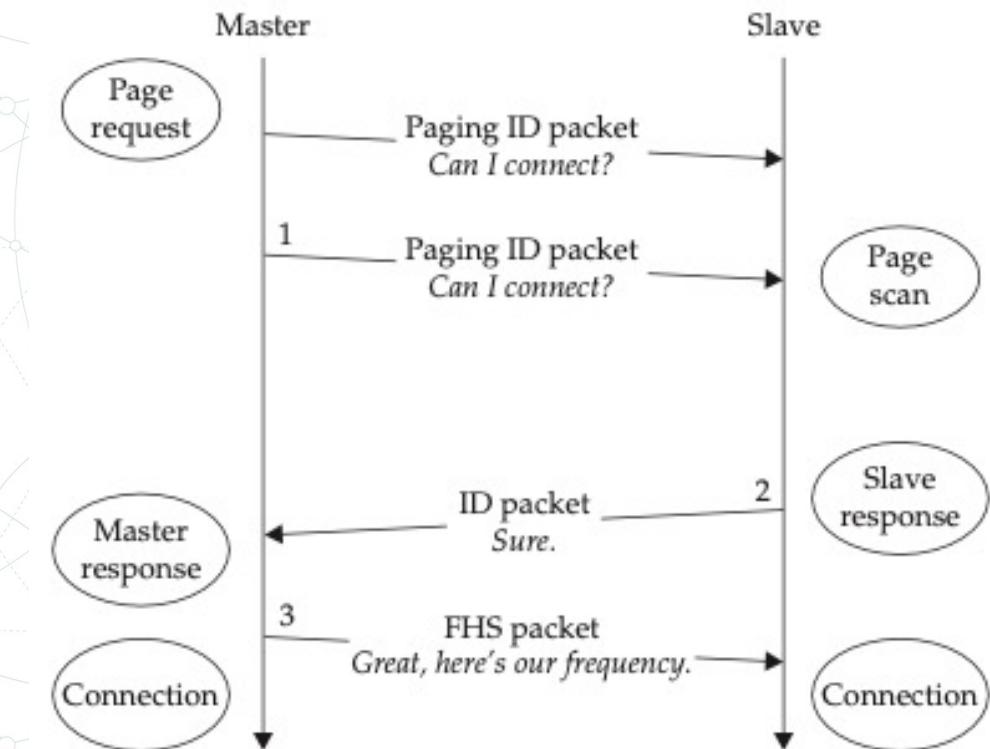
Device 3

| | | | | | | | | | | | |
|---|---|---|----|---|---|---|---|----|---|---|---|
| 6 | 4 | 5 | 10 | 1 | 2 | 6 | 3 | 11 | 8 | 9 | 7 |
|---|---|---|----|---|---|---|---|----|---|---|---|

Device 3 is not part of the piconet; it is unaware of the channel-hopping sequence in use by the other devices.

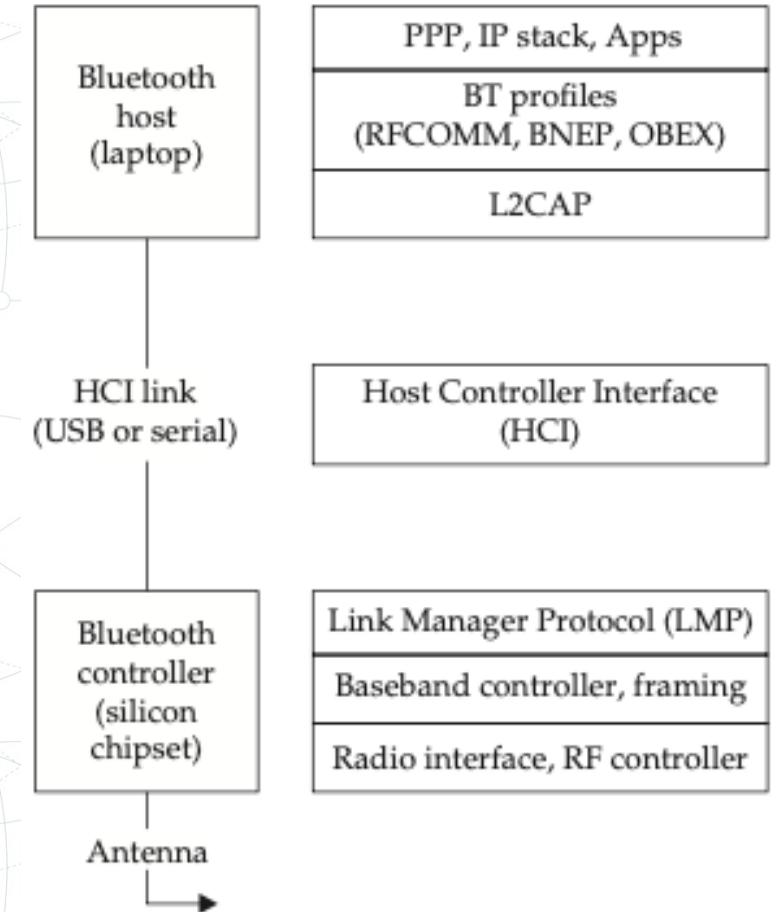
Recall: Bluetooth: Connection Establishment

- The diagram covers this in some detail
 - The most important thing to remember about “paging” or connection establishment is that in order to establish a connection you must know the target’s BD_ADDR, and that device must be interested in accepting connections



Recall: Bluetooth Protocol Overview

- The organization of layers in the Bluetooth stack and where each layer is typically implemented:
 - The controller is responsible for frequency hopping, baseband encapsulation, and returning the appropriate results back to the host
 - The host is responsible for higher-layer protocols
 - The Host Controller Interface (HCI) link is used as the interface between the Bluetooth host (your laptop) and the Bluetooth controller (the chipset in your Bluetooth dongle)

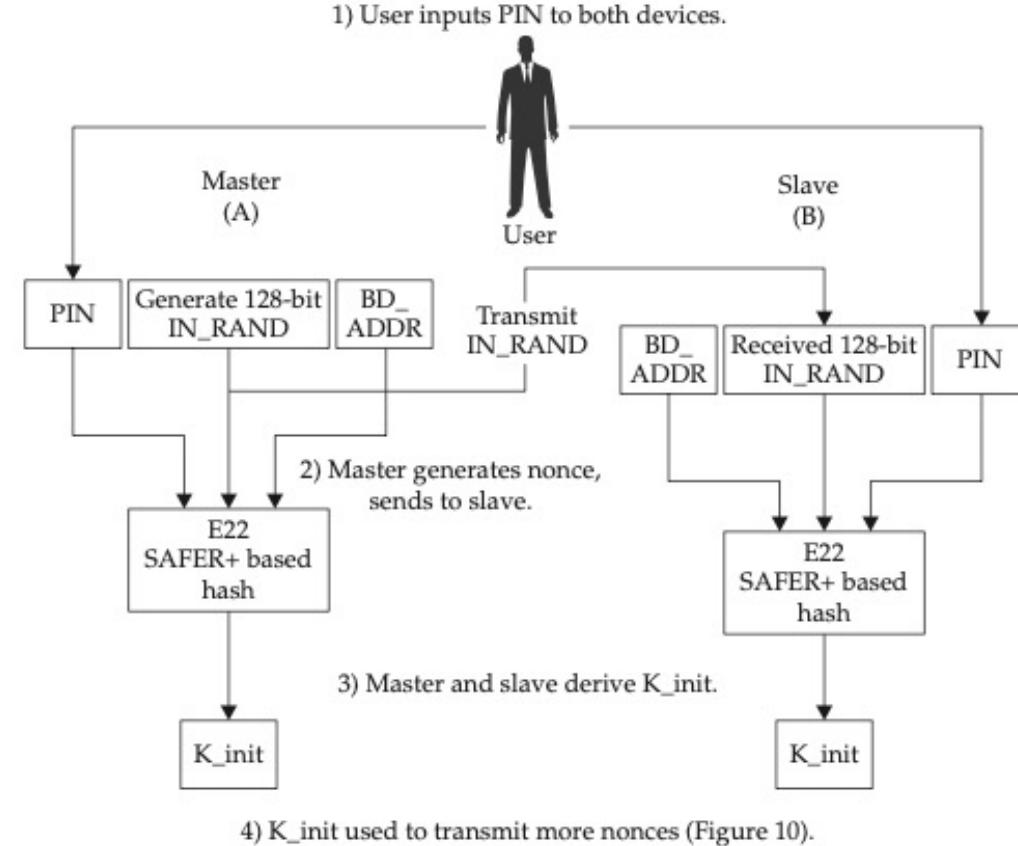


Bluetooth Encryption and Authentication

- Are built into the Bluetooth standard
 - Both are largely handled on the controller chip itself, not directly accessible to the Host Controller Interface layer
- Bluetooth devices can authenticate in two different ways:
 - Traditional pairing
 - Secure Simple Pairing (SSP)
 - SSP was added in version 2.1 of Bluetooth

Traditional Pairing Process: Link Key Creation

- Was superseded by the release of Bluetooth 2.1 by Secure Simple Pairing
 - The tradition process is still used in many Bluetooth devices
- When two devices connect for the first time, a link key is derived from a BD_ADDR, a PIN code, and a random number
 - Once both sides in the exchange have created K_init, they use it to generate a stronger link key

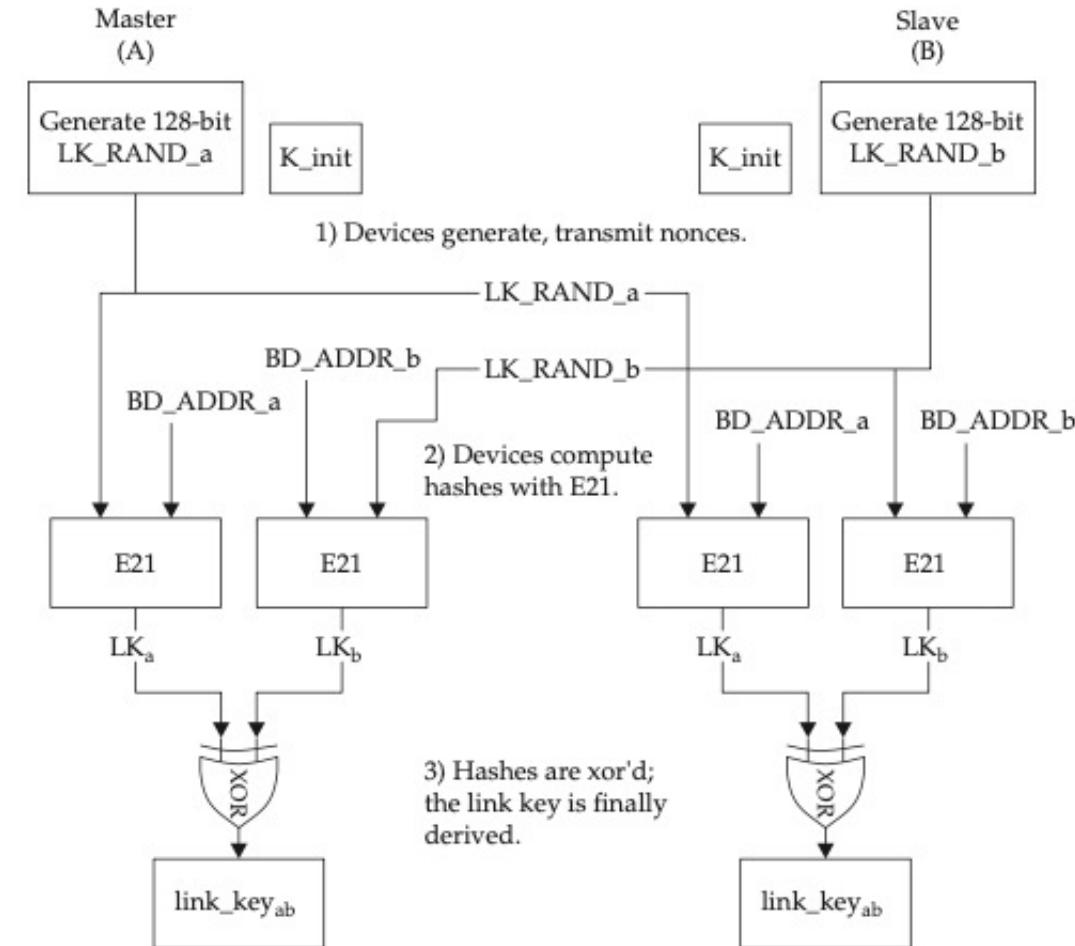


Traditional Pairing Process: PIN

- Some devices (like wireless earphones) the PIN is fixed and cannot be changed
 - In such cases, the fixed PIN is entered into the peer device
- If two devices have a fixed PIN, they cannot be paired, and therefore cannot communicate

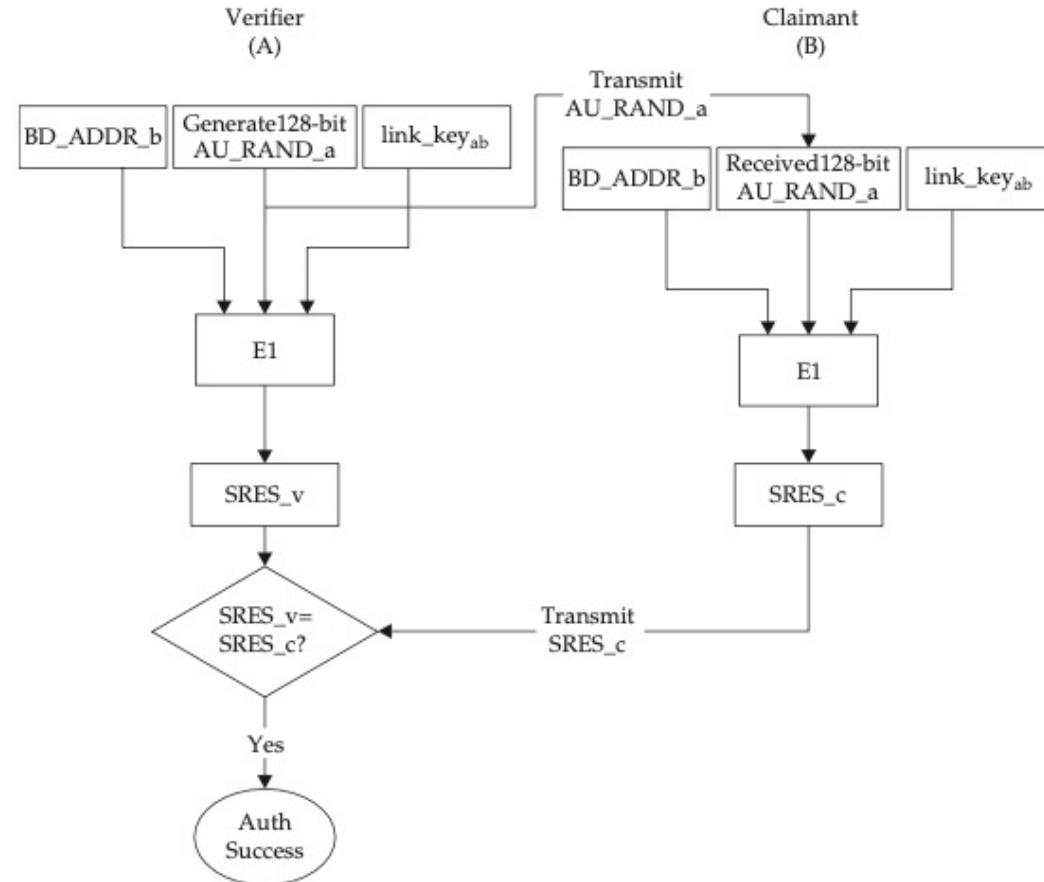
Traditional Pairing Process: Derivation of the Link Key

- Both devices have generated a 128-bit link key
- This link key will be stored alongside its peer's BD_ADDR, either on disk or in nonvolatile storage
- When the two devices want to communicate, they will go through a handshaking process
- The link-key generation will only happen each time the devices are paired (typically once)
 - For later authentication purposes, possession of this link key is used, not the PIN



Proving Link-Key Possession

- Bluetooth devices utilize possession of the link key to verify they are communicating with a device they have previously paired with
- Proving possession of this link key authenticates the peer
- The same exchange takes place if the link key was created using SSP or the traditional pairing technique



link_key_{ab} was derived during the pairing process.
This portion of the protocol is very similar to the four-way handshake used by WPA; instead of verifying possession of the PMK, we are verifying possession of the link_key.

Proving Link-Key Possession

- In this exchange, there are two entities, the Verifier and the Claimant
- The Verifier verifies that the Claimant has possession of the previously negotiated link key
 - The Verifier does this by transmitting a challenge in the clear (AU_RAND_a)
- Upon receiving the challenge, the Claimant computes the keyed hash of the challenge using an algorithm called E1 and the link key
 - The Claimant transmits the results (called the Signed Response, or SRES_c) back to the Verifier
 - Meanwhile, the Verifier computes the same hash
- If the Claimant's SRES_c matches the Verifier's computed hash, the Claimant has proven possession of the link key

Proving Link-Key Possession

- If mutual authentication is desired, the two devices reverse roles and repeat the process
- Once the devices have authenticated each other by verifying possession of the link key, they will likely proceed to derive an encryption key
 - The encryption key is derived from a hashing function, which will take the link key as input

Traditional Pairing Process: Security

- If an attacker observes the traditional link-key generation step, as well as the link-key authentication step, all he needs to do to derive the link key is to brute-force the PIN until he generates the observed value of SRES
- Once he gets the SRES to match, he possesses both the PIN and the link key

Secure Simple Pairing (SSP)

- The problem with the traditional pairing scheme is that a passive attacker who observes the pairing exchange can typically brute-force the PIN in seconds
- SSP attempts to prevent a passive observer from retrieving the link key
 - SSP accomplishes this by using public key crypto, specifically Elliptic Curve Diffie-Hellman
- A Diffie-Hellman key exchange allows two peers to exchange public keys and then derive a shared secret that an observer will not be able to reproduce
- The resulting secret key is called the *DHKey*
 - Ultimately, the link key will be derived from the *DHKey*

Secure Simple Pairing

- By using a Diffie-Hellman key exchange, a strong pool of entropy is used for deriving the link key
 - Solving the biggest problem with the standard pairing derivation, where the sole source of entropy was a PIN only a few digits in length
- SSP adds another layer of authentication to the pairing process
 - SSP depends on the devices both having some form of input and output you can use to communicate with
 - These IO capabilities are explicitly negotiated during the SSP exchange

Secure Simple Pairing: Overview

- Capabilities Exchange
- Key Exchange
- Authentication
- Link-key creation

1) Capabilities exchange

Capabilities exchange:
A and B exchange IO capabilities
(this will determine the auth technique used)

2) Key exchange

Public key exchange:
A transmits PK_A
B transmits PK_B

DHKey derived by both parties
(passive observers cannot compute this)

3) Authentication

Just Works, Numerical Comparison
Out of Band, or Passphrase authentication

Devices exchange confirmation values,
compare results

4) Link-key creation

Link-key derived
 $LK = F2(DHKey, Nmaster, Nslave, "btlk", BD_ADDR_m, BD_ADDR_s)$

Secure Simple Pairing: Capabilities Exchange

- During this exchange, the link managers on both devices trade information on whether they have the capability to input or display information
- This exchange is important because the Just Works authentication method, which is used when there is no better option, doesn't provide protection against active MITM attacks

Secure Simple Pairing: Capabilities Exchange

- Tables summarizing the capability information exchanged by devices:

| Capability | Description |
|-------------------|---|
| No input | Device cannot indicate yes or no, lacking any input capability. |
| Yes/No | Device has a button that the user can activate to indicate yes or no. |
| Keyboard | Device can input values 0–9, as well as indicate yes or no. |
| Capability | Description |
| No output | Device cannot display a 6-digit number. |
| Numeric output | Device can display a 6-digit number. |

Secure Simple Pairing: Capabilities Exchange

- Some combinations of IOCapabilities cannot provide a defense against a MITM attack
- When a link key is generated and the authentication used can protect against MITM attacks, the device key is called “authenticated”
- If the authentication scheme cannot protect against MITM attacks, the link key is said to be “unauthenticated”

Secure Simple Pairing: Capabilities Exchange

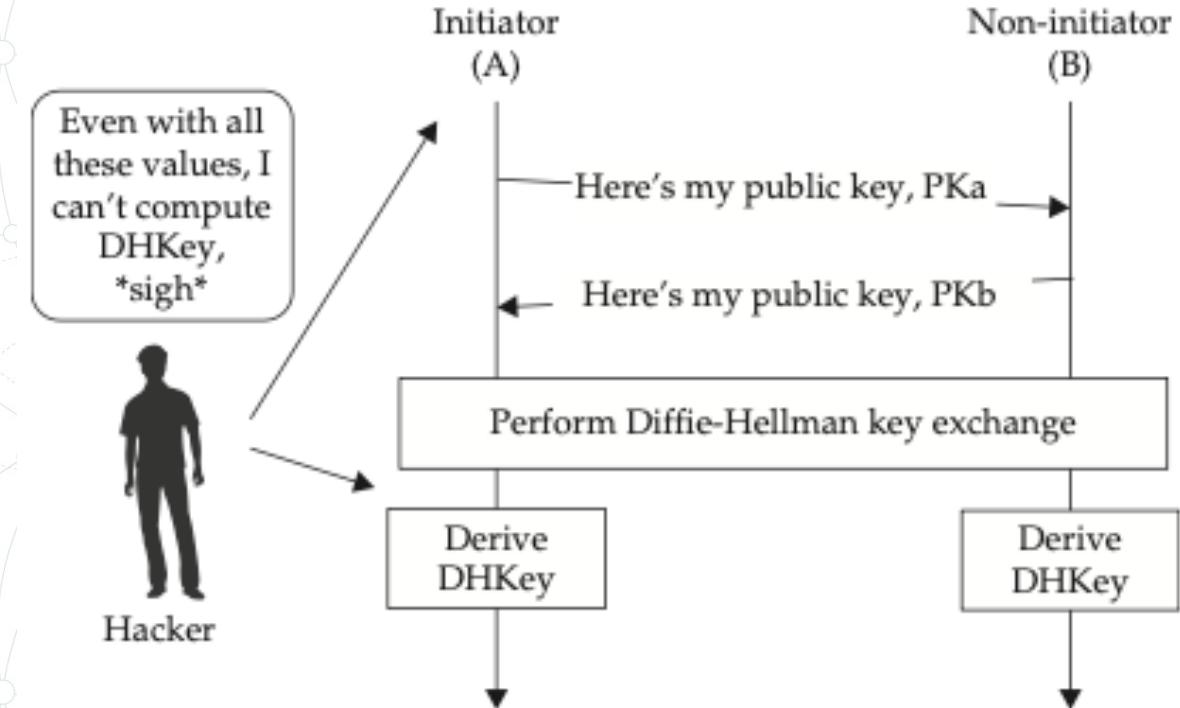
- Many devices use the weaker form of “Just Works” authentication, even when both devices support a more secure authentication scheme
 - This is because devices use the following algorithm to determine which authentication system to use
 - 1. Has any Out-Of-Band (OOB) data successfully been received from the other device? Then use the OOB authentication technique
 - 2. Do both devices support insecure (unauthenticated) link keys? Then use the Just Works (Numeric Comparison with no user confirmation)
 - 3. Does either device require an authenticated link key? Then choose the appropriate scheme

Secure Simple Pairing: Capabilities Exchange

- Notice that if both devices are configured to accept unauthenticated link keys, they will do so, even if they both have extensive IOCapabilities
- Once the IOCapabilities exchange has taken place, the devices should know what authentication technique will be used later
 - The next step is to exchange public keys and derive the DHKey

Key Exchange

- Is the easiest phase
- The devices simply transmit their public keys to each other and then perform a Diffie-Hellman key exchange operation to derive DHKey
- An attacker who captures this entire exchange will still be unable to compute DHKey, due to the cryptographic security of the Diffie-Hellman protocol



Authentication

- Once the DHKey has been derived, there are four possible authentication techniques
- In traditional pairing, you are verifying possession of a link key
 - At this current point in the SSP exchange, you haven't created the link key yet
 - Instead, you are trying to verify that the device you just performed a Diffie-Hellman key exchange with is the device you think it is
- Ultimately, verification of the link key will take place using the same algorithm used in traditional pairing

Authentication: Just Works

- This mode runs the same protocol as numeric comparison, but the user does not actually make a comparison
- This protocol is secure against passive attacks (due to the DHKey exchange), but an active MITM attack can succeed by sending both A and B its own public key and nonces at the right time

Authentication: Just Works

- Was a necessary accommodation to achieve the greatest level of compatibility between electronics design and Bluetooth security
- While other authentication mechanisms offer a greater level of security, they all require some level of Man-Machine Interface (MMI) to display content or collect responses from the end-user
 - This requirement would otherwise limit the scope of deployment options for Bluetooth developers, requiring an authentication mechanism that does not require any input from the user

Authentication: Just Works

- Just Works mechanism is also the easiest to use, leading developers and end-users to choose this authentication mechanism over stronger protocols
- Although the Just Works method protects against the passive eavesdropping attacks that plague legacy PIN authentication, it does not protect against active attacks
- Devices that accept Just Works authentication are incapable of authenticating the identity of the remote entity or defeating MITM attacks, leaving the link-layer exposed to a variety of attacks

Authentication: Just Works

- For example, consider a case where a Bluetooth USB HID interface is used on a PC intended for use with a Bluetooth keyboard
- If the PC is connectable, an adversary could impersonate a legitimate keyboard device and send arbitrary keystrokes to the host, perhaps downloading and running malware-ridden executables from the Internet
- In this example, the PC device is at fault for accepting the Just Works authentication technique when stronger input mechanisms are available
 - Though it isn't unreasonable to foresee a situation where Just Works is used by a device manufacturer to simplify the product setup process

Authentication: Numeric Comparison

- If it is used, if both devices have sufficient IO capabilities to display a six-digit number to the user
- When this technique is used, each device computes a hash of the exchanged public keys, as well as two more nonces
- The devices display six digits of this hash, and the end user is expected to verify that the displayed hashes match and select

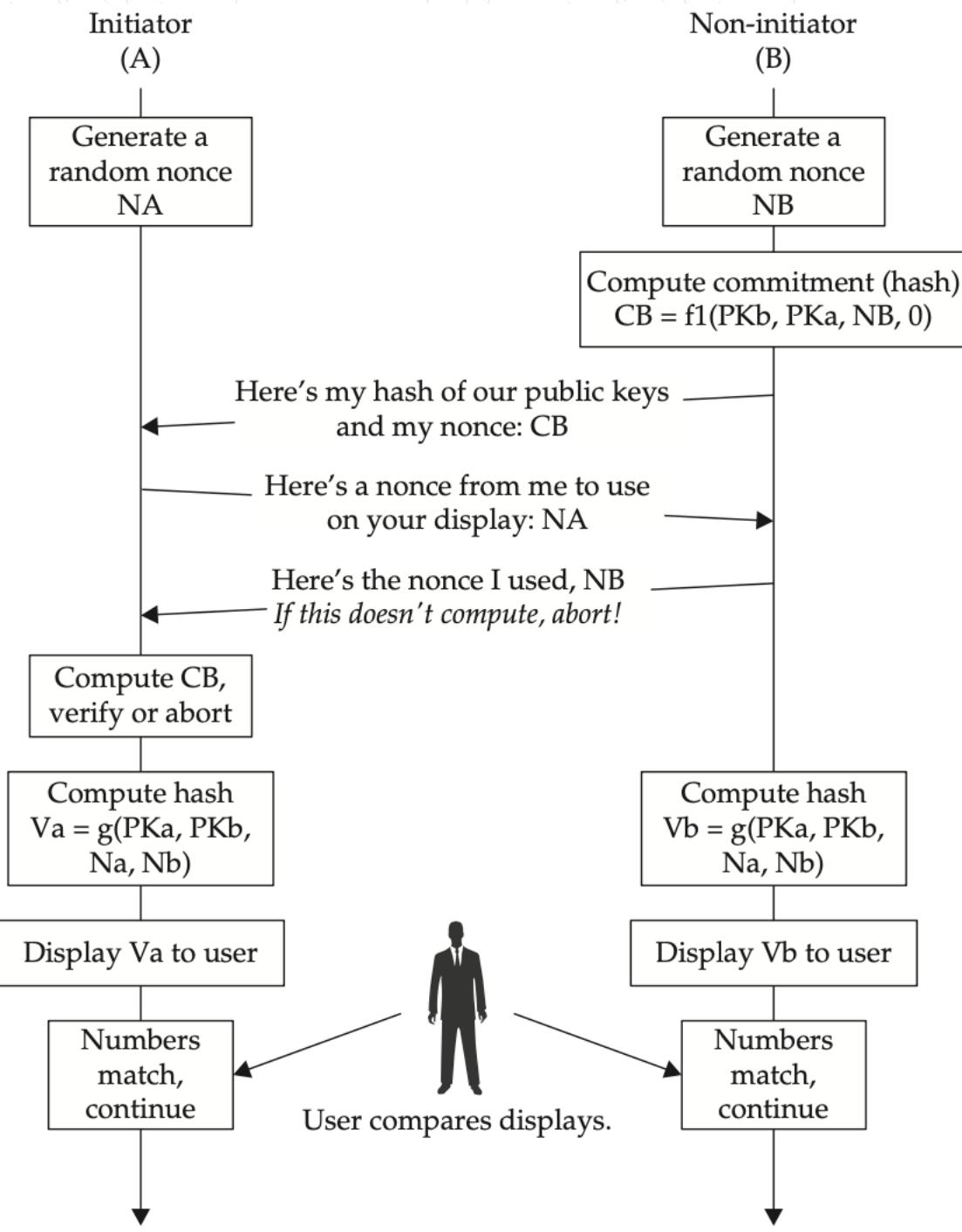
Yes or No

Authentication: Numeric Comparison

- Although this may appear to be similar to the traditional pairing process, cryptographically it is very different
- In this case, the six-digit values displayed to the user are an artifact of the pairing process
 - They are not used as input to any cryptographic functions
 - Rather he is comparing the number, which is a hash generated by both devices

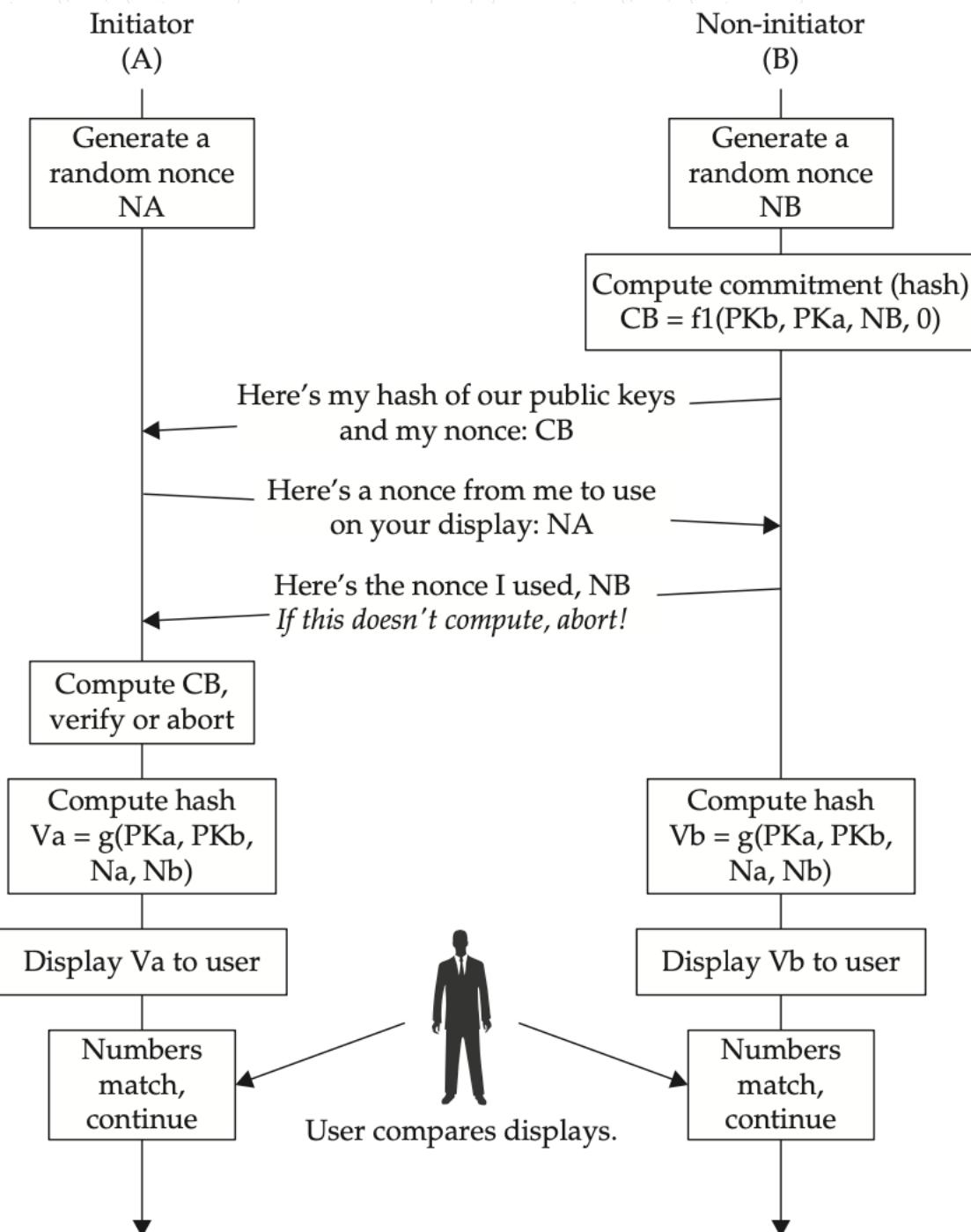
Authentication: Numeric Comparison

- Both devices generate and exchange nonces (a number used once), and then compute a hash of these nonces as well as the public keys exchanged previously
 - This hash is displayed to the user in the form of a six-digit number
 - The user is supposed to compare these numbers to verify that they match



Authentication: Numeric Comparison

- If so, the user presses the *Yes* button to indicate pairing should proceed
- If the hashes don't match, then an active attacker has tried to inject keying material or a transmission error has occurred
 - In either case, the pairing shouldn't proceed



Authentication: Out of Band

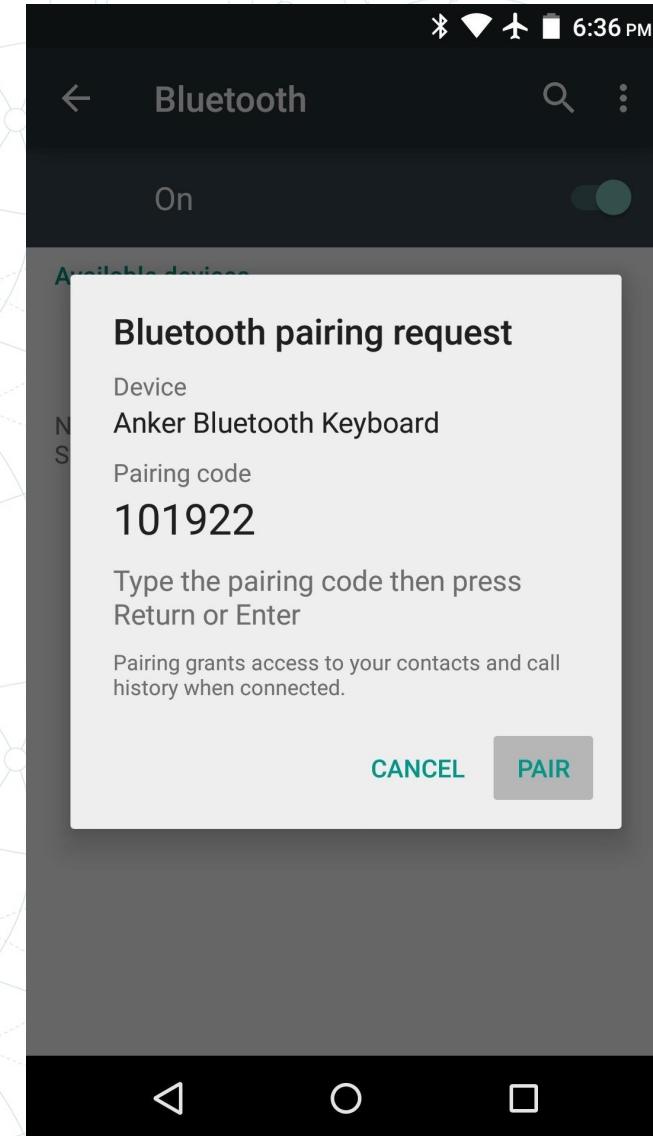
- This mode is used if the devices have some way other than Bluetooth to exchange cryptographic material
 - Near Field Communication (NFC) is described as one possibility
- If the OOB communications technique can be exploited with a MITM attack, this authentication technique will be similarly vulnerable

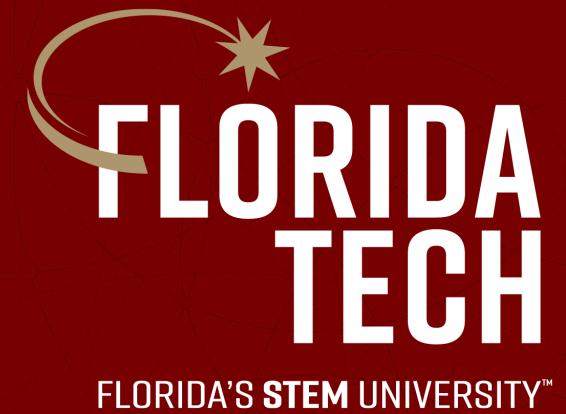
Authentication: Passkey Entry

- Passkey entry allows Bluetooth devices to authenticate each other by verifying they both possess a shared secret
- This secret can be entered into both devices or generated on one and entered into the other
- A typical use-case for this is a keyboard and computer
 - The computer generates a random PIN, and the user inputs it through the keyboard
 - When implemented poorly, this technique can be severely compromised

Authentication: Passkey Entry

- The goal of the protocol is to verify the passphrase with the other device, one bit at a time
- For every bit in the passphrase, the devices will compute a hash including the bit and transmit the hash to the other side
 - Both sides will transmit eight hashes

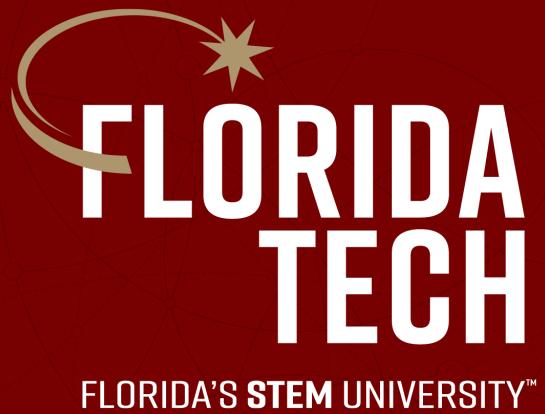




**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

11. Bluetooth Security Ct'd

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Bluetooth

Attacking Passphrase Authentication

SSP Nino Attack

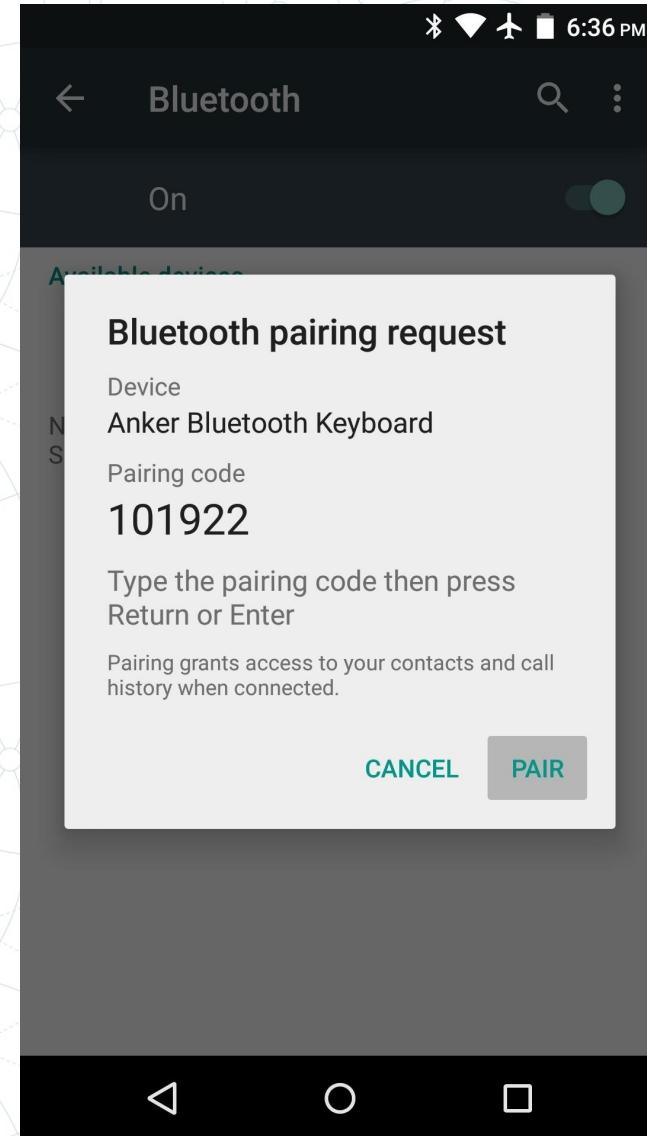
Recall: Secure Simple Pairing: Capabilities Exchange

- Tables summarizing the capability information exchanged by devices:

| Capability | Description |
|-------------------|---|
| No input | Device cannot indicate yes or no, lacking any input capability. |
| Yes/No | Device has a button that the user can activate to indicate yes or no. |
| Keyboard | Device can input values 0–9, as well as indicate yes or no. |
| Capability | Description |
| No output | Device cannot display a 6-digit number. |
| Numeric output | Device can display a 6-digit number. |

Recall: Authentication: Passkey Entry

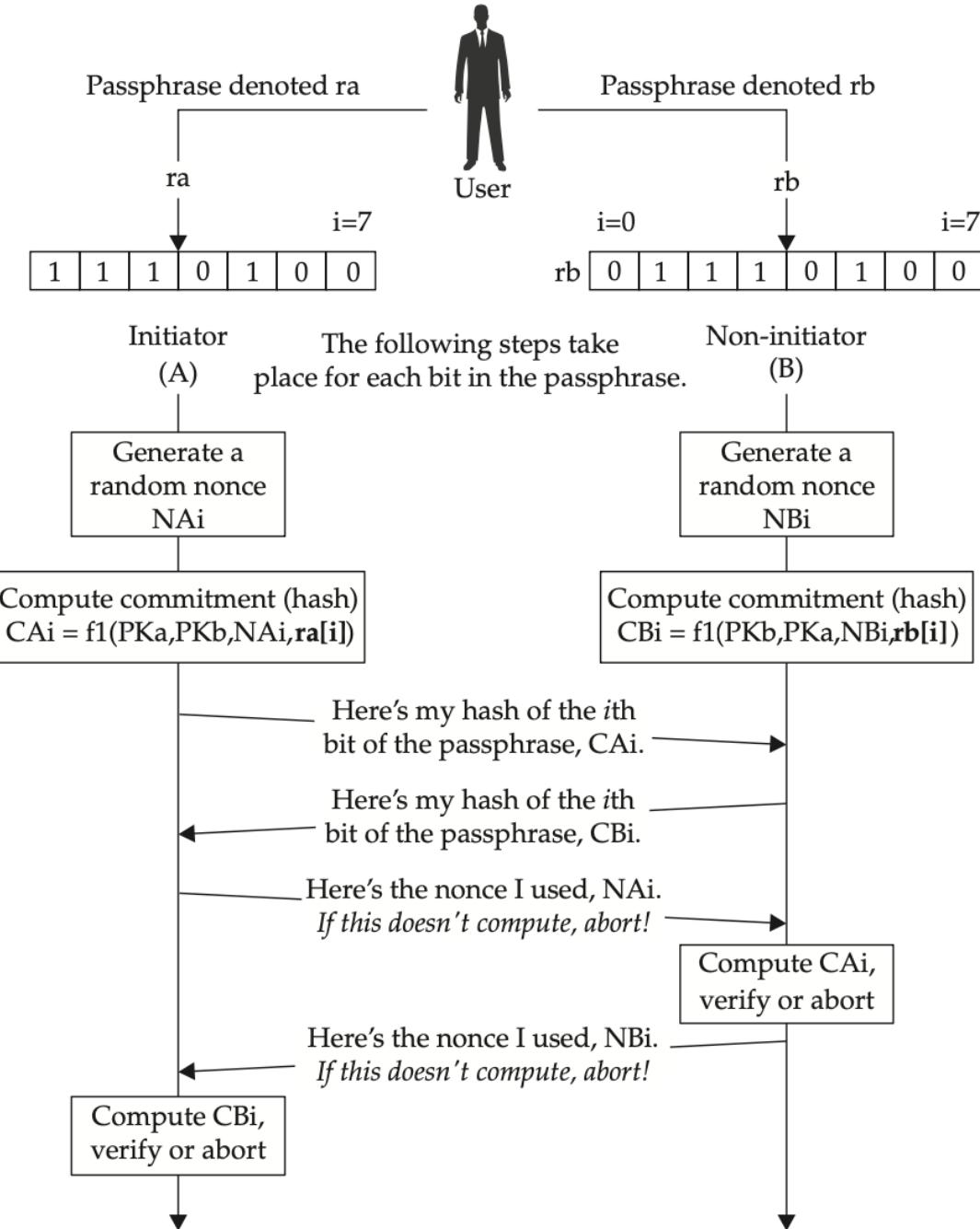
- The goal of the protocol is to verify the passphrase with the other device, one bit at a time
- For every bit in the passphrase, the devices will compute a hash including the bit and transmit the hash to the other side
 - Both sides will transmit eight hashes



Authentication: Passkey Entry

- The Bluetooth specification calls this protocol a “gradual release” procedure of the passphrase, and it is rationalized in the following manner
 - Device A reveals $ra[i]$ before device B does
 - B, however, has already transmitted his commitment hash, before A reveals the bit
 - This means that B cannot change his choice at this point (in other words, he is committed)

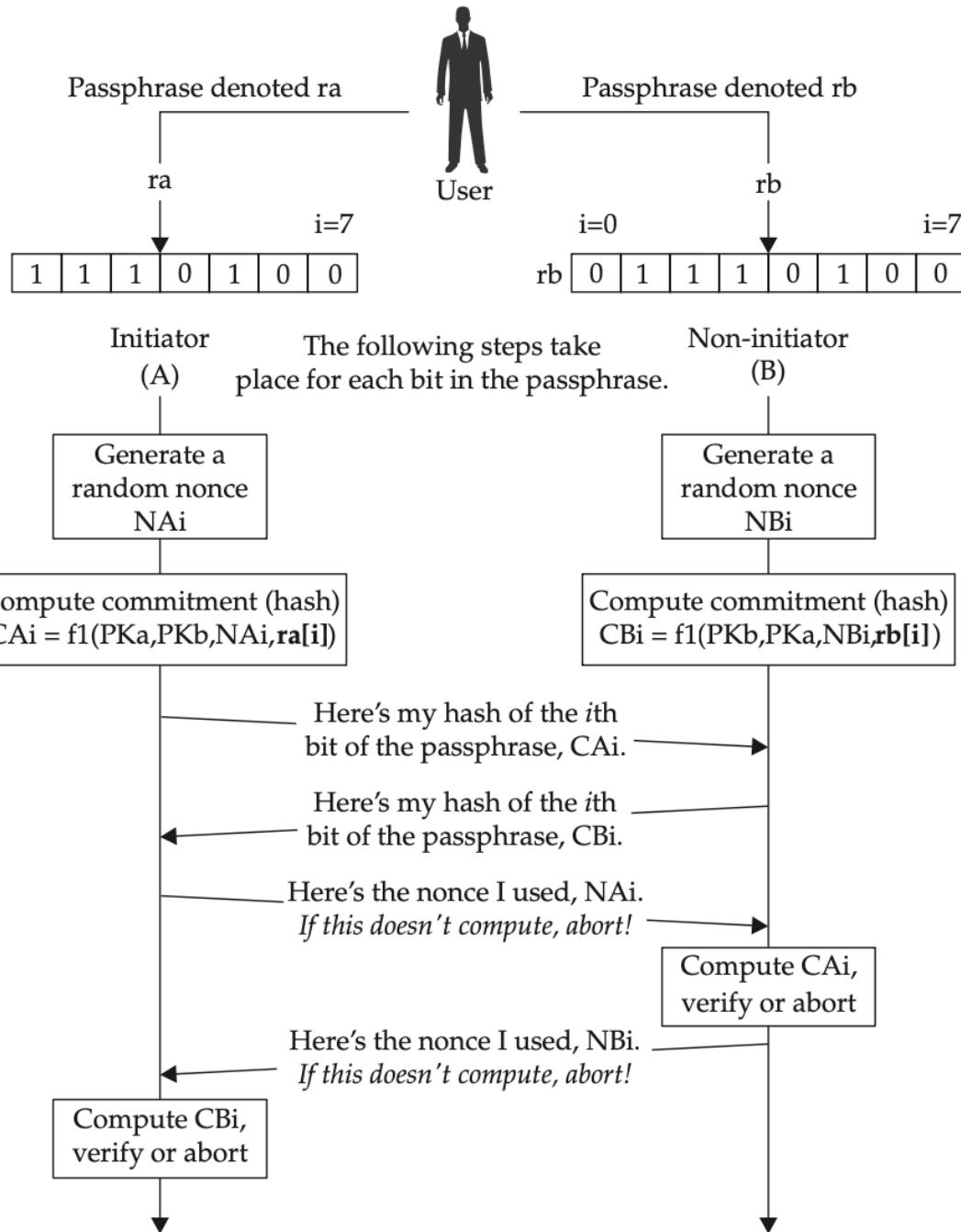
1) User inputs passphrase to both devices.



Authentication: Passkey Entry

- This process prevents a MITM attack because the commitment hash also has the public keys used in the DHKey exchange as input
 - A potential MITM attacker will only be able to guess with 50 percent accuracy
 - The odds that an attacker will successfully guess a randomly generated passphrase in such a matter are $1/2^i$, where i is the length of the passphrase in bits

1) User inputs passphrase to both devices.



Passively Attacking Passphrase Authentication

- An attacker who can observe the entire exchange can trivially recover the passphrase
- For every bit in the passphrase, the attacker will know the following values:
 - PK_a, PK_b : Public keys observed during the initial public key exchange
 - CA_i, Cb_i : The commitment hashes for the ith bit
 - NA_i, Nb_i : The nonces used as input to the above commitment hashes

Passively Attacking Passphrase Authentication

- Given these values, an attacker has what he needs to compute the i th bit himself
 - Remember, there are only two possible values for $ra[i]$: 1 or 0
 - The attacker simply has to compute $f1(PKa, PKb, Nai, 0)$
 - If the result equals CAi , then $ra[i] = 0$
 - If that doesn't pan out, the attacker can compute $f1(PKa, PKb, Nai, 1)$, which will reveal $ra[i]$ is, in fact, 1
 - The gist of this is that an attacker can retrieve the passphrase in its entirety, and all he has to do is observe the pairing and compute a few SHA256 hashes

Passively Attacking Passphrase Authentication

- In this regard, SSP is actually worse than the PIN-based scheme used in traditional pairing
 - Under the old system, if the user inputs a 32-bit pin, an attacker will need to compute 2^{32} hashes (worst case) before finding it
 - In the current system, the attacker will need to compute at worst 32 hashes
- Yet, an attacker who learns the passphrase still does not know the link key, because it will be derived from the *DHKey*, which an attacker cannot derive
 - Once the attacker has recovered the passphrase, he can try to convince the devices to pair with him

Actively Attacking Passphrase-Protected Devices

- Another attack can be levied against a passphrase-protected device
 - Assume that a device has a 32-bit passphrase (for simplicity)
 - Also assume that this device can be placed in pairing mode repeatedly
 - Finally, assume that you would like access to this device, and you don't have the link key or passphrase
- All you need to do is randomly choose a bit for $ra[i]$, starting with $ra[0]$ and working your way up
 - If the device continues, then you chose correctly
 - If the device aborts, you know that you chose incorrectly and, therefore, will choose correctly the next time

Actively Attacking Passphrase-Protected Devices

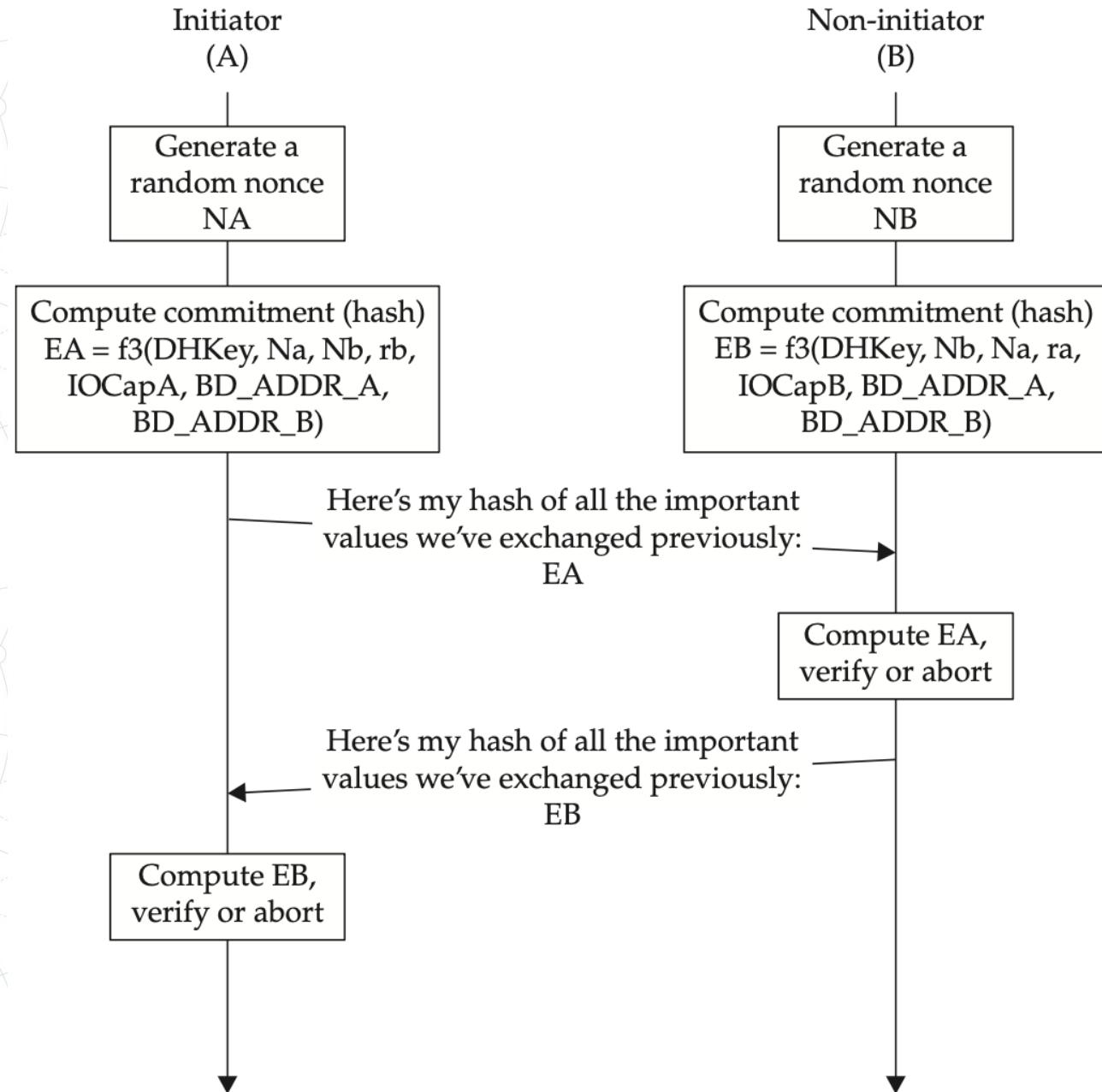
- Assuming the device has a 32-bit passphrase, you will be able to guess the entire passphrase correctly after 16 pairing attempts (on average)
- The problem is that the protocol reveals a bit of the passphrase to the attacker at every step, regardless of her choice
- The specification says that exponential back-off times should be used to slow down attacks such as this
 - But when such a small number of attempts are needed, it seems unlikely to help

Countermeasure

- The moral of this story is that devices using passphrase authentication should never be configured to reuse a static passphrase
- If the passphrase is randomly generated on every attempt, you will not be able to carry the partial information across attempts

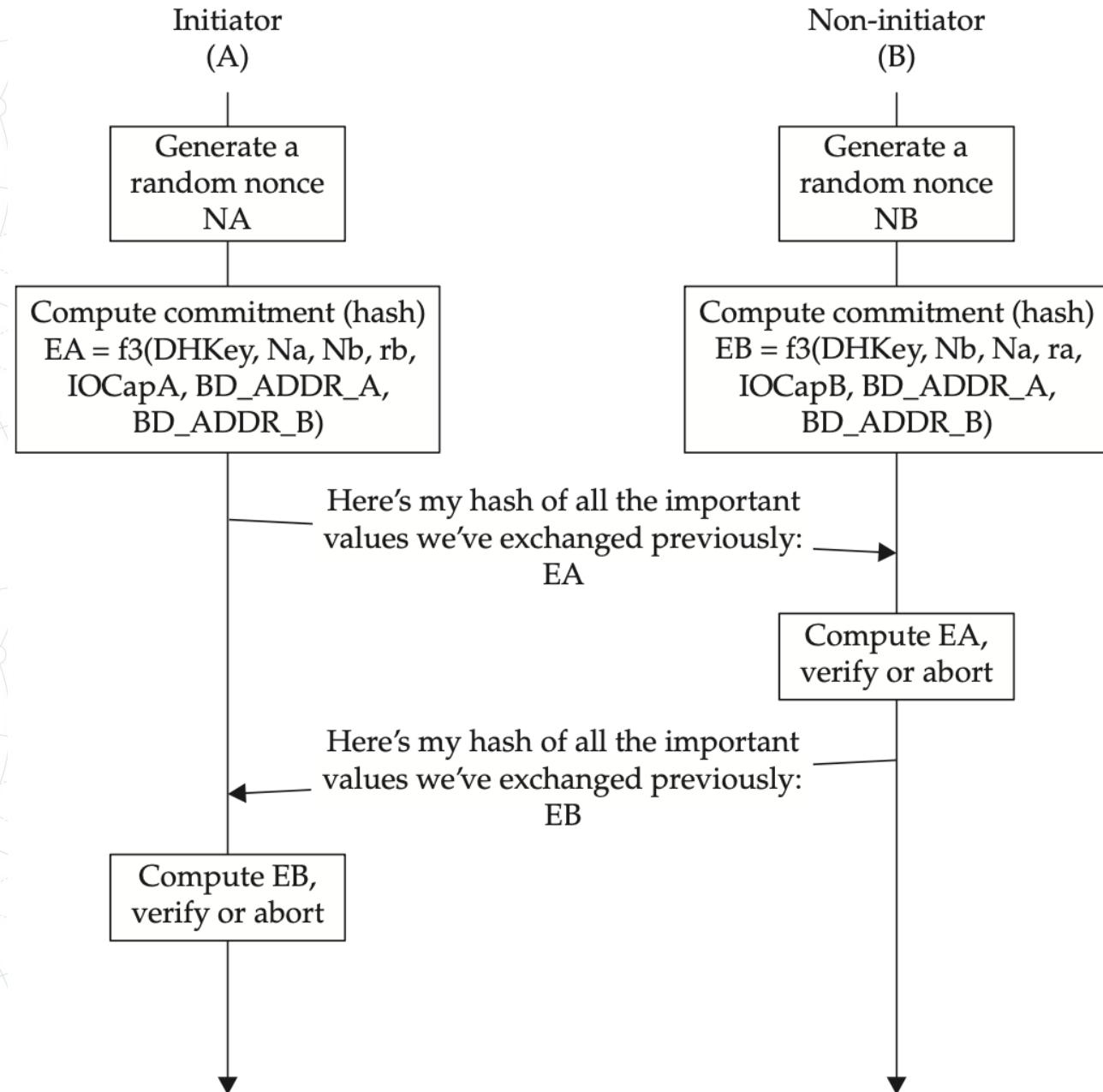
Authentication Phase 2

- Regardless of the authentication technique used, once authentication has completed, the following exchange takes place:
 - Device A computes a hash over all of the important values used previously (the DHKey, IOCapabilities, BD_ADDRs, etc.) and transmits the hash to device B



Authentication Phase 2

- This is device B's last chance to verify that they have agreed on everything so far
- Device B should verify this hash, and if it checks out, send a similar hash of his own
- At this point, the authentication phase is complete



Link-Key Derivation

- Once the authentication phase is complete, the devices are convinced that the DHKey was negotiated with the desired party
 - Now it's time to use it for creating the link key
- Link-key derivation is simple at this point because all of the authentication is out of the way
- The link key is derived from the DHKey using the following hash:
 - $LinkKeyAB = f2(DHKey, N_{master}, N_{slave}, "btlk", BDADDR_{master}, BD_ADDR_{slave})$

Link-Key Derivation

- Once the devices compute the link key, they will store it along with the peer's BD ADDR
- Then they can use it for future authentication sessions, without having to re-create it
- At this point, SSP becomes identical to traditional pairing
- Encryption keys will be derived using the same hashing values

Link-Key Derivation

- The same encryption system is used to protect the link key
- Note that an attacker who observes any of these exchanges will not be able to compute the link key because it was derived from a Diffie-Hellman key exchange

SSP Summary

- Despite attacks on passphrase authentication just outlined, on average Bluetooth security is enhanced via the use of SSP
- Unless there is a serious cryptographic breakthrough, a passive attacker should not be able to recover the link key since this would require a passive attack against the Diffie-Hellman key exchange
- The best passive attack known to date involves the misuse of the passphrase authentication scheme with a static key
 - An attacker who observes this can trivially compute the passphrase used for the pairing session

SSP Summary

- This is a serious flaw, because the attacker can potentially pair with the compromised devices herself, but she still cannot impersonate one device to the other, or decrypt intercepted traffic
- In order for an attacker to recover the link key between two devices, she must actively attack them during the pairing process
- A successful attack will result in the compromised devices pairing with the attacker, instead of themselves
 - The attacker can then read / write data to both devices

SSP Niñó (NoInputNoOutput) Attack

- When two devices pair, they negotiate the IOCapability information to identify a suitable authentication mechanism that is supported by both devices
 - This exchange happens before the link key is derived and, as such, is not a protected exchange
- This attack leverages this SSP IOCapabilities exchange deficiency to manipulate one or more devices, forcing the victim device to “dumb-down” its selected authentication mechanism to the Just Works technique
 - Once one or more devices are forced to this weaker authentication method, the attacker can eavesdrop on any data sent between devices

SSP Niñó Attack

- First, the attacker must force two devices to re-pair
 - One option is to launch a denial-of-service (DoS) attack against the Bluetooth devices in the area, designing a transmitter that hops along with the piconet master and jamming on each channel during the frequency-hopping exchange
 - A second option is to leverage a wide-band jammer that can jam all
- 79 Bluetooth channels simultaneously, ceasing all Bluetooth communication within range of the attacker
 - Once the end-user becomes frustrated with the lack of communication between two Bluetooth devices, the attacker would hope that the user attributes the failure to the devices themselves and attempts to repair the devices to resolve the issue, at which time the attacker would stop the DoS attack

SSP Niñó Attack

- Immediately before the devices re-pair, the attacker would impersonate the BD_ADDR and friendly name of both devices and implement a MITM attack, brokering the authentication exchange between the victim devices
- Instead of allowing the legitimate advertised IOCapabilities to pass between devices, however, the attacker would indicate to the responder that the initiator only supports the NoInputNoOutput capability, and vice versa, effectively dumbing-down the connection exchange and leaving the Just Works authentication method as the only plausible method

SSP Niñó Attack

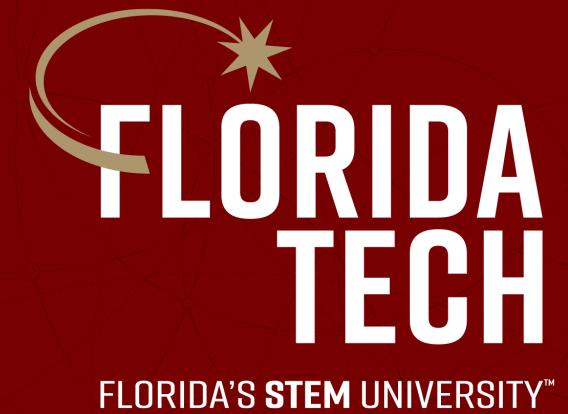
- Because the Just Works method does not provide protection against MITM attacks, the attacker can complete the authentication process with each device to conclude the pairing exchange and finish the connection establishment
- With knowledge of the link key derived by both devices, the attacker can now eavesdrop on the traffic between the devices, optionally manipulating the content as desired

Nino: Countermeasure

- The Niño attack leverages a design concession in the SSP specification to accommodate devices with no man-machine interface
- Due to the lack of cryptographic integrity protecting the IOCapabilities exchange, an attacker could leverage this weakness to manipulate one or more victim devices into creating a connection with the attacker
- There are no available exploit tools to implement the Niño attack
 - With integrated support in the BlueZ stack for SSP, however, it would be possible to impersonate both devices in anticipation of a pairing exchange and force the use of the Just Works authentication mechanism
 - Additional research and experimentation is needed to evaluate the practicality of this attack against real-world implementations

Nino: Countermeasure

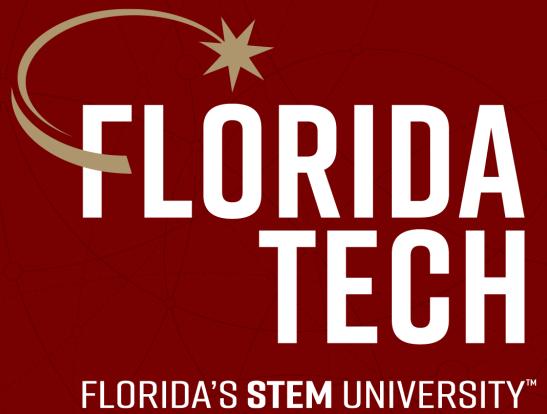
- One significant requirement for the Niño attack is to force a Bluetooth user to delete the prior pairing information and re-pair devices, or to actively catch a user pairing two devices for the first time
- You can leverage this attack requirement as a countermeasure, by not pairing Bluetooth devices in any location that is susceptible to traffic sniffing attacks
 - If one or more of your Bluetooth devices prompts you to spuriously repair, disable Bluetooth on both devices temporarily until you can return to a safe location and re-pair



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

12. Midterm Review

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

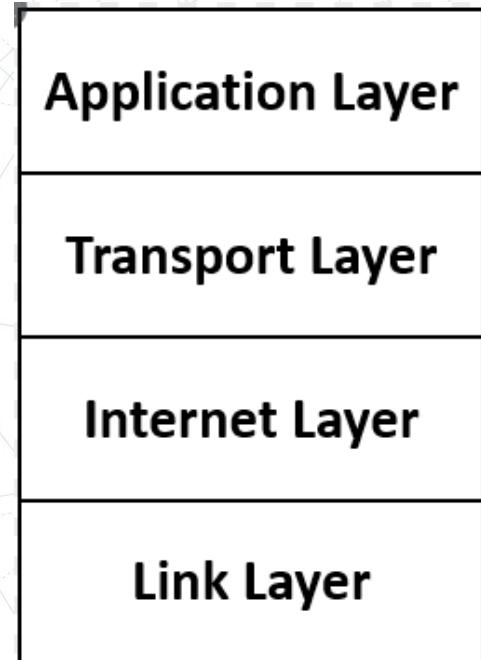
L1: Outline

Background: Computer Networks

Background: Cybersecurity Basics

Network Protocols

- **Application Layers:** End-user applications
- **Transport Layer:** Data transfer from end to end
- **Internet (Network) Layer:** Routing of data from source to destination
- **Link (Physical) Layer:** Physical media carrying the data



Discussion

- Which attack type(s) do you think the most used against Wifi/Mobile?
 - DDoS
 - Phishing
 - MiTM
 - Malware

L2/3: Outline

WiFi

WiFi Security

WEP

WPA/WPA2/WPA3

802.11 Packet Types

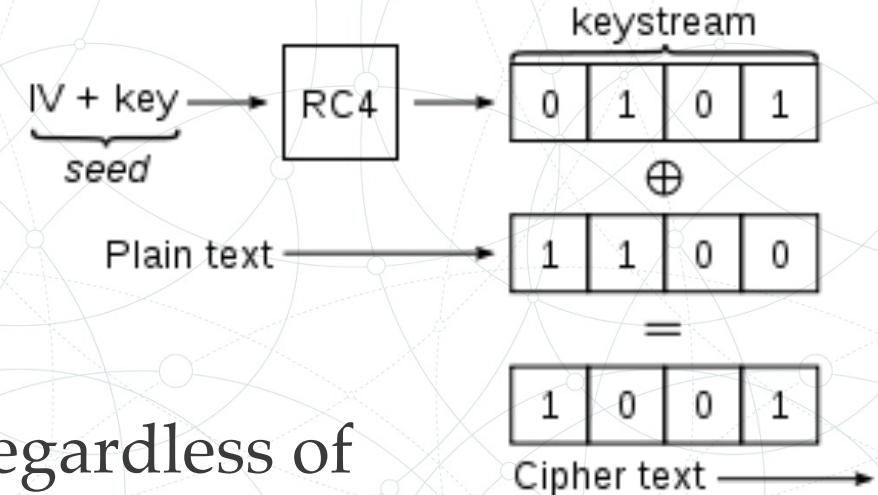
- Data packets are used for carrying high-level data (e.g., IP packets)
- Management packets to control the network
 - Attackers are interested in these mostly
 - Beacon and De-authentication are examples of management
- Control packets are used to mediating access to shared medium
 - Examples: RTS (Request to Send) and CTS (Clear to Send)

WEP: Encryption

- WEP initially used a 64-bit key with the RC4 stream encryption algorithm to encrypt data transmitted wirelessly
- Later versions of the protocol added support for 128-bit keys and 256-bit keys for improved security
- WEP uses a 24-bit initialization vector, which resulted in effective key lengths of 40, 104 and 232 bits

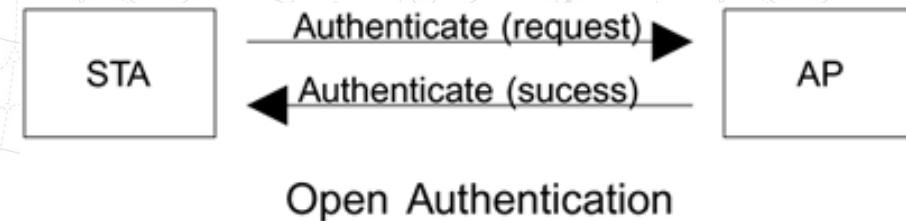
WEP: Encryption

- This is a static key, which means all traffic, regardless of device, is encrypted using a single key
- A WEP key allows computers on a network to exchange encoded messages while hiding the messages' contents from intruders
- This key is what is used to connect to a wireless-security-enabled network



WEP: Authentication: Shared Key

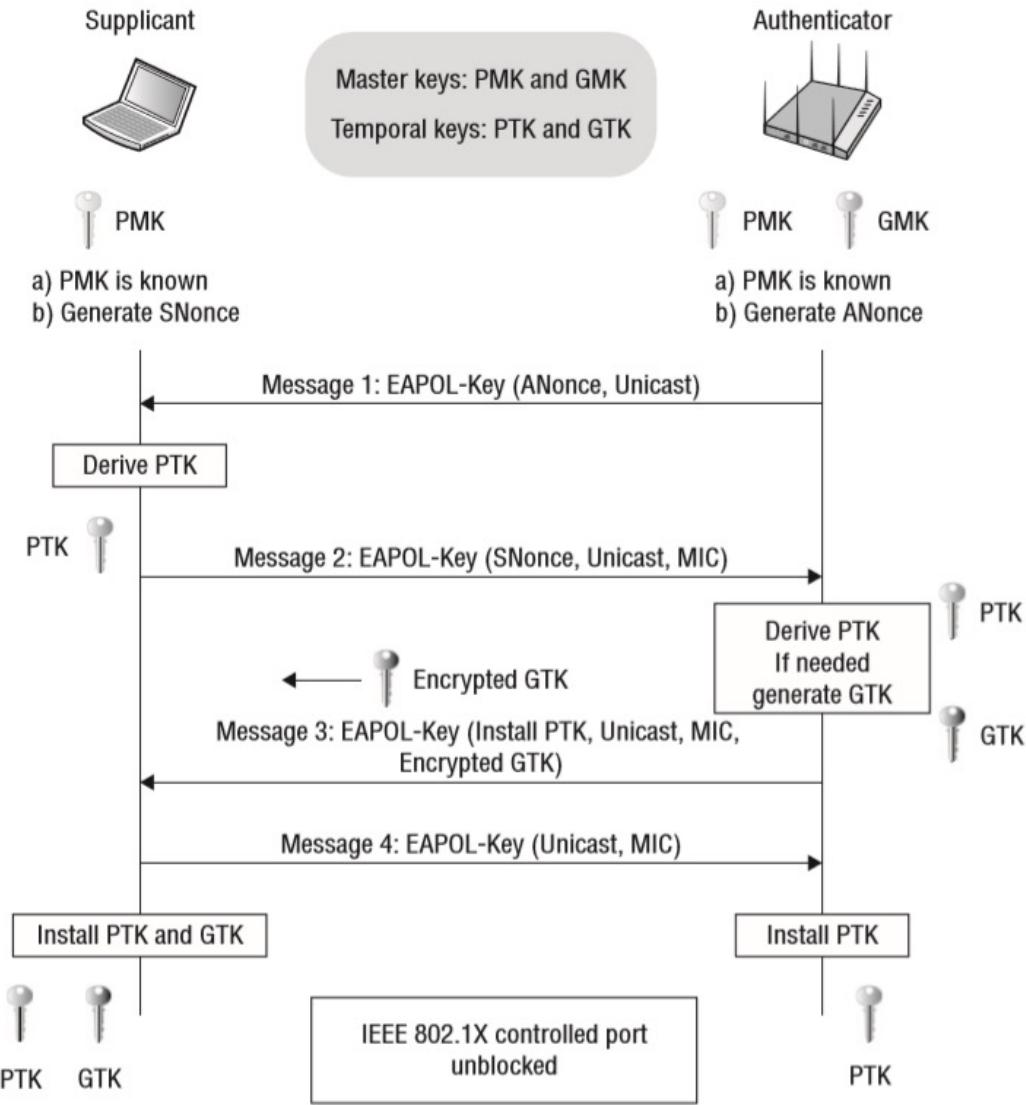
- A four-step challenge-response handshake:
 - The client sends an authentication request to the AP
 - The AP replies with a clear-text challenge
 - The client encrypts the challenge-text using the configured WEP key and sends it back in 'authentication response'
 - The AP decrypts the response. If this matches the challenge text, the AP sends back a positive reply



WPA: Pre-shared Key

- **Message1:** AP sends EAPOL message with Anonce (random number) to the device to generate PTK
 - Client device knows AP's MAC because its connected to it
 - It has PMK, Snonce and its own MAC address
 - Once it receives Anonce from AP, it has all the inputs to create the PTK

$$PTK = PRF (PMK + Anonce + SNonce + Mac (AA) + Mac (SA))$$



WPA2

- WPA still uses the RC4 encryption algorithm, and retained other weaknesses from WEP
- WPA2 was introduced in 2004 and was an upgraded version of WPA
- WPA2 is based on the robust security network (RSN) mechanism and operates on two modes:
 - **Personal mode or Pre-shared Key (WPA2-PSK)** – which relies on a shared passcode for access and is usually used in home environments
 - **Enterprise mode (WPA2-EAP)** – as the name suggests, this is more suited to organizational or business use

WPA2

- Both modes use the CCMP (Counter Mode Cipher Block Chaining Message Authentication Code Protocol)
- The CCMP protocol is based on the Advanced Encryption Standard (AES) algorithm, which provides message authenticity and integrity verification
- CCMP is stronger and more reliable than WPA's original TKIP, making it more difficult for attackers to spot patterns

WPA2

- However, WPA2 still has drawbacks
 - For example, it is vulnerable to key reinstallation attacks (KRACK)
 - KRACK exploits a weakness in WPA2, which allows attackers to pose as a clone network and force the victim to connect to a malicious network instead
- This enables the hacker to decrypt a small piece of data that may be aggregated to crack the encryption key
- Yet, WPA2 is still considered sufficiently secure and more secure than WEP or WPA

WPA3

- WPA3 is the third iteration of the Wi-Fi Protected Access protocol
- The Wi-Fi Alliance introduced WPA3 in 2018
- WPA3 devices became widely available in 2019 and are backwards compatible with devices that use the WPA2 protocol
- WPA3 introduced new features for both personal and enterprise use

L4: Outline

Definitions

Frequency, Amplitude, Wavelength

Modulations

Multiple Access

L5: Outline

802.11 Enumeration

Airodump-ng

- Airodump-ng is used for packet capture, capturing raw 802.11 frames
- It is particularly suitable for collecting WEP IVs (Initialization Vector) or WPA handshakes for the intent of using them with aircrack-ng
- If you have a GPS receiver connected to the computer, airodump-ng is capable of logging the coordinates of the found access points

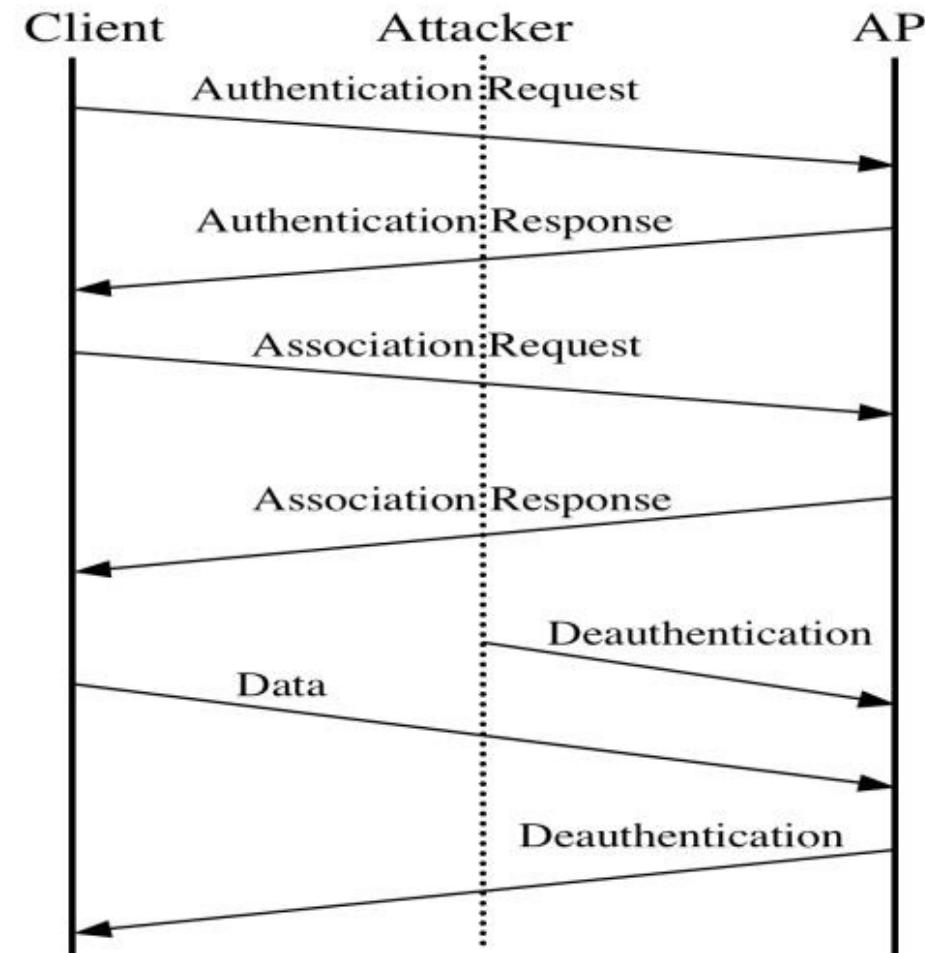
L6: Outline

Security through Obscurity

WiFi Di-association/De-authentication

De-authenticating user

- Management frames in 802.11 are not authenticated
 - Send a packet to user that looks like coming from AP
 - The user can't tell the difference
 - Wireless driver will reconnect immediately
 - Reassociation request with the SSID in it will be sent



Di-association vs. De-authentication

- In the case of a regular home router, you both authenticate and associate to the same AP
 - And if you disconnect, you both deauthenticate and disassociate to the same AP
- But in a larger network made out of multiple APs, you might disassociate from one AP and associate to a new one while staying authenticated to the same network

How can De-auth be Exploited?

- Deauthentication frames are very simple in their structure
 - You basically only need a sender or receiver MAC address
 - And you can obtain such by simply scanning for WiFi devices nearby
- Thus, it's very easy to spoof a deauth packet
 - And keep in mind that if the target receives it, it has to drop its connection

How can De-auth be Exploited?

- The target can reconnect immediately and it can do that quite fast, maybe without the user noticing that the connection was ever dropped
- But if these deauth packets are sent continuously, it results in a denial of service attack, and network access is blocked for the entirety of the attack
- Luckily this was addressed, and we now have protected management frames!
 - This feature allows packets like deauthentication frames to be safe against spoofing

Protected Management Frames

- PMF provide protection for unicast and multicast management action frames
 - Unicast management action frames are protected from both eavesdropping and forging, and multicast management action frames are protected from forging
 - PMF is required for all new certified devices
 - However, may not be implemented in all devices out there

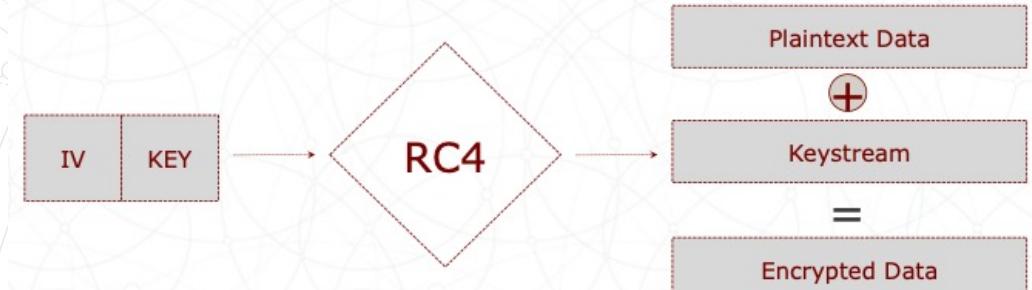
L7: Outline

WEP Security Analysis

MAC Filtering

WEP Keys

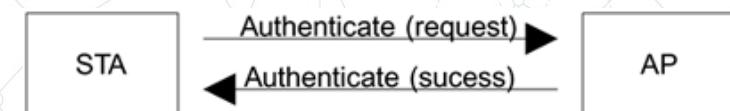
- Protects eavesdropping by preventing repetition of RC4 Key



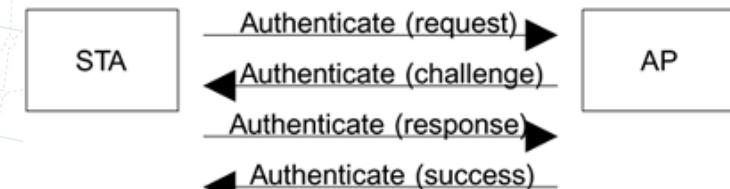
- IV is 24 bits; so total IVs = $2^{24} = 16,777,216$

- Probability of IV Repetition

after 5,000 frames = 50 %



Open Authentication



WEP Authentication

Direct Key Attacks

- The method can be tuned to attack each secret key byte in turn so eventually the entire secret key can be extracted
 - Note that increasing the key size from 40 bits to 104 bits means that it takes 2.5 times longer to extract the key (linear, not exponential)
- All the previous weaknesses of WEP pale into insignificance compared to this attack
 - Remember that extracting the keys is the ultimate goal of an attacker, and here is a method that directly extracts the keys in linear time
 - This attack blew apart the remnants of WEP security
 - Because it used a fairly mechanical approach, it was feasible to create a script tool that would do the job unattended
- Within months, some "helpful" person invested their time into generating a cracker tool

L8: Outline

WPS

WPA/WPA2 Brute Force

Reaver Brute-force Attack

- Was a radical new weapon for Wi-Fi hacking when it was presented in 2011
 - Now obsolete against most routers
- One of the first practical attacks against WPA- and WPA2-encrypted networks, it totally ignored the type of encryption a network used, exploiting poor design choices in the WPS protocol
- Reaver allowed a hacker to sit within range of a network and brute-force the WPS PIN, spilling all the credentials for the router
 - Worse, the 8-digit-long PIN could be guessed in two separate halves, allowing for the attack to take significantly shorter than working against the full length of the PIN

WPA Brute Force

- To authenticate with a WPA/WPA2-Personal AP, a station must complete a four-way handshake, securely providing the PSK (Pre-shared Key) without disclosing it in plaintext
- To perform the handshake, both the AP and the station must generate a nonce (a number used only once) to share with one another
 - The AP and the station then both feed the nonce and the pre-shared key (PSK) into a pseudo-random function which generates a pairwise transient key (PTK)
 - The created PTK is unique to both the AP and the station and is used to encrypt communications between the devices, completing the authentication

WPA Brute Force

- However, authentication to an AP is conducted through management frames; meaning, if an attacker can capture the four-way handshake, they will have access to all factors which generated the PTK
 - Isolating the wireless password as the missing variable
- To crack the password, loop this pseudo-random function (with the same nonce's) and a list of possible password as input for the pre-shared key
 - Starting with dictionary list (dictionary attack)
 - Once the transient key generated matches the one from the captured traffic, the password is correct

<https://www.wikihow.com/Hack-WPA/WPA2-Wi-Fi-with-Kali-Linux>

WPA Brute Force

- No difference between cracking WPA or WPA2 networks
 - The authentication methodology is basically the same between them
 - Thus, the techniques you use are identical
- This can be done either actively or passively
 - “Actively” means you will accelerate the process by deauthenticating an existing wireless client
 - “Passively” means you simply wait for a wireless client to authenticate to the WPA / WPA2 network

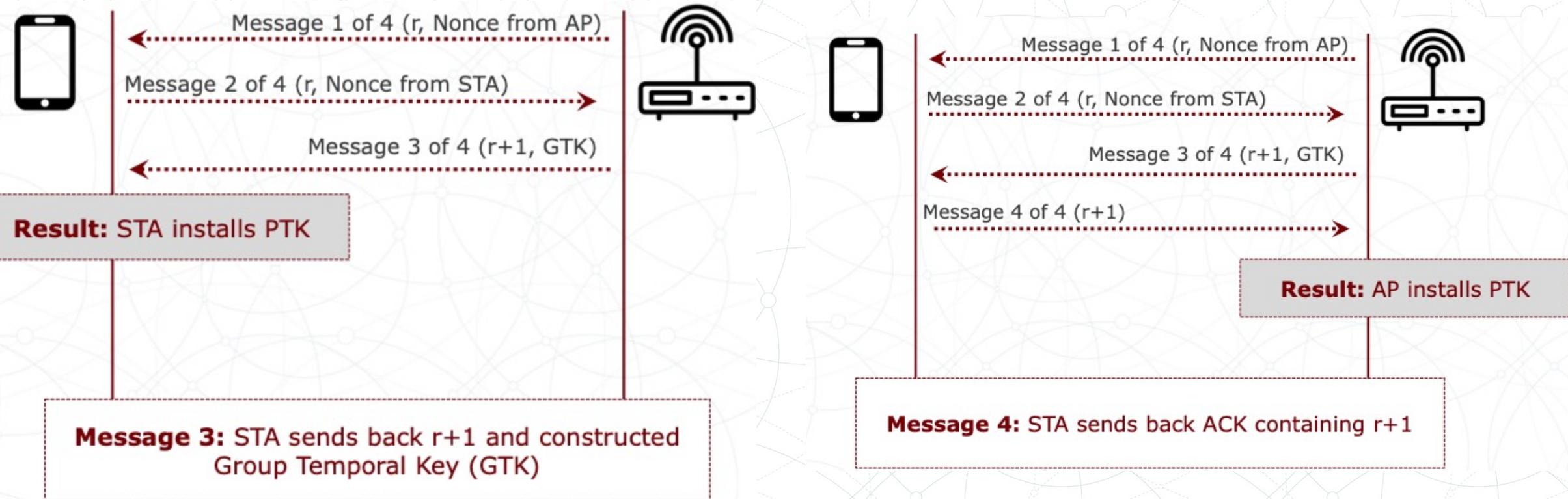
Decrypting the Traffic

- Every user has a unique PTK (pairwise transient key)
- Attacker obtains PMK but not PTK for each user
- Need to capture handshake for that specific user to get PTK
 - Force client to disconnect
 - Then watch / capture re-connection

KRACK (Key Reinstallation attack)

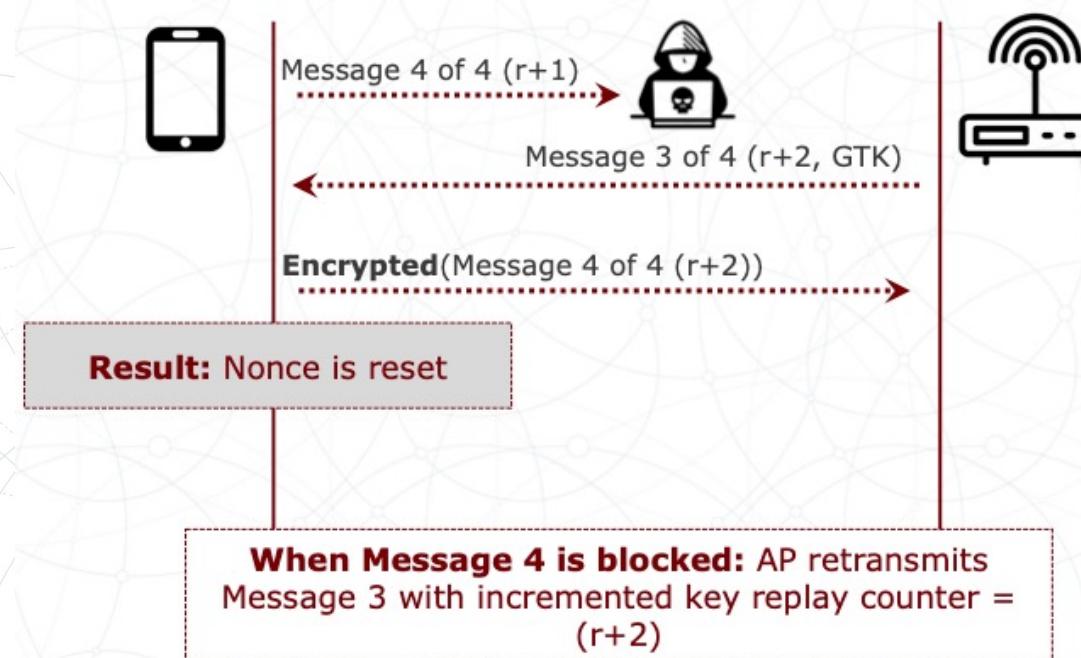
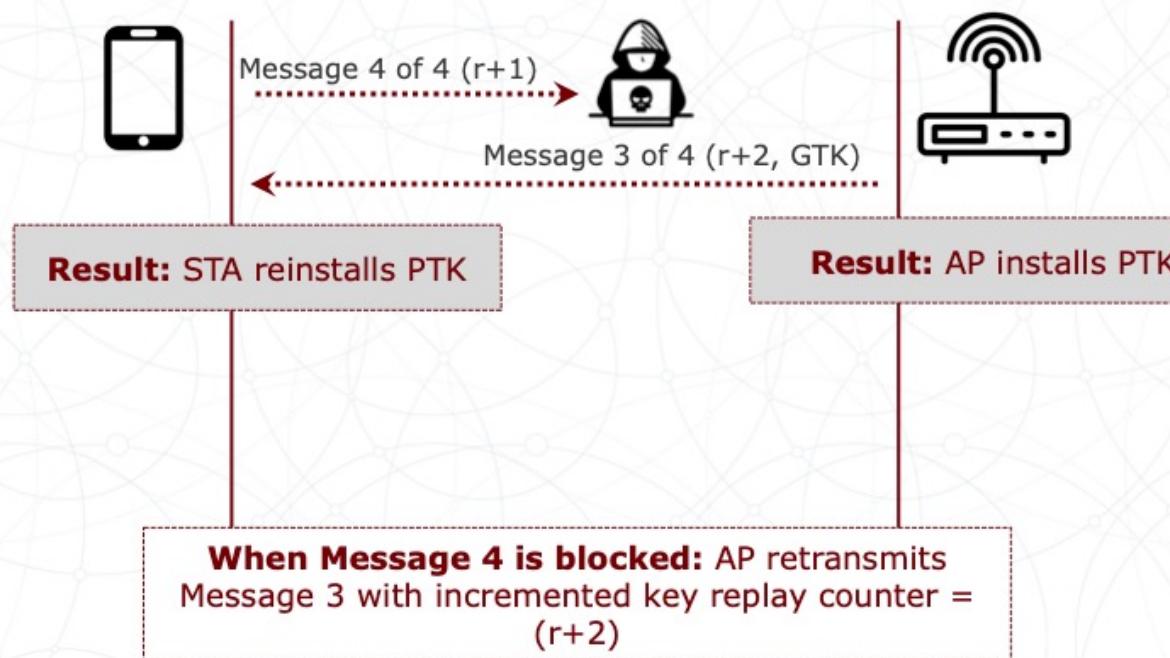
- High Level Idea: attack targets WPA2 four-way handshake protocol flaw specification that allows for third message to be sent repeatedly without changing encryption key
- Discovered in 2016;
 - 14 years after WPA2 was certified by WiFi Alliance
- Attack Discussed in: Vanhoef, Mathy, and Frank Piessens. "Key reinstallation attacks: Forcing nonce reuse in WPA2." Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017.

KRACK (Key Reinstallation attack)



KRACK (Key Reinstallation attack)

- Attacker blocks message 4
 - Re-transmit message 3 with increasing ‘r’



KRACK: Example Scenario

- KRACK allow an adversary to decrypt a TCP packet, learn the sequence number, and hijack the TCP stream to inject arbitrary data
 - Without knowing the password of WiFi
 - This enables one of the most common attacks over Wi-Fi networks: injecting malicious data into an unencrypted HTTP connection

L9: Outline

Bluetooth

Basics

Standards

Device Discovery

Connection Establishment

Protocol Overview

Bluetooth Basics

- Defines 79 channels across the 2.4-GHz ISM band, each channel occupying 1-MHz of spectrum
- Devices hop across these channels at a rate of 1600 times a second (every 625 microseconds)
- This channel-hopping technique is Frequency Hopping Spread Spectrum (FHSS), and the user can achieve a rate of 3 Mbps of bandwidth across 100 meters
 - FHSS provides robustness against noisy channels by rapidly changing frequencies
 - Later revisions of the standard have added support for adaptive hopping, which allows noisy channels to be detected and avoided all together

Device 1 and 2 form a piconet; they are channel hopping in step with each other.

Device 1 (master)

| | | | | | | | | | | | |
|---|---|---|---|---|---|----|---|---|----|---|----|
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |
| 1 | 8 | 5 | 4 | 7 | 6 | 10 | 2 | 9 | 12 | 3 | 11 |

Device 2 (slave)

Device 3

| | | | | | | | | | | | |
|---|---|---|----|---|---|---|---|----|---|---|---|
| 6 | 4 | 5 | 10 | 1 | 2 | 6 | 3 | 11 | 8 | 9 | 7 |
|---|---|---|----|---|---|---|---|----|---|---|---|

Device 3 is not part of the piconet; it is unaware of the channel-hopping sequence in use by the other devices.

Bluetooth: Device Discovery

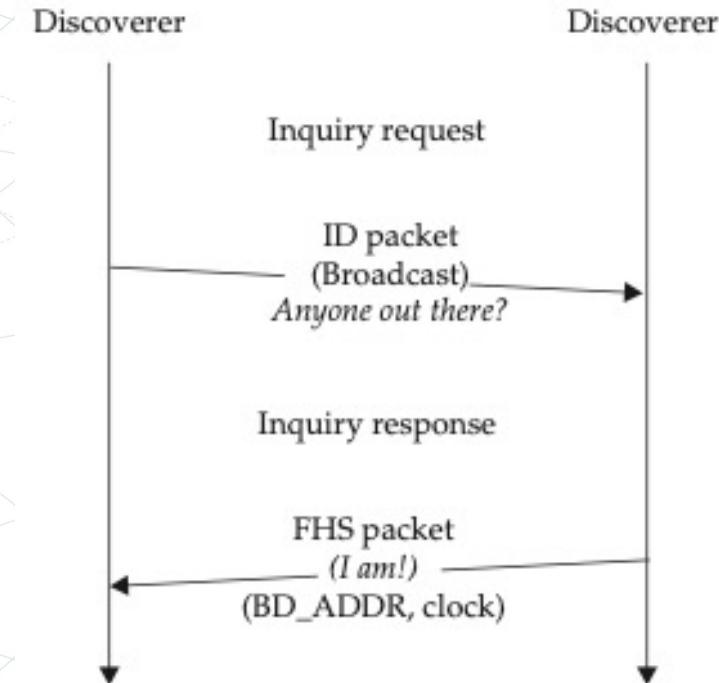
- Technically, discoverable devices are devices that enter the inquiry scan substate
 - These devices respond to inquiry requests
- On the other end of this frequency-hopping dance is the device doing the discovery
 - This device has no knowledge of its potential peer channels at the moment
 - Thus, it must transmit discovery requests (ID packets) into the air in a (mostly) random pattern, hoping to cross paths with a device on the same channel at the same time

Bluetooth: Device Discovery

- Even assuming that the discoverable device sees this request, how is it supposed to respond?
 - It needs to transmit a response, but can't be sure what channel its discovering buddy wandered off to
 - Therefore, it will start responding on a lot of channels, on the assumption that its discovering device will see one of the responses
- The protocol has a few optimizations to help devices find each other, and there is an upper-bound on the time it takes for this entire exchange to happen (10.24 seconds), but the process still seems remarkably difficult
 - If you've ever wondered what your computer was doing when it was looking for your cell phone or Bluetooth mouse the first time, this is it

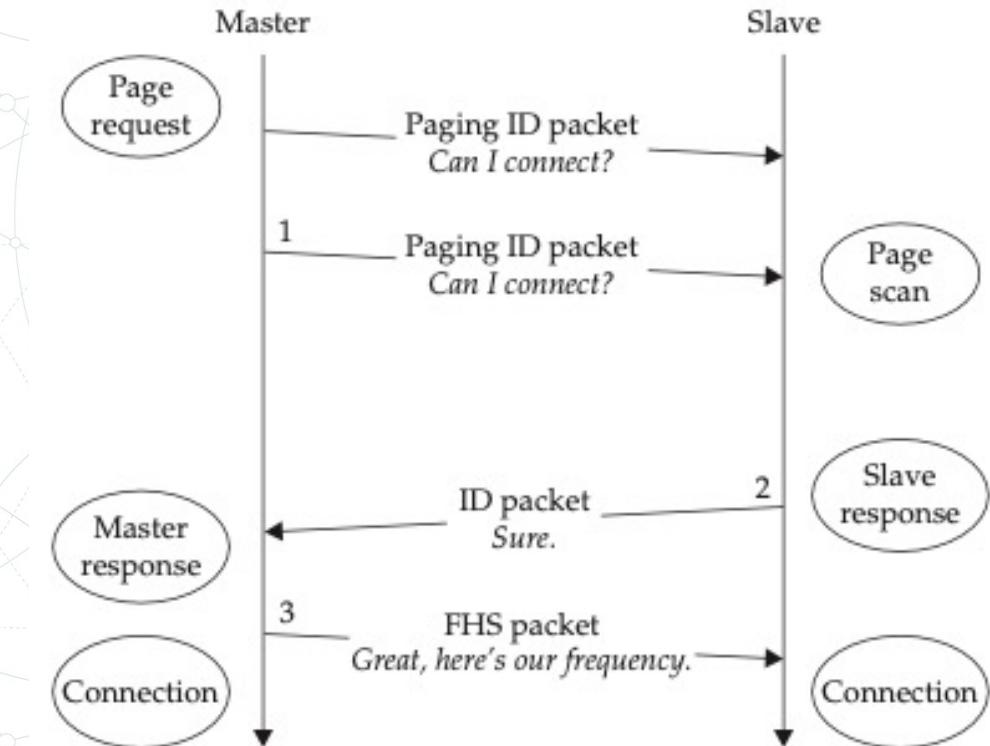
Bluetooth: Device Discovery

- A device is said to be nondiscoverable if it simply ignores (or doesn't look for) inquiry requests
 - The only way to establish a connection to one of these nondiscoverable devices is to determine its Bluetooth device address (BD_ADDR) through some other means
 - Once the discoverer has the BD_ADDR and clock of the discoveree, it can then attempt to initiate a connection



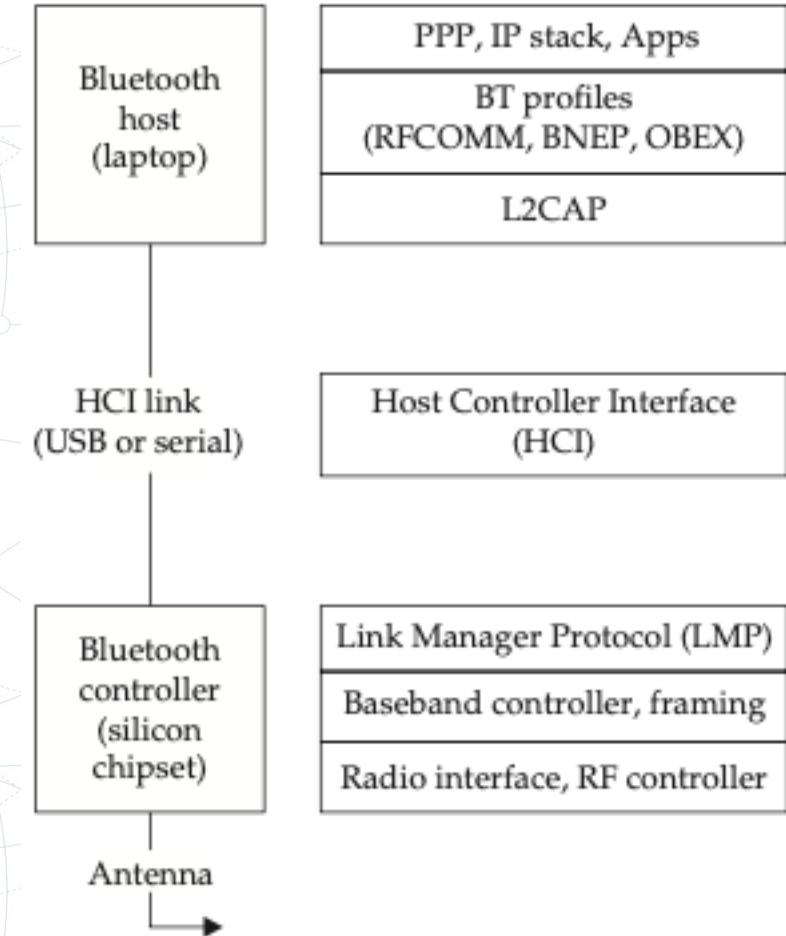
Bluetooth: Connection Establishment

- The diagram covers this in some detail
 - The most important thing to remember about “paging” or connection establishment is that in order to establish a connection you must know the target’s BD_ADDR, and that device must be interested in accepting connections



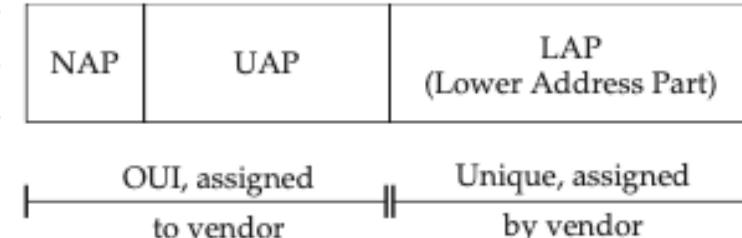
Bluetooth Protocol Overview

- The organization of layers in the Bluetooth stack and where each layer is typically implemented:
 - The controller is responsible for frequency hopping, baseband encapsulation, and returning the appropriate results back to the host
 - The host is responsible for higher-layer protocols
 - The Host Controller Interface (HCI) link is used as the interface between the Bluetooth host (your laptop) and the Bluetooth controller (the chipset in your Bluetooth dongle)



Bluetooth Device Addresses (BD_ADDR)

- Bluetooth devices come with a 6-byte 802-compliant MAC address, similar to that of Ethernet and 802.11 devices
- In Bluetooth, these devices have a little more structure to them and are rarely transmitted over the air
 - As outlined previously, the lower 24 bits of a BD_ADDR is expanded into a 64-bit sync word, which is, in turn, transmitted in the access code of a Bluetooth baseband packet



L10/11: Outline

Bluetooth

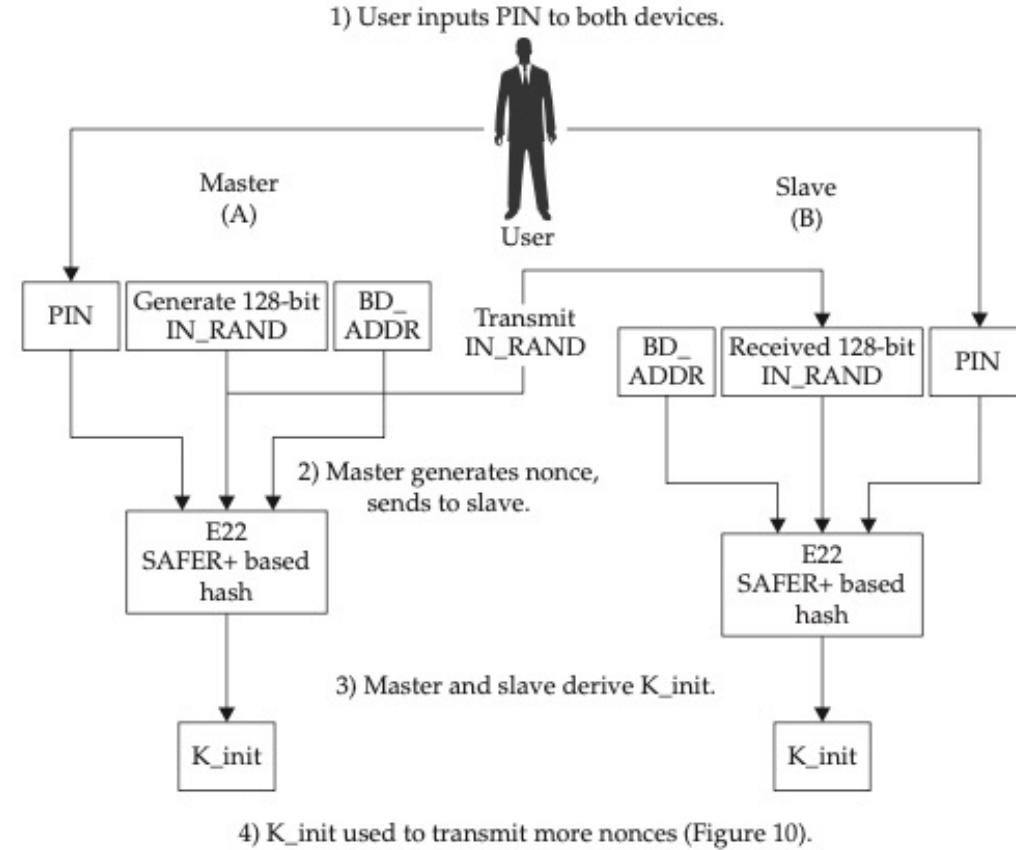
Encryption and Authentication

Traditional Pairing

Secure Simple Pairing

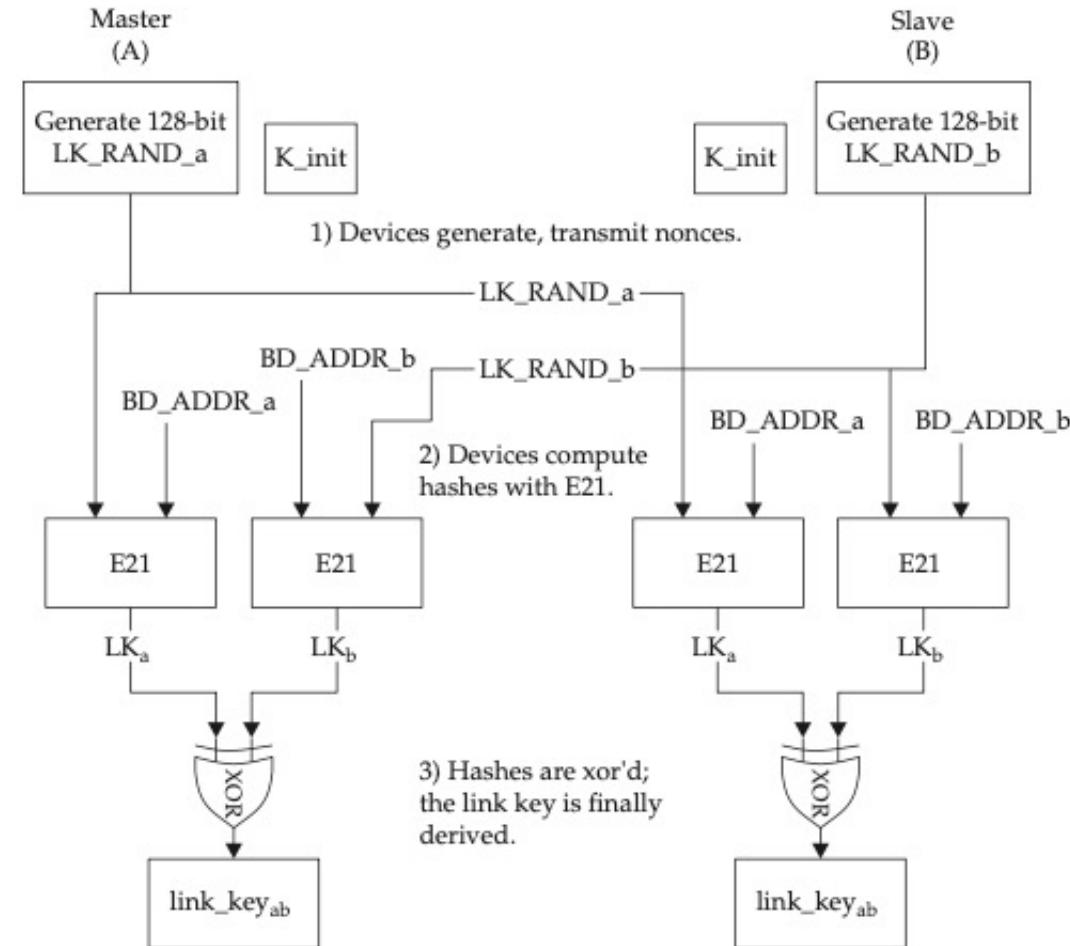
Traditional Pairing Process: Link Key Creation

- Was superseded by the release of Bluetooth 2.1 by Secure Simple Pairing
 - The tradition process is still used in many Bluetooth devices
- When two devices connect for the first time, a link key is derived from a BD_ADDR, a PIN code, and a random number
 - Once both sides in the exchange have created K_init, they use it to generate a stronger link key



Traditional Pairing Process: Derivation of the Link Key

- Both devices have generated a 128-bit link key
- This link key will be stored alongside its peer's BD_ADDR, either on disk or in nonvolatile storage
- When the two devices want to communicate, they will go through a handshaking process
- The link-key generation will only happen each time the devices are paired (typically once)
 - For later authentication purposes, possession of this link key is used, not the PIN



Traditional Pairing Process: Security

- If an attacker observes the traditional link-key generation step, as well as the link-key authentication step, all he needs to do to derive the link key is to brute-force the PIN until he generates the observed value of SRES
- Once he gets the SRES to match, he possesses both the PIN and the link key

Secure Simple Pairing (SSP)

- The problem with the traditional pairing scheme is that a passive attacker who observes the pairing exchange can typically brute-force the PIN in seconds
- SSP attempts to prevent a passive observer from retrieving the link key
 - SSP accomplishes this by using public key crypto, specifically Elliptic Curve Diffie-Hellman
- A Diffie-Hellman key exchange allows two peers to exchange public keys and then derive a shared secret that an observer will not be able to reproduce
- The resulting secret key is called the *DHKey*
 - Ultimately, the link key will be derived from the *DHKey*

Secure Simple Pairing: Overview

- Capabilities Exchange
- Key Exchange
- Authentication
- Link-key creation

1) Capabilities exchange

Capabilities exchange:
A and B exchange IO capabilities
(this will determine the auth technique used)

2) Key exchange

Public key exchange:
A transmits PK_A
B transmits PK_B

DHKey derived by both parties
(passive observers cannot compute this)

3) Authentication

Just Works, Numerical Comparison
Out of Band, or Passphrase authentication

Devices exchange confirmation values,
compare results

4) Link-key creation

Link-key derived
 $LK = F2(DHKey, Nmaster, Nslave, "btlk", BD_ADDR_m, BD_ADDR_s)$

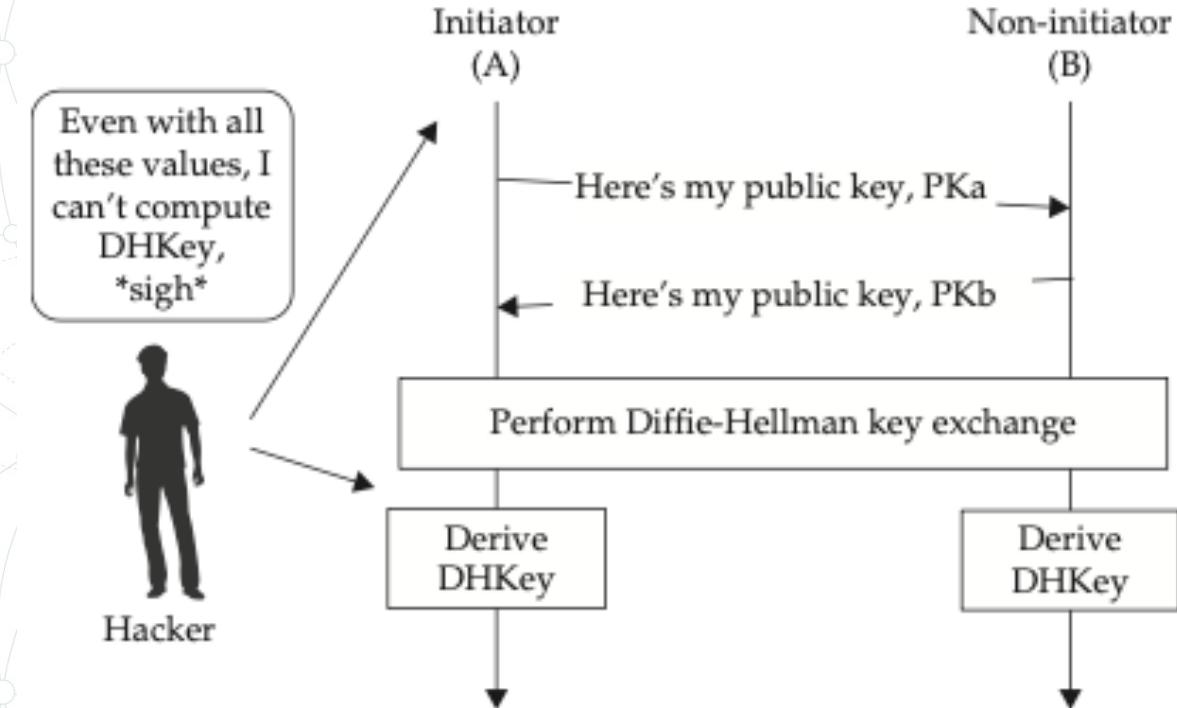
Secure Simple Pairing: Capabilities Exchange

- Tables summarizing the capability information exchanged by devices:

| Capability | Description |
|-------------------|---|
| No input | Device cannot indicate yes or no, lacking any input capability. |
| Yes/No | Device has a button that the user can activate to indicate yes or no. |
| Keyboard | Device can input values 0–9, as well as indicate yes or no. |
| Capability | Description |
| No output | Device cannot display a 6-digit number. |
| Numeric output | Device can display a 6-digit number. |

Key Exchange

- Is the easiest phase
- The devices simply transmit their public keys to each other and then perform a Diffie-Hellman key exchange operation to derive DHKey
- An attacker who captures this entire exchange will still be unable to compute DHKey, due to the cryptographic security of the Diffie-Hellman protocol



Authentication: Just Works

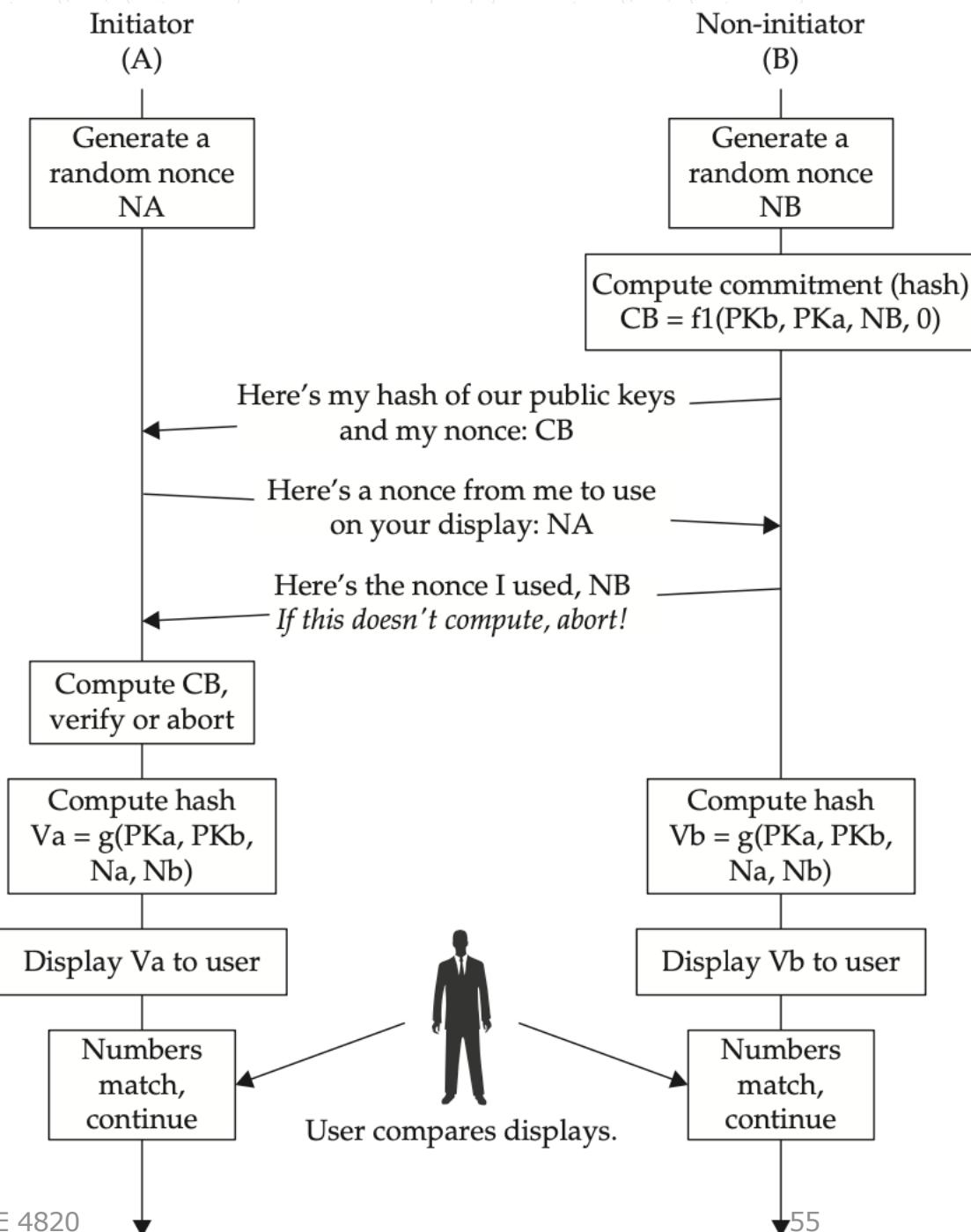
- This mode runs the same protocol as numeric comparison, but the user does not actually make a comparison
- This protocol is secure against passive attacks (due to the DHKey exchange), but an active MITM attack can succeed by sending both A and B its own public key and nonces at the right time

Authentication: Numeric Comparison

- If it is used, if both devices have sufficient IO capabilities to display a six-digit number to the user
- When this technique is used, each device computes a hash of the exchanged public keys, as well as two more nonces
- The devices display six digits of this hash, and the end user is expected to verify that the displayed hashes match and select Yes or No

Authentication: Numeric Comparison

- Both devices generate and exchange nonces (a number used once), and then compute a hash of these nonces as well as the public keys exchanged previously
 - This hash is displayed to the user in the form of a six-digit number
 - The user is supposed to compare these numbers to verify that they match



Authentication: Passkey Entry

- Passkey entry allows Bluetooth devices to authenticate each other by verifying they both possess a shared secret
- This secret can be entered into both devices or generated on one and entered into the other
- A typical use-case for this is a keyboard and computer
 - The computer generates a random PIN, and the user inputs it through the keyboard
 - When implemented poorly, this technique can be severely compromised

Passively Attacking Passphrase Authentication

- An attacker who can observe the entire exchange can trivially recover the passphrase
- For every bit in the passphrase, the attacker will know the following values:
 - PK_a, PK_b : Public keys observed during the initial public key exchange
 - CA_i, Cb_i : The commitment hashes for the i th bit
 - NA_i, Nb_i : The nonces used as input to the above commitment hashes

Passively Attacking Passphrase Authentication

- In this regard, SSP is actually worse than the PIN-based scheme used in traditional pairing
 - Under the old system, if the user inputs a 32-bit pin, an attacker will need to compute 2^{32} hashes (worst case) before finding it
 - In the current system, the attacker will need to compute at worst 32 hashes
- Yet, an attacker who learns the passphrase still does not know the link key, because it will be derived from the *DHKey*, which an attacker cannot derive
 - Once the attacker has recovered the passphrase, he can try to convince the devices to pair with him

Actively Attacking Passphrase-Protected Devices

- Another attack can be levied against a passphrase-protected device
 - Assume that a device has a 32-bit passphrase (for simplicity)
 - Also assume that this device can be placed in pairing mode repeatedly
 - Finally, assume that you would like access to this device, and you don't have the link key or passphrase
- All you need to do is randomly choose a bit for $ra[i]$, starting with $ra[0]$ and working your way up
 - If the device continues, then you chose correctly
 - If the device aborts, you know that you chose incorrectly and, therefore, will choose correctly the next time

Link-Key Derivation

- Once the authentication phase is complete, the devices are convinced that the DHKey was negotiated with the desired party
 - Now it's time to use it for creating the link key
- Link-key derivation is simple at this point because all of the authentication is out of the way
- The link key is derived from the DHKey using the following hash:
 - $LinkKeyAB = f2(DHKey, N_{master}, N_{slave}, "btlk", BDADDR_{master}, BD_ADDR_{slave})$

SSP Summary

- Despite attacks on passphrase authentication just outlined, on average Bluetooth security is enhanced via the use of SSP
- Unless there is a serious cryptographic breakthrough, a passive attacker should not be able to recover the link key since this would require a passive attack against the Diffie-Hellman key exchange
- The best passive attack known to date involves the misuse of the passphrase authentication scheme with a static key
 - An attacker who observes this can trivially compute the passphrase used for the pairing session

SSP Niñó (NoInputNoOutput) Attack

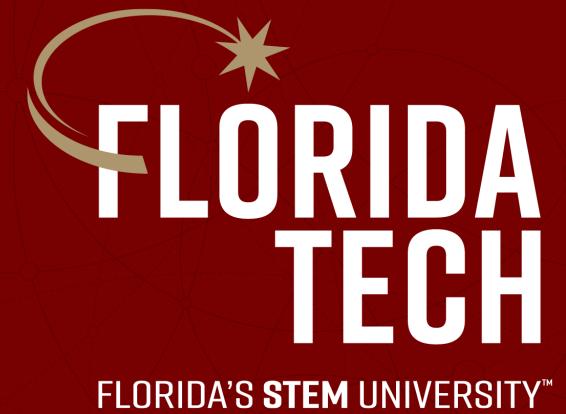
- When two devices pair, they negotiate the IOCapability information to identify a suitable authentication mechanism that is supported by both devices
 - This exchange happens before the link key is derived and, as such, is not a protected exchange
- This attack leverages this SSP IOCapabilities exchange deficiency to manipulate one or more devices, forcing the victim device to “dumb-down” its selected authentication mechanism to the Just Works technique
 - Once one or more devices are forced to this weaker authentication method, the attacker can eavesdrop on any data sent between devices

SSP Niñó Attack

- First, the attacker must force two devices to re-pair
 - One option is to launch a denial-of-service (DoS) attack against the Bluetooth devices in the area, designing a transmitter that hops along with the piconet master and jamming on each channel during the frequency-hopping exchange
 - A second option is to leverage a wide-band jammer that can jam all
- 79 Bluetooth channels simultaneously, ceasing all Bluetooth communication within range of the attacker
 - Once the end-user becomes frustrated with the lack of communication between two Bluetooth devices, the attacker would hope that the user attributes the failure to the devices themselves and attempts to repair the devices to resolve the issue, at which time the attacker would stop the DoS attack

SSP Niñó Attack

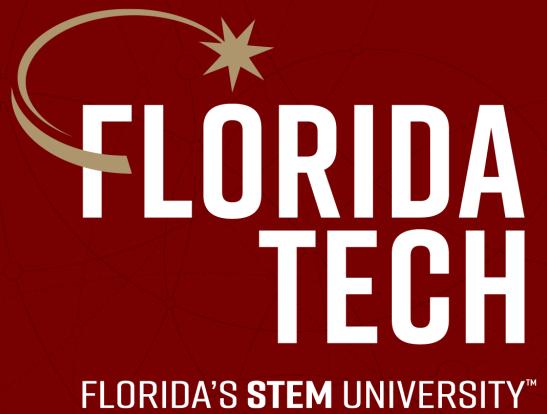
- Immediately before the devices re-pair, the attacker would impersonate the BD_ADDR and friendly name of both devices and implement a MITM attack, brokering the authentication exchange between the victim devices
- Instead of allowing the legitimate advertised IOCapabilities to pass between devices, however, the attacker would indicate to the responder that the initiator only supports the NoInputNoOutput capability, and vice versa, effectively dumbing-down the connection exchange and leaving the Just Works authentication method as the only plausible method



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

13. Software Defined Radios

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

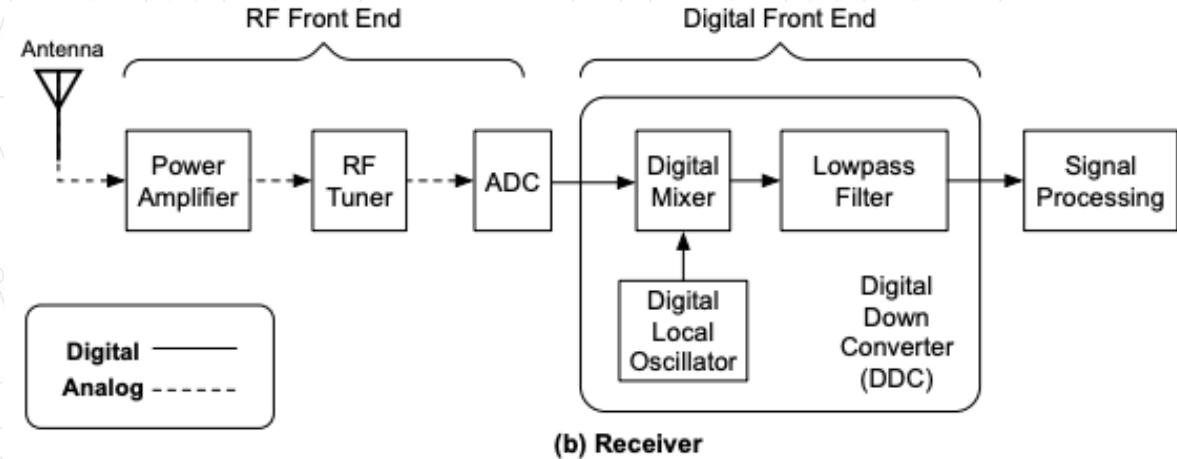
SDR

Software Defined Radios

- The process of reverse engineering a radio signal to understand elements such as frequency, modulation, encoding, and protocol in order to intercept or replicate the signal
 - For ex. replace sound waves with radio waves
 - By using special 'radio soundcard', you can receive and transmit arbitrary signals
- Has redefined wireless hacking
 - Instead of being limited by black box radios, unfettered access to RF
 - Access to Radio modules/protocols that were obscured

SDR Architecture

- Three main parts:
 - Radio Frequency (RF) amplifier, the tuner, and the Analog-to-Digital Converter (ADC)
 - RF amplifier is responsible for boosting weak signals
 - Tuner; select portion of radio spectrum to analyze
 - Like tuning on an old radio
 - ADC; sampling (analog waveform to stream of digital numbers)
 - Antennas; isotropic (any direction) or directional



Choosing SDR

- Sample Rate/ Bandwidth:
 - Maximum bandwidth you are able to view simultaneously
 - Measured in MSPS (Millions of samples per second)
 - For 802.11b / g, at least 20 MSPS, however 2 MSPS is fine for most
- Dynamic Range/ ADC Resolution:
 - Similar to contrast ratio and dots-per-inch on TVs
 - Higher ADC lets you view loud and quiet signals together

Choosing SDR

- Transmit Capability:
 - Some SDRs are receive only
 - Full (transmit / receive at the same time) or half duplex
- Tuner Range
 - What frequencies you are able to receive

RTL-SDR

- Digital Video Tuner converted into SDR
- Receiver only
- Frequency Range: 50MHz to 1.7GHz
- ADC: 8 bits
- Popular among hobbyist due to low cost
- RTL-SDR Source block available in Gnuradio

<https://amzn.to/321mYwB>



RTL-SDR Source

Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 32k
Ch0: Frequency (Hz): 100M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

HackRF

- Wide Band Software Defined Radio (SDR)
- Half-duplex transceiver
- Frequency Range: 10MHz to 6GHz, ADC: 8 bits
- Power: 30 mW - 1 mW (depending on band)
- Popular among researchers due to low cost
- Open sourced hardware

<https://greatscottgadgets.com/hackrf/one/>



Software: GNU Radio

- Gnuradio's modular design allows us to process and prepare signals to test methods of intercepting and replicating the signal
- Helpful blocks: hardware sinks / sources, file sink / sources, modulators, signal multipliers, signal sources, vector sources, variables, and QT Gui sinks

Software: Universal Radio Hacker

- Discussed in Pohl, Johannes, and Andreas Noack. "Universal radio hacker: a suite for analyzing and attacking stateful wireless protocols." 12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18). 2018.
- Complete suite for wireless protocol investigations with native support for many common SDRs
- The URH allows easy demodulation of signals combined with an automatic detection of modulation parameters to identify the bits and bytes that fly over the air

<https://github.com/jopohl/urh>

Software: Universal Radio Hacker

- URH's protocol reverse-engineering:
 - You can either manually assign protocol fields and message types or
 - Let URH automatically infer protocol fields with a rule-based intelligence
- Finally, URH entails a fuzzing component aimed at stateless protocols and a simulation environment for stateful attacks
- Some examples of using URH to decode signals includes reversing: restaurant pagers, cloning car-key remotes, wireless keyboards, or intercepting weather signals

<https://github.com/jopohl/urh>

How to try this in wild?

- Finding a target
 - For ex., key fob, garage opener, wireless mouse, and RC car
- Device reconnaissance;
 - Figure out what frequency it uses
- Finding and capturing signal
- Replaying/ changing



Device reconnaissance

- All RF devices subject to FCC (Federal Communications Commission) Certificate require registration with the FCC
- They are given an FCC ID
 - Usually it is printed somewhere on the device
- Looking up the FCC ID yields information about the RF signal to include the frequency

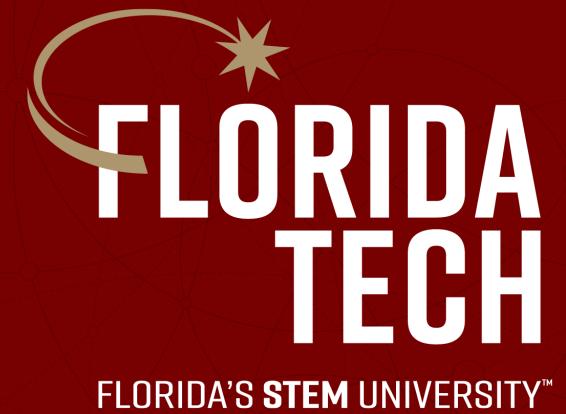
2 results were found that match the search criteria:

Grantee Code: **B8Q** Product Code: **ACSCT**

Displaying records 1 through 2 of 2.

| View Form | Display Exhibits | Display Grant | Display Correspondence | Applicant Name | Address | City | State Country | Zip Code | FCC ID | Application Purpose | Final Action Date | Lower Frequency In MHz | Upper Frequency In MHz | |
|---|--------------------------------|---|---|---|--------------|----------|---------------|---------------|--------|---------------------|--------------------|------------------------|------------------------|-------|
|  | Detail Summary |  |  | The Genie Company a Division of Overhead Door Corporation | 1 Door Drive | Mt. Hope | TX | United States | 44660 | B8QACSCT | Original Equipment | 01/08/1997 | 390.0 | 390.0 |

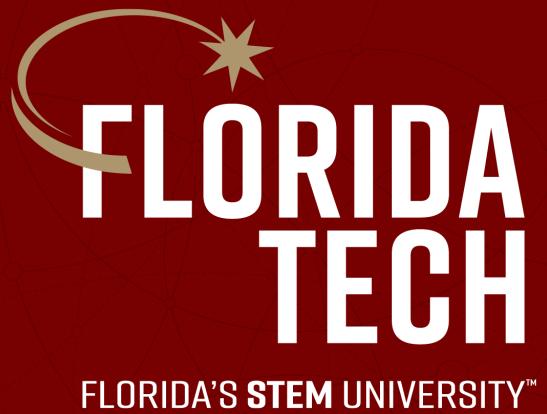
<https://www.fcc.gov/oet/ea/fccid#:~:text=FCC%20ID%20numbers%20consists%20of,string%20representing%20the%20Grantee%2FApplicant.>



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

14. Zigbee Overview

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Zigbee

History

Layers

Security

Recall: Software Defined Radios

- The process of reverse engineering a radio signal to understand elements such as frequency, modulation, encoding, and protocol in order to intercept or replicate the signal
 - For ex. replace sound waves with radio waves
 - By using special ‘radio soundcard’, you can receive and transmit arbitrary signals
- Has redefined wireless hacking
 - Instead of being limited by black box radios, unfettered access to RF
 - Access to Radio modules/protocols that were obscured

Recall: How to try this in wild?

- Finding a target
 - For ex., key fob, garage opener, wireless mouse, and RC car
- Device reconnaissance;
 - Figure out what frequency it uses
- Finding and capturing signal
- Replaying / changing



Zigbee

- Set of standards for low-power wireless networking
 - Devices with up to 5 years battery life
- Low data-rate transfers, short-range, persistent-powered network coordinators/routers, and simple protocol stack
- Found in industrial and home applications
 - For ex., home theater remote controls to hospital patient monitoring systems

Zigbee as Wireless Standard

- Strong position to be the wireless tech for IoT
 - Already used in Google Nest Smart Thermostat, Philips Hue led light bulbs, Comcast Xfinity home security router to connect home light switches and other peripherals to public internet
- Why do we need Zigbee?
 - Compared to Wifi and BLE, Zigbee is much simpler protocol with a fully functional stack implemented in 120kb of NVRAM where some vendors claim to make reduced-functionality stacks as small as 40kb

Zigbee as Wireless Standard

- Wireless networks transmit at least 54mbps, BLE 1-3mbps, and Zigbee 20-250kbps
 - Not the right protocol for high-speed data transfers
- Wifi devices; relatively short battery life
 - Bluetooth relatively comparable to Zigbee
- Deployed mostly for the home automation;
 - Connectivity among home control systems such as electrical appliances, lighting controls, home security, etc.

Zigbee Overview

| Solution | Description |
|------------------------------------|---|
| Network Protocol | Zigbee PRO 2015 (or newer) |
| Network Topology | Self-Forming, Self-Healing MESH |
| Network Device Types | Coordinator (routing capable), Router, End Device, Zigbee Green Power Device |
| Net. Size (theoretical # of nodes) | Up to 65,000 |
| Radio Technology | IEEE 802.15.4-2011 |
| Frequency Band / Channels | 2.4 GHz (ISM band) 16-channels (2 MHz wide) |
| Data Rate | 250 Kbits/sec |
| Security Models | Centralized (with Install Codes support) Distributed |
| Encryption Support | AES-128 at Network Layer AES-128 available at Application Layer |
| Communication Range (Avg) | Up to 300+ meters (line of sight) Up to 75-100 meter indoor |
| Low Power Support | Sleeping End Devices Zigbee Green Power Devices (energy harvesting) |
| Legacy Profile Support | Zigbee 3 devices can join legacy Zigbee profile networks. Legacy devices may join Zigbee 3 networks (based on network's security policy) |
| Logical device support | Each physical device may support up to 240 end-points (logical devices) |

Zigbee History

- First Zigbee specification Zigbee-2004
 - Attractive to organizations in which rival wireless protocols were not a good fit
- Zigbee-2006; group addressing capabilities where one device sends message to multiple clients with a single frame
 - Simplifying the process of developing cross platform compatible apps over Zigbee

Zigbee History

- Zigbee Pro in 2007 defined enhanced security features and ability to send large messages through data fragmentation
- Zigbee in 2012; multihop mesh network range capability
 - Also, 'Green Energy' feature where devices can join and interact with Zigbee network without the need for an outside power source (harness energy from other 'green' ways)

ZigBee Layers

Defined by Zigbee Alliance

Defined in IEEE 802.15.4
(Low-rate wireless personal area network)

Application Layer (APL)

App.
Framework

App Support
(APS)

Zigbee Device
Option
(ZDO)

Network Layer
(NWK)

Medium Access Control Layer
(MAC)

Physical Layer
(PHY)

<https://csa-iot.org/all-solutions/zigbee/>

Zigbee: Physical Layer

- Physical layer defined by IEEE 802.15.4 Specification
- To avoid interference, uses Direct Sequence Spread Spectrum or Parallel Sequence Spread Spectrum
- Operates on 915 MHz (North and South America), 866 MHz (Europe), or 2.4 GHz (worldwide)

| Channel | Width | Freq | Data |
|---------|---------|---------|----------|
| 0 | 600 KHz | 868 MHz | 100 Kb/s |
| 1-10 | 2 MHz | 915 MHz | 250 Kb/s |
| 11-26 | 5 MHz | 2.4 GHz | 250 Kb/s |

Zigbee: MAC Layer

- Includes functionality needed to build extensive Zigbee networks
 - Including the design of device interconnect topologies, device roles, packet framing, and network association / disassociation
 - Each device has set of capabilities defined by operational roles:
 - Trust Center, Coordinator, Router, and End Device

Zigbee: MAC Layer

- **ZigBee Trust Center (TC):**

- Fully functional Zigbee device (FFD) responsible for the authentication of devices that join Zigbee network
- When a device attempts to join, nearest router notifies the TC that a device has joined
 - TC instructs router to authenticate or terminate new node's connection

Zigbee: MAC Layer

- **ZigBee Coordinator (ZC):**

- Fully controls the Personal Area Network (PAN)
- Performs replay messages on behalf of other devices
- Allow other Zigbee devices to join them and participate in the network
- Selects network channel to reduce interference
- Issues 16-bit device addresses

Zigbee: MAC Layer

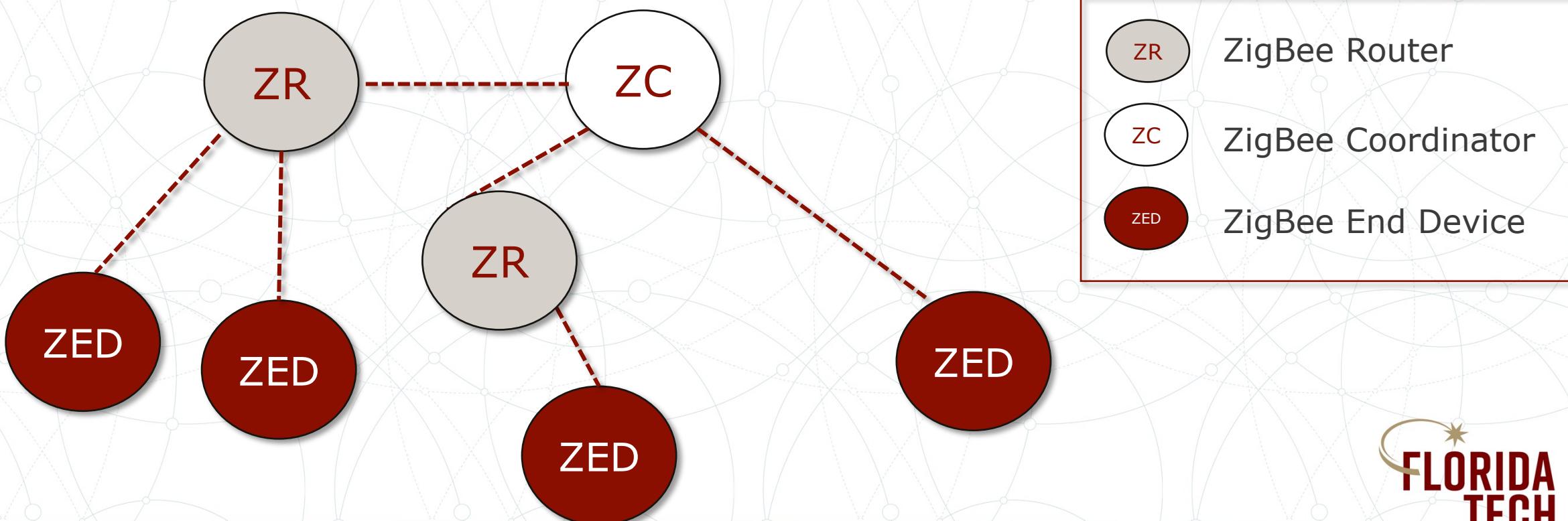
- **ZigBee Router (ZR):**
 - Similar to ZC but defers network management to ZC;
 - Allows devices to join network
 - Performs replay messages on behalf of other devices

Zigbee: MAC Layer

- **ZigBee End Device (ZED):**
 - Participates in network but cannot relay ZigBee frames for other devices;
 - Directly connected to ZR or ZC, cannot connect to another ZED

Example Zigbee Network

- One ZC for the network, additional ZRs



Example Zigbee Network

- Can be deployed in a star or mesh topology
- ZRs are essential to build and bridge traffic to and from downstream nodes (ZEDs or other ZRs), whereas ZC manages network operation
- Period of inactivity (aka sleep mode) where ZEDs can shut down all transceiver functions for a period of time (microsecs to hours)
 - Can wake up/ transmit anytime and go back sleep afterwards
 - Mechanism to allow long battery life
 - ZRs and ZC are generally deployed with persistent power source since they need to be ready to receive anytime

IEEE 802.15.4 MAC Frame Format

| Frame Control | Seq. No. | Dest. Pan ID | Dest. Addr | Source Pan ID | Source Addr | Aux Sec Hdr | Payload | FCS |
|---------------|----------|--------------|------------|---------------|-------------|-------------|---------|-----|
|---------------|----------|--------------|------------|---------------|-------------|-------------|---------|-----|

- Specified in IEEE 802.15.4 Spec
- Frame Control tells what type of frame (beacon, command, data, ack...)
- Sequence Numbers enables in-order delivery
- Dest/Src Pan ID/ Addr handles delivery of frame
- Auxiliary Security Header optionally implements security
- FCS is CRC 16-bit checksum of the mac layer frame

Zigbee MAC Frame Types

- Beacon Frames – used for network discovery; scan the network for ZC or ZR
- Command Frames – same as 802.11 management (association/disassociation)
- Data Frames – same as 802.11 data; exchange data b/w devices
- Acknowledgement Frames – acknowledge frames that were received

ZigBee Network Layer (NWK)

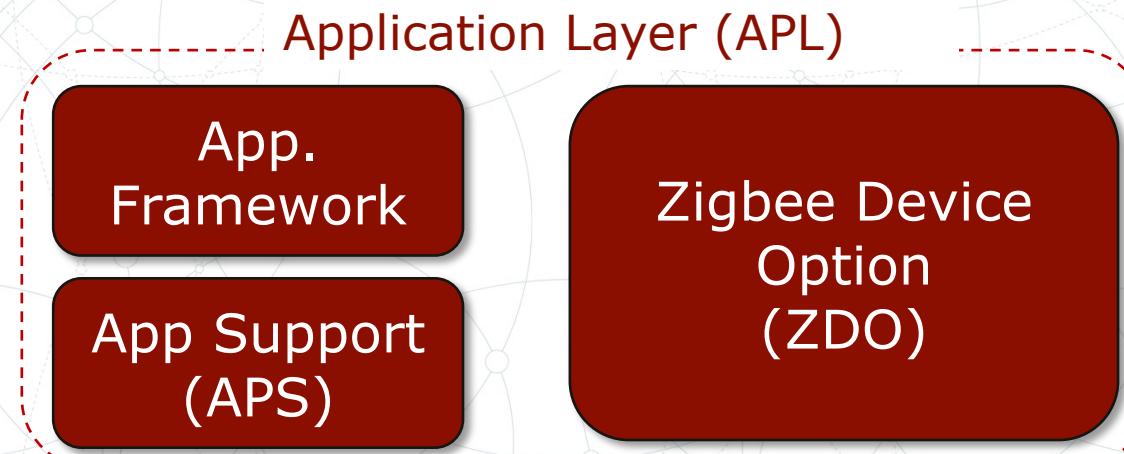
- ZigBee Network Layer (NWK) defined by ZigBee Alliance
 - Responsible for network formation, address allocation, and routing
- Network formation; FFD establishes itself as network coordinator
- Through device discovery, the coordinator must;
 - Select suitable channel to avoid interference
 - Choose random Pan ID that doesn't conflict with nearby Pans
 - Select and issue 16-bit device addresses for devices

ZigBee Application Layer (APL)

- Specifying operation and interface for application objects that define Zigbee device's functionality
- Application objects are developed by Zigbee Alliance as standard functionality profiles, or
 - By manufacturers for proprietary device functionality using the APL as mechanism to communicate with lower layers of Zigbee stack

ZigBee Application Layer

- ZDO: provides core functionality
 - Setting roles (ZC, ZR, ZED)
 - Encryption
 - Network management (association)
- APS: provides functionality to application profiles (such as reliable data delivery)
- Application Profiles: Define actual functionality of devices
 - ZigBee Link Lighting (ZLL)
 - ZigBee Home Automation (AHA)



Zigbee Security

- AES encryption, device and data authentication using a network key
- Two operational modes:
 - Standard; TC authorizes devices through the use of ACL (Access control list), each device uses a single shared key
 - High Security; TC keeps track of all encryption and authentication keys used on the network, enforcing policies for network authentication and key updates
 - If TC fails, no device will be permitted to join the network

Zigbee Security Design Rules

- Each layer originates a frame is responsible for securing it
 - If APL requires data to be secure, APL will protect the data (can be both at NWK as well)
 - If protection from unauthorized access is required, NWK security will be used on all frames following association and key derivation

Zigbee Security Design Rules

- An open trust model is used within a single device where key reuse is permitted between layers (NWK and APL)
- End-to-end security is accommodated
 - Only source and destination can understand (not the TC)
 - Same security level must be used by all devices in the network and by all layers of device

Zigbee Encryption

- Uses 128-bit AES
 - Assumed to be strong security
 - Could be leveraging AES in an insecure manner
 - How?
- Three types of keys to manage security;
 - Master key, network key, and link key

Zigbee Keys

- Master key; optional except the Zigbee Pro stack
 - Used in conjunction with Zigbee symmetric key-key establishment (SKKE) process to derive other keys
- Network key; protect broadcast and group traffic, as well as authenticating to the network
 - Common key among all nodes
 - Can be distributed to a device in plaintext when it joins the network
- Link key: protect unicast traffic between two devices

Zigbee Encryption Keys

- Global Link Key: used by all nodes on the network
- Unique Link Key: used to encrypt communication between a pair of nodes
 - Preconfigured Link Key: used between trust center and a node;
 - Derived prior to joining
 - Trust Center Link Key: used between trust center and a node; distributed to node
 - Application Link Key: used between pair of nodes; distributed to node

Zigbee: Key Provisioning

- Significant challenge; process of provisioning, rotating, and revoking keys on devices
- Zigbee Pro; Administrator can use the SKKE method to derive the network and link keys on devices
 - Requires devices to have master key provisioned on the TC and device joining the network

Zigbee: Key Provisioning

- Key transport; network and link keys are sent in plaintext over the wireless network to the device when it joins
 - Can Easily be intercepted
- Pre-installation; administrator preconfigures all devices with the desired encryption keys at the manufacturing process
 - How to accommodate key revocation and rotation methods?
 - Manual changes to each device to change keys

Zigbee Authentication

- MAC address validation through ACL;
 - A list of authorized devices is maintained on each node
 - Challenging to keep the list up-to-date (memory req.)
- Standard mode; TC grants access by issuing a network key
 - Sent in plaintext
- High security mode; SKKE method to derive the network key
 - 4-way handshake

Zigbee Authentication's Vulnerability

- No mutual authentication is used in standard Zigbee (except high security mode with SKKE)
 - Authenticating node accepts the identity of TC for the delivery of network key without any validity check to verify the identity of the network
 - Easy to impersonate a legitimate network by using the same PAN ID as target, potentially on a different channel

Zigbee Attacks

- KillerBee:
 - Python-based framework for manipulating and penetration testing Zigbee and IEEE 802.15.4 networks
 - Written and tested on Linux, free and open-source
 - Includes support for Scapy
 - Includes a variety of tools including zbwreshark, zbdump, and zbreplay

<https://github.com/riverloopsec/killerbee>

IEEE 802.15.4/ZigBee Security Research Toolkit

www.riverloopsecurity.com

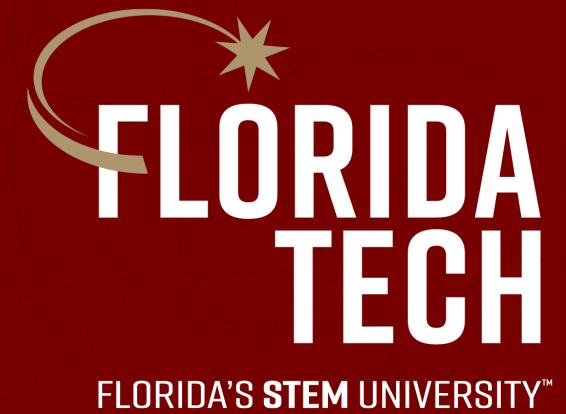
Readme
View license
657 stars
47 watching
210 forks

Releases 5
[3.0.0-beta.2](#) (Latest) on Jul 14, 2021
+ 4 releases

Packages
No packages published

Contributors 29

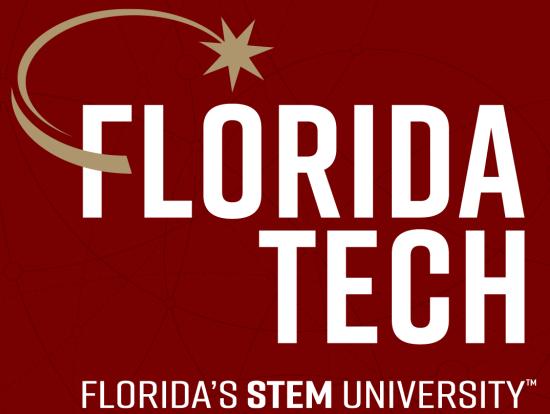
| File | Description | Time Ago |
|-----------------|--|---------------|
| doc | Removed extra level of killerbee folder structure per discussion with... | 8 years ago |
| firmware | Bugfix/cc2531 assorted (#252) | 7 months ago |
| killerbee | Bugfix/cc2531 assorted (#252) | 7 months ago |
| sample | Removed extra level of killerbee folder structure per discussion with... | 8 years ago |
| scripts | implemented goodfet and apimote changes, added mac install instru... | 16 months ago |
| tests | Sewio updates (#233) | 16 months ago |
| tools | Restore APS CMD payload parsing for NWK transport key disclosure | 3 months ago |
| zigbee_crypt | py3 support - merged secureab | 3 years ago |
| .gitignore | Bugfix/cc2531 assorted (#252) | 7 months ago |
| ARCHITECTURE.md | Clean up main readme doc for readability, update other docs with py... | 2 years ago |
| DEVELOPMENT.md | Clean up main readme doc for readability, update other docs with py... | 2 years ago |
| FAQ.md | implemented goodfet and apimote changes, added mac install instru... | 16 months ago |
| LICENSE.txt | Removed extra level of killerbee folder structure per discussion with... | 8 years ago |
| README.md | Goodfet update - bugfixes (#232) | 16 months ago |
| setup.py | Bugfix/cc2531 assorted (#252) | 7 months ago |



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

15. Zigbee Security

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Attacking Zigbee

Recall: Zigbee Overview

| Solution | Description |
|------------------------------------|---|
| Network Protocol | Zigbee PRO 2015 (or newer) |
| Network Topology | Self-Forming, Self-Healing MESH |
| Network Device Types | Coordinator (routing capable), Router, End Device, Zigbee Green Power Device |
| Net. Size (theoretical # of nodes) | Up to 65,000 |
| Radio Technology | IEEE 802.15.4-2011 |
| Frequency Band / Channels | 2.4 GHz (ISM band) 16-channels (2 MHz wide) |
| Data Rate | 250 Kbits/sec |
| Security Models | Centralized (with Install Codes support) Distributed |
| Encryption Support | AES-128 at Network Layer AES-128 available at Application Layer |
| Communication Range (Avg) | Up to 300+ meters (line of sight) Up to 75-100 meter indoor |
| Low Power Support | Sleeping End Devices Zigbee Green Power Devices (energy harvesting) |
| Legacy Profile Support | Zigbee 3 devices can join legacy Zigbee profile networks. Legacy devices may join Zigbee 3 networks (based on network's security policy) |
| Logical device support | Each physical device may support up to 240 end-points (logical devices) |

Recall: ZigBee Layers

Defined by Zigbee Alliance

Defined in IEEE 802.15.4
(Low-rate wireless personal
area network)

Application Layer (APL)

App.
Framework

App Support
(APS)

Zigbee Device
Option
(ZDO)

Network Layer
(NWK)

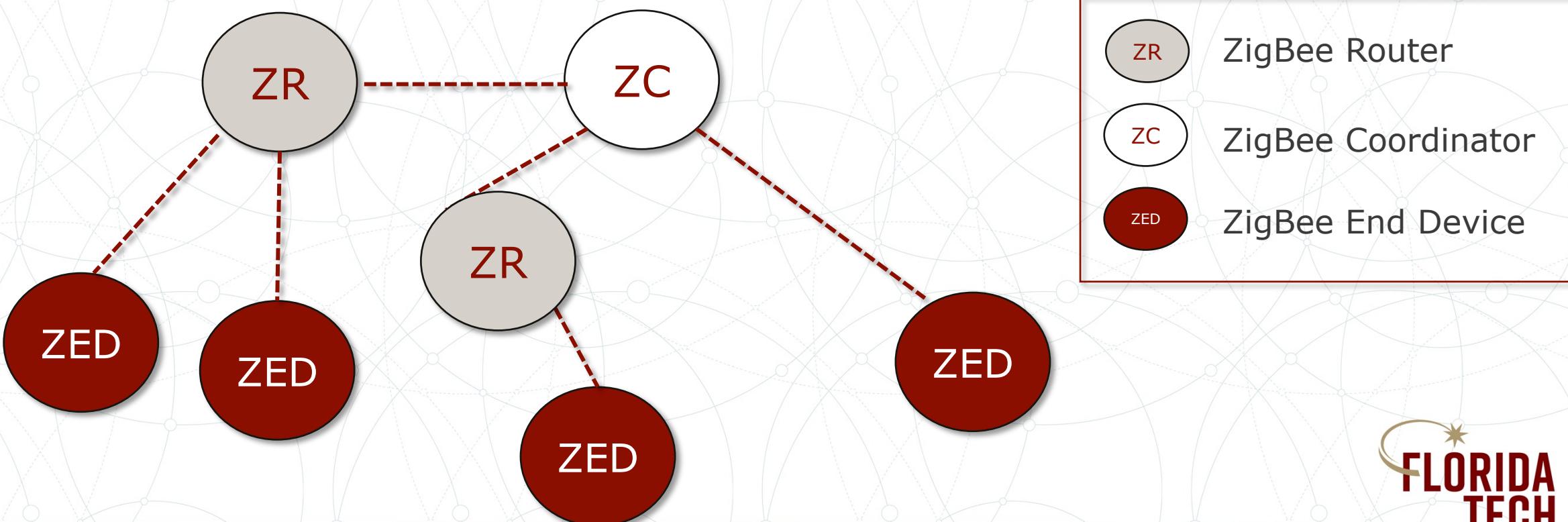
Medium Access Control Layer
(MAC)

Physical Layer
(PHY)

<https://csa-iot.org/all-solutions/zigbee/>

Recall: Example Zigbee Network

- One ZC for the network, additional ZRs



Recall: Zigbee: Key Provisioning

- Significant challenge; process of provisioning, rotating, and revoking keys on devices
- Zigbee Pro; Administrator can use the SKKE method to derive the network and link keys on devices
 - Requires devices to have master key provisioned on the TC and device joining the network

Recall: Zigbee Attacks

- KillerBee:
 - Python-based framework for manipulating and penetration testing Zigbee and IEEE 802.15.4 networks
 - Written and tested on Linux, free and open-source
 - Includes support for Scapy
 - Includes a variety of tools including zbwreshark, zbdump, and zbreplay

<https://github.com/riverloopsec/killerbee>

The screenshot shows the GitHub repository page for 'IEEE 802.15.4/ZigBee Security Research Toolkit'. The page includes a summary of 748740d commits, 398 pull requests, and 5 releases. It also features sections for Readme, View license, 657 stars, 47 watching, and 210 forks. The repository's main content is a list of files and their commit history:

| File | Description | Age |
|-----------------|--|---------------|
| doc | Removed extra level of killerbee folder structure per discussion with... | 8 years ago |
| firmware | Bugfix/cc2531 assorted (#252) | 7 months ago |
| killerbee | Bugfix/cc2531 assorted (#252) | 7 months ago |
| sample | Removed extra level of killerbee folder structure per discussion with... | 8 years ago |
| scripts | implemented goodfet and apimote changes, added mac install instru... | 16 months ago |
| tests | Sewio updates (#233) | 16 months ago |
| tools | Restore APS CMD payload parsing for NWK transport key disclosure | 3 months ago |
| zigbee_crypt | py3 support - merged secureab | 3 years ago |
| .gitignore | Bugfix/cc2531 assorted (#252) | 7 months ago |
| ARCHITECTURE.md | Clean up main readme doc for readability, update other docs with py... | 2 years ago |
| DEVELOPMENT.md | Clean up main readme doc for readability, update other docs with py... | 2 years ago |
| FAQ.md | implemented goodfet and apimote changes, added mac install instru... | 16 months ago |
| LICENSE.txt | Removed extra level of killerbee folder structure per discussion with... | 8 years ago |
| README.md | Goodfet update - bugfixes (#232) | 16 months ago |
| setup.py | Bugfix/cc2531 assorted (#252) | 7 months ago |

Zigbee: Network Discovery

- First assessment is to discover networks within range and enumerate the configuration of devices
 - Simple way; mimic Zigbee network discovery process with Killerbee
- Part of network discovery process in Zigbee Standard, ZDEs transmit beacon request on a given channel
 - All ZR and ZCs receiving beacon -> respond by sending a beacon frame
 - Disclose PAN ID, ZC or ZR source address, stack profile / version, extended IEEE address information
- Using same technique to actively scan for the presence of Zigbee network

Zigbee: Network Discovery

- Killerbee tool zbstumbler (similar to Wifi discovery tool Netstumbler):
 - Channel hops and transmits beacon request frames
 - Every two seconds hopping to a new channel
 - Display useful information from response beacon frames

```
test@test-HP-EliteBook-840-G3:~/killerbee$ sudo zbstumbler
[sudo] password for test:
Warning: You are using pyUSB 1.x, support is in beta.
zbstumbler: Transmitting and receiving on interface '1:11'
New Network: PANID 0xC762 Source 0xE2DA          Stack Profile: ZigBee Enterprise
  Ext PANID: 79:21:70:53:a3:d7:fc:34
  Stack Version: ZigBee 2006/2007
  Channel: 11
New Network: PANID 0xC762 Source 0xFF4F          Stack Profile: ZigBee Enterprise
  Ext PANID: 79:21:70:53:a3:d7:fc:34
  Stack Version: ZigBee 2006/2007
  Channel: 11
```

Zigbee Network Scanning Countermeasure

- Beacon request mechanism is integral to Zigbee
 - Cannot be disabled
 - Attacker can use it freely
- Best countermeasure is to understand the impact and evaluate your own networks to identify the information attacker can gain

Eavesdropping Attacks

- Zigbee networks are mostly not encrypted
 - Extremely easy to eavesdrop
 - Even if it uses encryption
 - Many unencrypted fields useful; MAC header, config of network, node address and PAN ID
 - Might substitute network discovery?
 - Killerbee zbdump -> similar to tcpdump

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|------------------------|-------------|-----------|--------|--|
| 19 | 11.657403 | 0x0000 | Broadcast | ZigBee | 51 | Command, Dst: Broadcast, Src: 0x0000 |
| 20 | 13.098522 | 0x0000 | Broadcast | ZigBee | 28 | Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5 |
| 21 | 15.596531 | | | IEEE 8... | 10 | Beacon Request |
| 22 | 15.724666 | 0x0000 | | ZigBee | 28 | Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5 |
| 23 | 16.493441 | 0x0000 | | ZigBee | 28 | Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5 |
| 24 | 16.621446 | 0x0000 | | ZigBee | 28 | Beacon, Src: 0x0000, EPID: 31:44:80:c9:ca:7f:4a:d5 |
| 25 | 16.749531 | | | IEEE 8... | 5 | Ack |
| 26 | 16.949566 | | | IEEE 8... | 111 | Ack, Bad FCS |
| 27 | 17.021844 | 0x42c9 | 0x0000 | IEEE 8... | 97 | Data, Dst: 0x0000, Src: 0x42c9, Bad FCS |
| 28 | 17.061898 | 0x0000 | 0x42c9 | IEEE 8... | 113 | Data, Dst: 0x42c9, Src: 0x0000, Bad FCS |
| 29 | 17.093841 | 0x0000 | 0x42c9 | ZigBee | 50 | Data, Dst: 0x42c9, Src: 0x0000 |
| 30 | 17.132974 | | | IEEE 8... | 57 | Ack, Bad FCS |
| 31 | 17.164971 | 0x0000 | 0x42c9 | IEEE 8... | 56 | Data, Dst: 0x42c9, Src: 0x0000, Bad FCS |
| 32 | 17.196972 | 0x0000 | 0x42c9 | IEEE 8... | 51 | Data, Dst: 0x42c9, Src: 0x0000, Bad FCS |
| 33 | 17.237021 | 0x0000 | 0x42c9 | IEEE 8... | 108 | Data, Dst: 0x42c9, Src: 0x0000, Bad FCS |
| 34 | 17.309167 | 0x0000 | 0x42c9 | IEEE 8... | 105 | Data, Dst: 0x42c9, Src: 0x0000, Bad FCS |
| 35 | 17.437044 | 00:15:5f:00:b4:4d:2... | 0x0000 | IEEE 8... | 27 | Association Request, RFD, Bad FCS |
| 36 | 17.956659 | 00:15:5f:00:b4:4d:2... | 0x0000 | IEEE 8... | 86 | Data Request, Bad FCS |
| 37 | 18.380939 | 0x87c4 | 0x0000 | IEEE 8... | 90 | Data Request, Bad FCS |
| 38 | 18.420964 | 0x87c4 | 0x0000 | IEEE 8... | 73 | Data Request, Bad FCS |
| 39 | 18.684087 | 0x87c4 | 0x0000 | IEEE 8... | 67 | Data Dst: 0x0000 Src: 0x87c4 Bad FCS |

► Frame 27: 97 bytes on wire (776 bits), 97 bytes captured (776 bits)
▼ IEEE 802.15.4 Data, Dst: 0x0000, Src: 0x42c9, Bad FCS
► Frame Control Field: 0x8861, Frame Type: Data, Acknowledge Request, PAN ID Compression, Destination Addressing Mode: Short/16-bit, Frame Ver...
Sequence Number: 11
Destination PAN: 0x872b
Destination: 0x0000
Source: 0x42c9
FCS: 0x6c04 (Incorrect, expected FCS=0xc2aa)
► [Expert Info (Warning/Checksum): Bad FCS]
► Data (86 bytes)

DoS Zigbee

- Silva/Nunes attack exploits a flaw in how recipients process inbound packets with regard to the IEEE 802.15.4 frame counter (FC) value
- When a transmitting node sends a secure packet, it includes a sequential frame counter value in each frame with a range of 0 to 0xffffffff-1
 - FC value is not encrypted but it is included in the calculation of MIC (Message Integrity Check) for a packet

DoS Zigbee

- A receiving node remembers the last observed FC value for all of the nodes on the network
- To defeat replay attacks and avoid reprocessing packet retransmissions, receiving node only accepts packets with greater FC than last observed
- FC is also used to make the ‘nonce’ unique for each packet transmitted by a specific node

DoS Zigbee

- The unique nonce is important for AES-CTR to avoid initialization vector collision
 - Attacker can use plaintext/ciphertext data to get the key, if collision
- IEEE 802.15.4 specifies that when FC is equal to 0xffffffff, receiving node must stop processing all further data from the device
 - Add the transmitter to a device blacklist
 - Only way to recover, administrator updating the network key on all devices (firmware update)

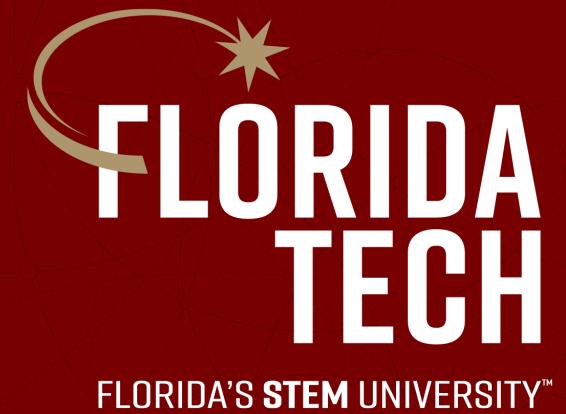
DoS Zigbee

- Under intended use circumstances in IEEE 802.15.4, devices are not likely to reach max FC value
 - 1 packet per second, a node will need 136 years to reach
- If node receives packet with increasing FC, it will accept and update FC value prior to validating the encrypted packet
 - Attacker forges 0xffffffff-1 and blacklist the legitimate transmitter

Zbscapy

```
$ sudo zbscapy
```

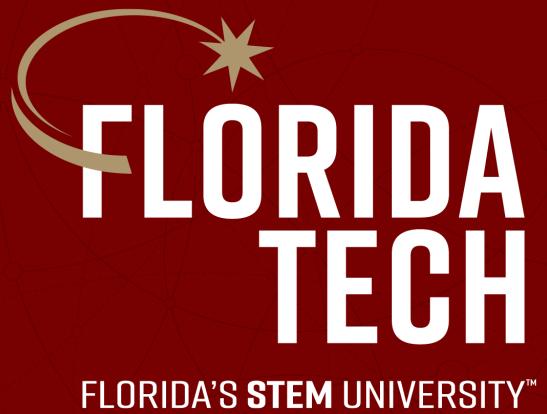
```
>>> kb = KillerBee()  
  
>>> conf.killerbee_channel = 15  
  
>>> f = Dot15d4()/Dot15d4Cmd(cmd_id=7)  
  
>>>kbsendp(f,iface=kb) # start killerbee framework  
# acquire killerbee-enabled device  
# set to channel #15  
# craft a IEEE 802.15.4 Command Frame  
# with the CMD_ID = 7 (Beacon Req)  
# send the frame
```



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

16. Z-Wave

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

Z-wave

Overview

Protocol Stack

Security

Recall: Zigbee: Network Discovery

- First assessment is to discover networks within range and enumerate the configuration of devices
 - Simple way; mimic Zigbee network discovery process with Killerbee
- Part of network discovery process in Zigbee Standard, ZDEs transmit beacon request on a given channel
 - All ZR and ZCs receiving beacon -> respond by sending a beacon frame
 - Disclose PAN ID, ZC or ZR source address, stack profile / version, extended IEEE address information
- Using same technique to actively scan for the presence of Zigbee network

Recall: Zigbee: Network Discovery

- Killerbee tool zbstumbler (similar to Wifi discovery tool Netstumbler):
 - Channel hops and transmits beacon request frames
 - Every two seconds hopping to a new channel
 - Display useful information from response beacon frames

```
test@test-HP-EliteBook-840-G3:~/killerbee$ sudo zbstumbler
[sudo] password for test:
Warning: You are using pyUSB 1.x, support is in beta.
zbstumbler: Transmitting and receiving on interface '1:11'
New Network: PANID 0xC762 Source 0xE2DA          Stack Profile: ZigBee Enterprise
  Ext PANID: 79:21:70:53:a3:d7:fc:34
  Stack Version: ZigBee 2006/2007
  Channel: 11
New Network: PANID 0xC762 Source 0xFF4F          Stack Profile: ZigBee Enterprise
  Ext PANID: 79:21:70:53:a3:d7:fc:34
  Stack Version: ZigBee 2006/2007
  Channel: 11
```

Recall: DoS Zigbee

- Silva/Nunes attack exploits a flaw in how recipients process inbound packets with regard to the IEEE 802.15.4 frame counter (FC) value
- When a transmitting node sends a secure packet, it includes a sequential frame counter value in each frame with a range of 0 to 0xffffffff-1
 - FC value is not encrypted but it is included in the calculation of MIC (Message Integrity Check) for a packet

Z-Wave

- Low-energy, mesh-networking protocol
- Predominately used in home automation (locks, garage door openers, thermostats)
 - Over 100 million products in use in homes
- Proprietary design by Sigma Systems and governed by standards established by Z-Wave Alliance
 - Does not share details of protocol outside of NDA (nondisclosure agreement)
 - Controls all fabrication and delivery of Z-Wave chips to product manufacturers

Z-Wave

- Aggressive power conservation for long battery life
 - Like Zigbee
- Using a mesh networking model to accommodate greater device range
- Relatively simple protocol
- Support for positive acknowledgement and frame retransmission, self-forming and dynamic routing topology updates, and application-specific profiles

Z-Wave Overview

- Z-Wave is based on a mesh network topology
 - Each (non-battery) device installed in the network becomes a signal repeater
 - As a result, the more devices you have in your home, the stronger the network becomes
- While Z-Wave signals easily travel through most walls, floors and ceilings, the devices can also intelligently route themselves around obstacles to attain seamless, robust, whole-home coverage

Z-Wave Overview

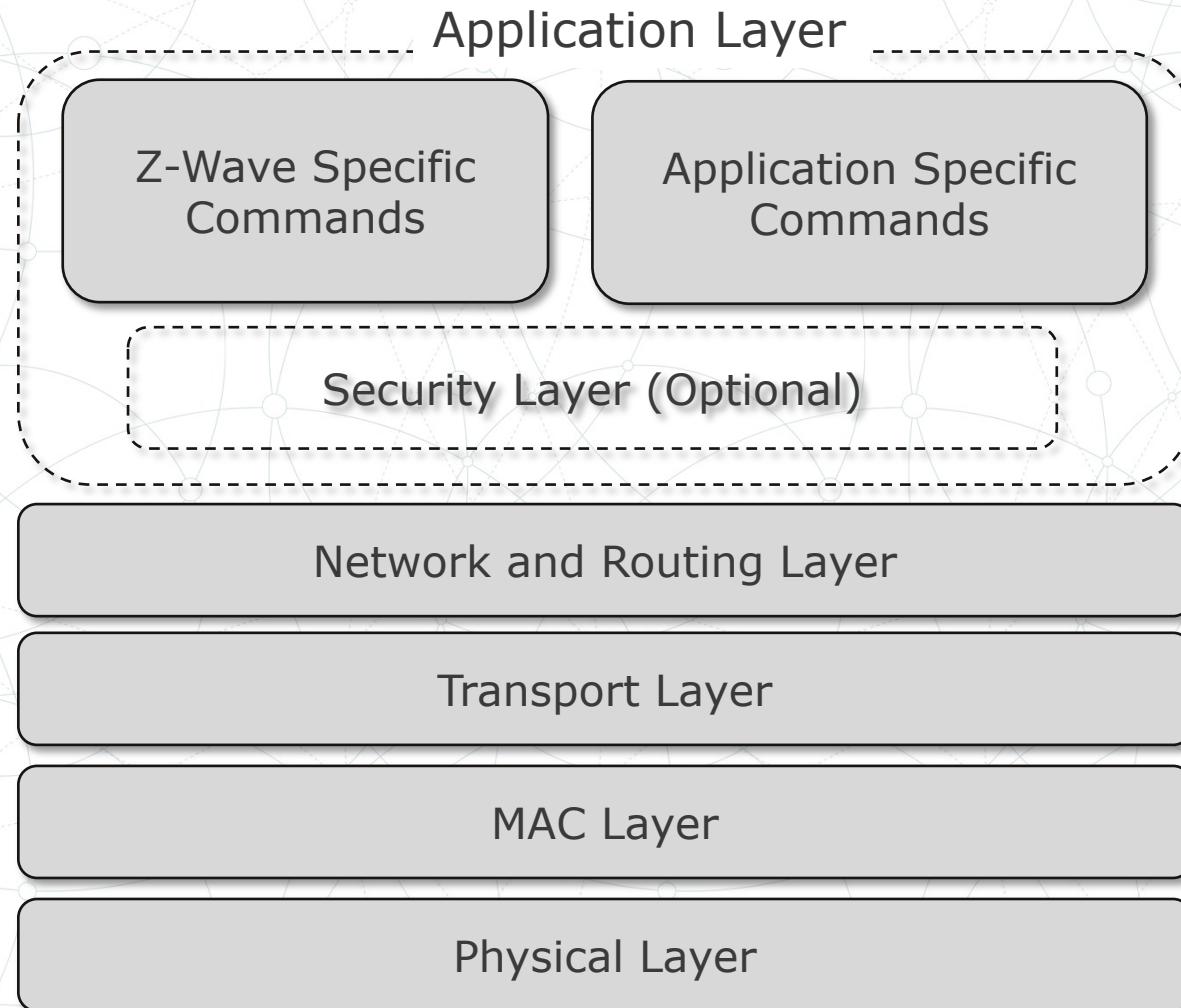
- While Z-Wave has a range of 100 meters (or 328 feet) in open air, building materials reduce that range (roughly every 30 feet)
 - Operates at 908.42 MHz in the United States and Canada
- The Z-Wave networks can be linked together for even larger deployments
- Each Z-Wave network can support up to 232 Z-Wave devices

Z-Wave Offers

- Extremely simple setup
 - Plug & Play
- Wireless Mesh Network
- Every non-battery node is a repeater
- Extremely robust
- Ultra-low power
 - Ideal for battery powered sensors
- Sub – 1GHz Frequency
- Multi speed: 40...100 kbps
- Low communication latency
- Interoperable
- Largest Home Control Community

Z-Wave Protocol Stack

- Uses structured protocol stack



Z-Wave PHY Layer

- Uses sub-1-GHz band to accommodate frequency variations in different countries
- Offers 3 different RF Profiles (R1, R2, and R3) with unique data rates, encodings, modulation, and packet frame sizes
 - R1 is deprecated by Z-Wave alliance but might be in use for some
 - Range from 3 – 75 feet; depending on transmit power

| Profile | Data Rate | Encoding | Modulation | Packet Size |
|---------|-----------|------------|------------|-------------|
| R1 | 9.6 Kb/s | Manchester | FSK | 64 bytes |
| R2 | 40 Kb/s | NRZ | FSK | 64 bytes |
| R3 | 100 Kb/s | NRZ | FSK | 170 bytes |

Z-Wave MAC Layer

- Responsible for several attributes;
 - Packet framing and formatting
 - Positive ACK
 - Error detection
 - Retransmission of packets
 - Unicast, broadcast, and multicast processing
 - Address selection and allocation functions

Z-Wave MAC Layer

- Basic architecture of Z-Wave network: Controller device and slave devices
- Single primary controller device is responsible for establishing the network and selecting unique network identifier (i.e., HomeID)
- Controller devices are able to initiate a transmission on the network (polling or updating target devices) and responsible for maintaining network routing information
- Slave devices follow the instructions of controllers without dealing with how

Z-Wave MAC Layer

- Z-Wave controller;
 - Portable controller device; typically battery powered and is capable of relearning network topology as it moves about the home
 - Static controller device; powered through a consistent source and may also be connected to other networks, providing gateway services between the Z-Wave and IP network

Z-Wave MAC Layer Frame Format

- Varies depending on the RF profile
- Each Z-Wave network is uniquely identified by randomly selected value when the network is established, HomeID, that is transmitted as the first four bytes of each packet
 - Similar to IEEE 802.11 BSSID or Zigbee PAN ID
 - Used to differentiate Z-Wave networks in close physical proximity and associate all the nodes participating in the same network

Z-Wave MAC Layer Frame Format

- HomeID: Randomly selected 4-byte value chosen by controller
- NodeID: 1-byte value assigned to the node by the controller
 - Max 232 nodes (22 left for reserved, 1 for broadcast, 1 uninitialized)
- Frame Control: 16-bit field with several subfields

Z-Wave R1/R2 Frame Format

| | | | | | | |
|--------|----------------|---------------|--------|---------------------|---------|-----|
| HomeID | Source Node ID | Frame Control | Length | Destination Node ID | Payload | FCS |
|--------|----------------|---------------|--------|---------------------|---------|-----|

Z-Wave R3 Frame Format

| | | | | | | | |
|--------|----------------|---------------|--------|-----------------|---------------------|---------|-----|
| HomeID | Source Node ID | Frame Control | Length | Sequence number | Destination Node ID | Payload | FCS |
|--------|----------------|---------------|--------|-----------------|---------------------|---------|-----|

Z-Wave MAC Layer Frame Control Format

- 16-bit frame control field represents several subfields with two reserved bits:
 - Routed; to indicate if packet has been routed by another node prior to delivery
 - Ack Request; to indicate that the receiving node should ACK the packet
 - Low Power; to indicate that the packet was transmitted using low-power output for reduced range
 - Speed modified (R1 / R2 only); when a packet is transmitted at a lower data rate than what is supported by src / dst

Z-Wave MAC Layer Frame Control Format

- Header type; packet type (unicast/multicast/ACK/broadcast)
- Beam control; node shall be woken from power conservation state with continuous transmission
- Seq number (R1/R2 only); identify packet for subsequent ACK

Z-Wave R1/R2 Frame Control Format

| | | | | | | | | |
|--------|---------|-----------|------------|-------------|------|--------------|------|-----------------|
| Routed | Ack Req | Low Power | Speed Mod. | Header Type | Res. | Beam Control | Res. | Sequence Number |
|--------|---------|-----------|------------|-------------|------|--------------|------|-----------------|

Z-Wave R3 Frame Control Format

| | | | | | | | |
|---------|-----------|------|-------------|------|--------------|------|----------|
| Ack Req | Low Power | Res. | Header Type | Res. | Beam Control | Res. | Reserved |
|---------|-----------|------|-------------|------|--------------|------|----------|

Z-Wave MAC Layer Fields

- Length field; indicates the length of the entire packet including the header, payload, and Frame Check Sequence (FCS)
- Destination NodeID is present in unicast and multicast frames to indicate the intended recipient
- Data payload; 0-54 bytes in R1/R2 nonmulticast, 0-25 bytes in multicast
 - R3; 0-158 bytes and 0-129 bytes
- FCS provides simple integrity check using XOR checksum for R1/R2 and CRC-16 for R3

Z-Wave R3 Frame Format

| | | | | | | | |
|--------|----------------|---------------|--------|-----------------|---------------------|---------|-----|
| HomeID | Source Node ID | Frame Control | Length | Sequence number | Destination Node ID | Payload | FCS |
|--------|----------------|---------------|--------|-----------------|---------------------|---------|-----|



Z-Wave Network Layer

- Defines device responsibilities and responsible for other network components such as HomeID selection and NodeID allocation process as well as network route establishment
- Z-Wave inclusion and exclusion for network connections

Z-Wave Network Layer: Inclusion

- Involves configuring the controller in inclusion mode (allowing it to accept new nodes) by pressing a physical button or choosing a menu item, and pressing a button on the new node to initiate an inclusion exchange
- When the new node initiates the inclusion process, it sends a Z-Wave node information frame using homeID of 0x00000000 and nodeID of 0x00 and a broadcast dest NodeID
 - Discloses the capabilities of the new device to the controller, which, in turn, allocates a NodeID to the new device for subsequent use on the network and updates routing tables to accommodate packet delivery to the new node

Z-Wave Network Layer: Exclusion

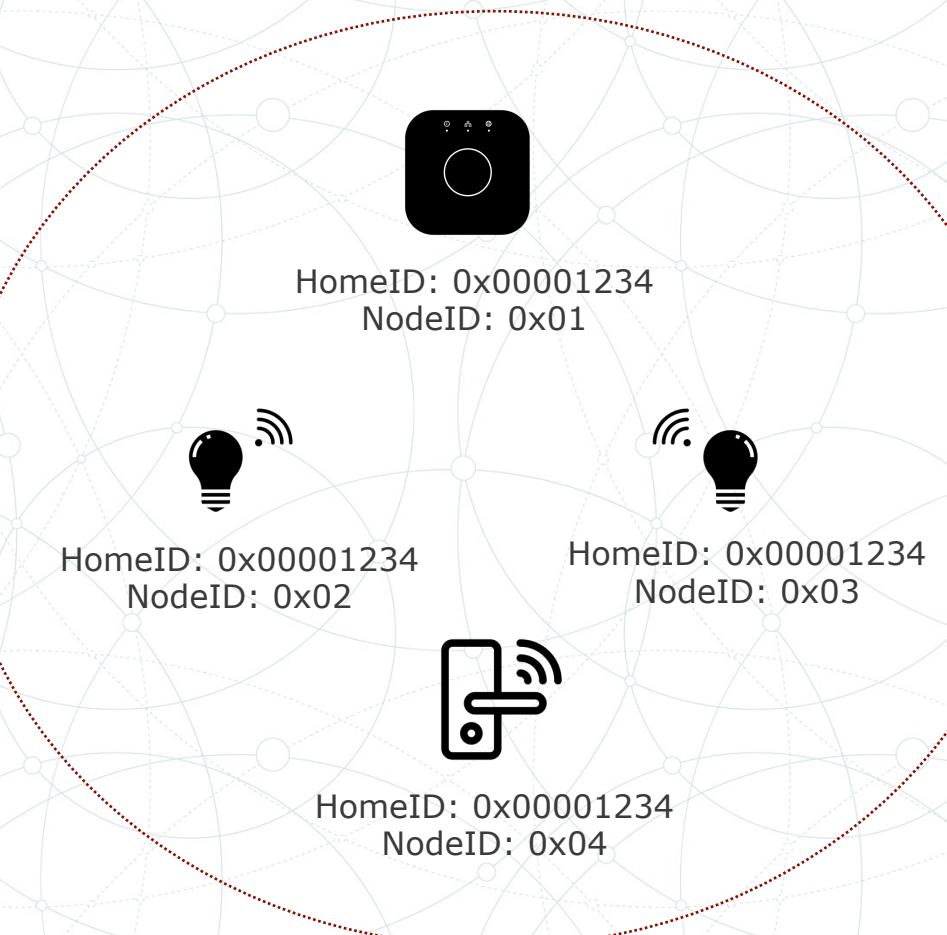
- Similar to inclusion but functionally opposite
- A node joined the Z-Wave network via inclusion cannot leave the network to join a different Z-Wave controller without completing the exclusion process
- Involves pressing a physical button on the controller and the device node, causing device to return to unallocated nodeID 0x00

Z-Wave Inclusion & Exclusion

- Strong component of the overall Z-Wave security
 - Requires physical access to the controller
- Is it secure enough?
 - What about spoofing?

Z-Wave Network Topology

- Nodes in a Z-Wave have a 1-byte NodeID, which must be different than every other node in the network
- Nodes in a Z-Wave network have a 4-byte HomeID, assigned by the controller at the time of inclusion
- Nearby networks must have different HomeIDs



Z-Wave Application Layer

- Responsible for parsing and processing the data requests and responses in the packet payload
- Handles both application-specific and Z-Wave application control data
- Basic application payload format:



Z-Wave Application Layer

- Z-Wave uses application command classes to differentiate actions and responses on the network
- Each command class supports one or more commands within the class that define the basic functionality of the application layer
 - For ex., CLASS_SWITCH_ALL command class is used to control multiple network devices for power on/off control so that the user can shut them all off with one single button

Z-Wave Application Layer

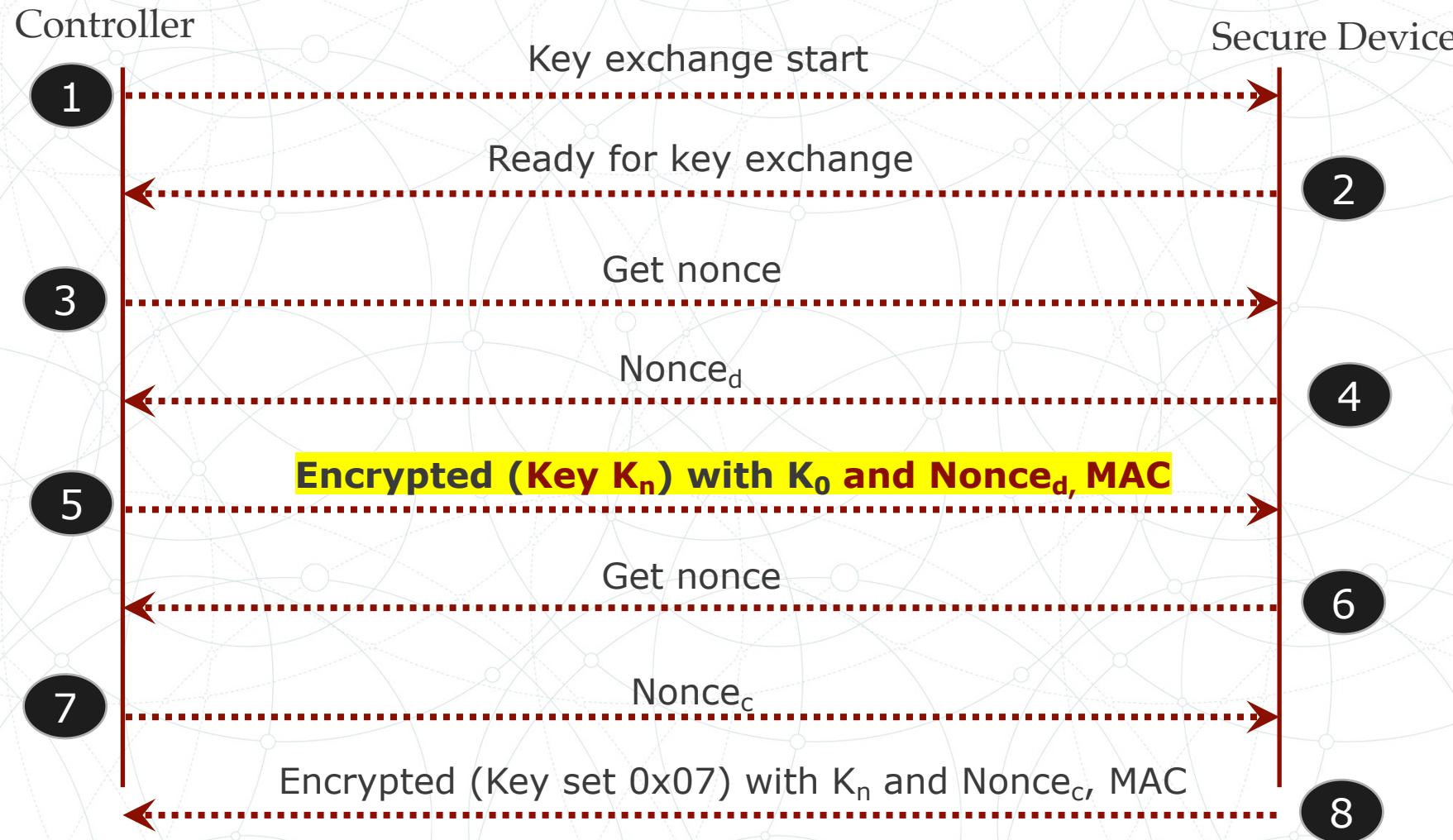
- Each Z-Wave application has well-defined set of functionality based on the device type and manufacturer application command class support
- At the application layer, Z-Wave uses command classes specified in ITU-T G.9959
 - Open source project such as OpenZWave are instrumental to understand and document the proprietary Z-Wave

<https://github.com/openzwave/>

Z-Wave Security

- Uses AES-OFB (Output Feedback Mode) to provide data confidentiality on the network
 - Conserve the amount of payload content transmitted in Z-Wave frames while being NIST (National Institute of Standards and Technology) approved
- AES CBC-MAC (cipher block chaining message authentication code) for data integrity protection
- CLASS_SECURITY command; key exchange process to derive keys

Z-Wave Key Exchange Process

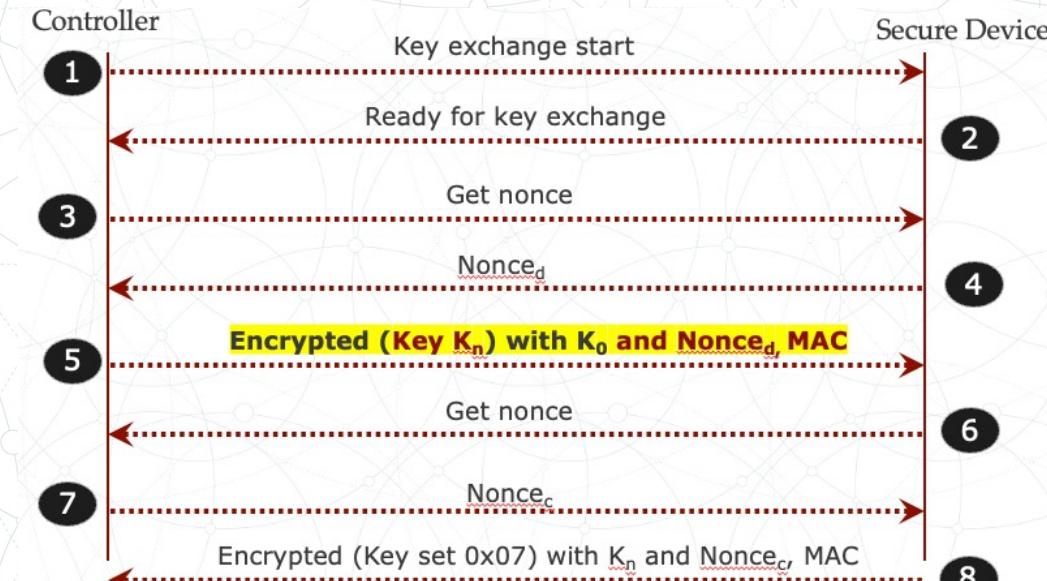


Temporal Key
 $K_0 = 16$ bytes of 0x00

Network Key
 $K_n = \text{Chosen by controller}$

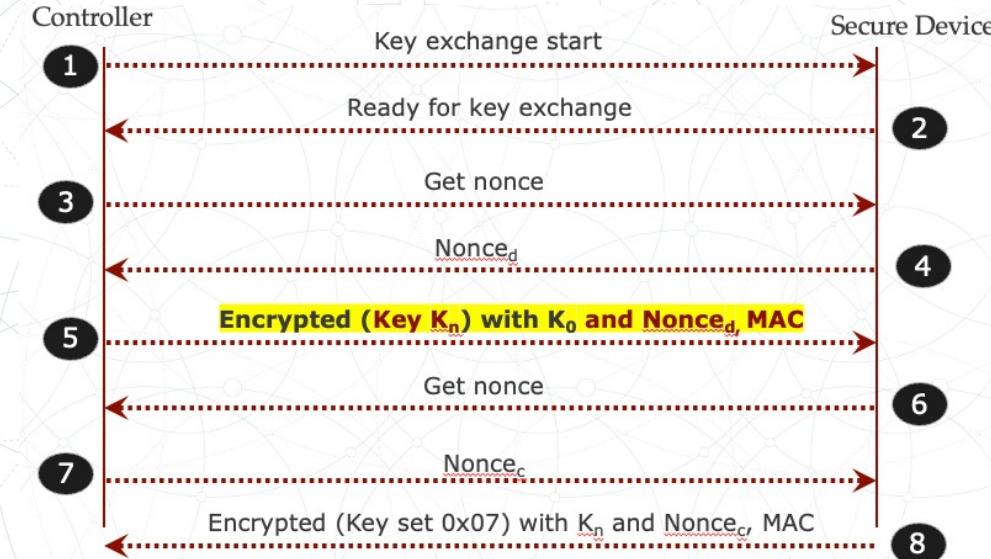
Z-Wave Key Exchange Process

- Step 1,2; controller and secure device prepare for the key exchange
- Step 3; controller requests nonce
- Step 4; with the nonce value, controller encrypts the network key K_n , using temporary key K_0
 - Network key is randomly selected by controller when the network is established and is unique for each Z-Wave network
 - Temporary key is an array of 16 bytes 0x00



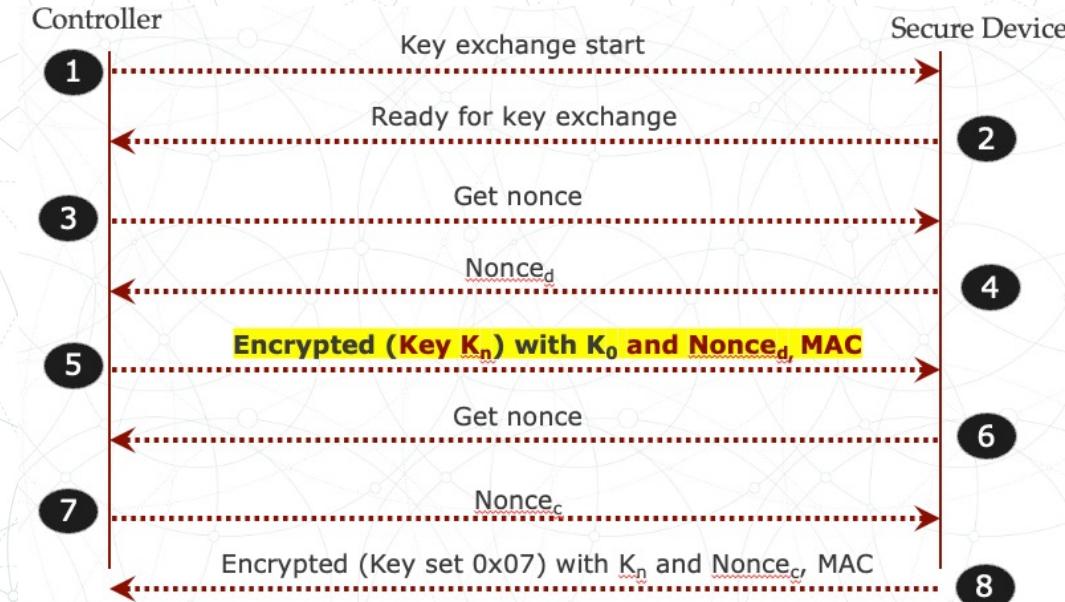
Z-Wave Key Exchange Process

- When the secure device receives encrypted network key K_n , and MAC from the controller, it validates the MAC and decrypts the message with K_0
 - Secure device then registers the decrypted K_n as the current key
 - Next, step 6, secure device requests nonce from the controller



Z-Wave Key Exchange Process

- With the nonce value, secure device encrypts a "key set OK" message (hex 0x07) with K_n
- The controller receives the "key set OK" message validates that the packet was encrypted using K_n by validating MAC



Z-Wave Key Exchange Vulnerabilities

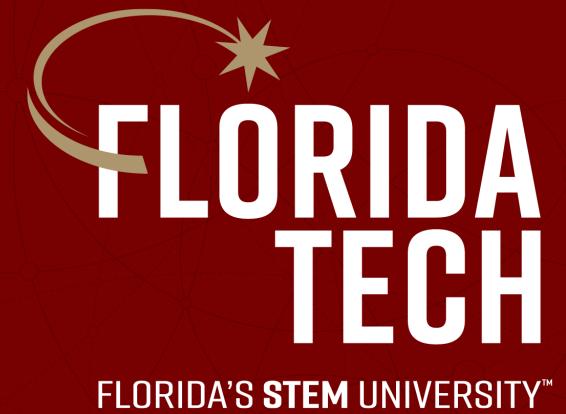
- MitM:
 - The secure device does not validate the identity of controller other than validating the MAC of encrypted Kn message using the temporary key K_0
 - Attacker can use any Z-Wave controller that support CLASS_SECURITY command class to intercept the inclusion process with a target device
 - Causing victim to associate to a malicious network

Z-Wave Key Exchange Vulnerabilities

- Key recovery attack:
 - There is no confidentiality protection in the delivery of the K_n key over the network since the K_0 is well known
 - Attacker passively observing the inclusion process can recover the network key K_n
 - Use it later to decrypt or forge arbitrary packets on the network

Z-Wave Key Exchange's Solution

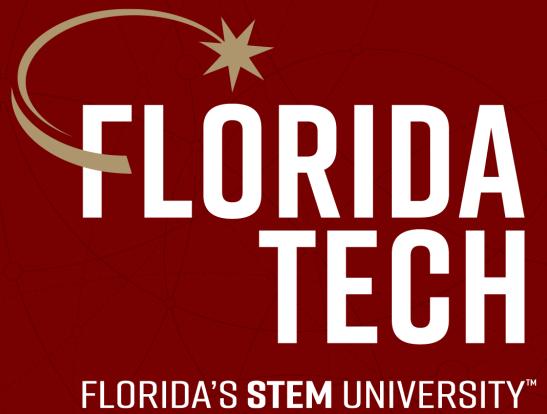
- Low power inclusion mode
 - Controller and secure device transmit using minimal power capabilities
 - Require no more than 3 feet apart to complete the process
 - Also infrequent practice of adding new devices
 - Results in less opportunity for the attacker



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

17. Z-Wave Security

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

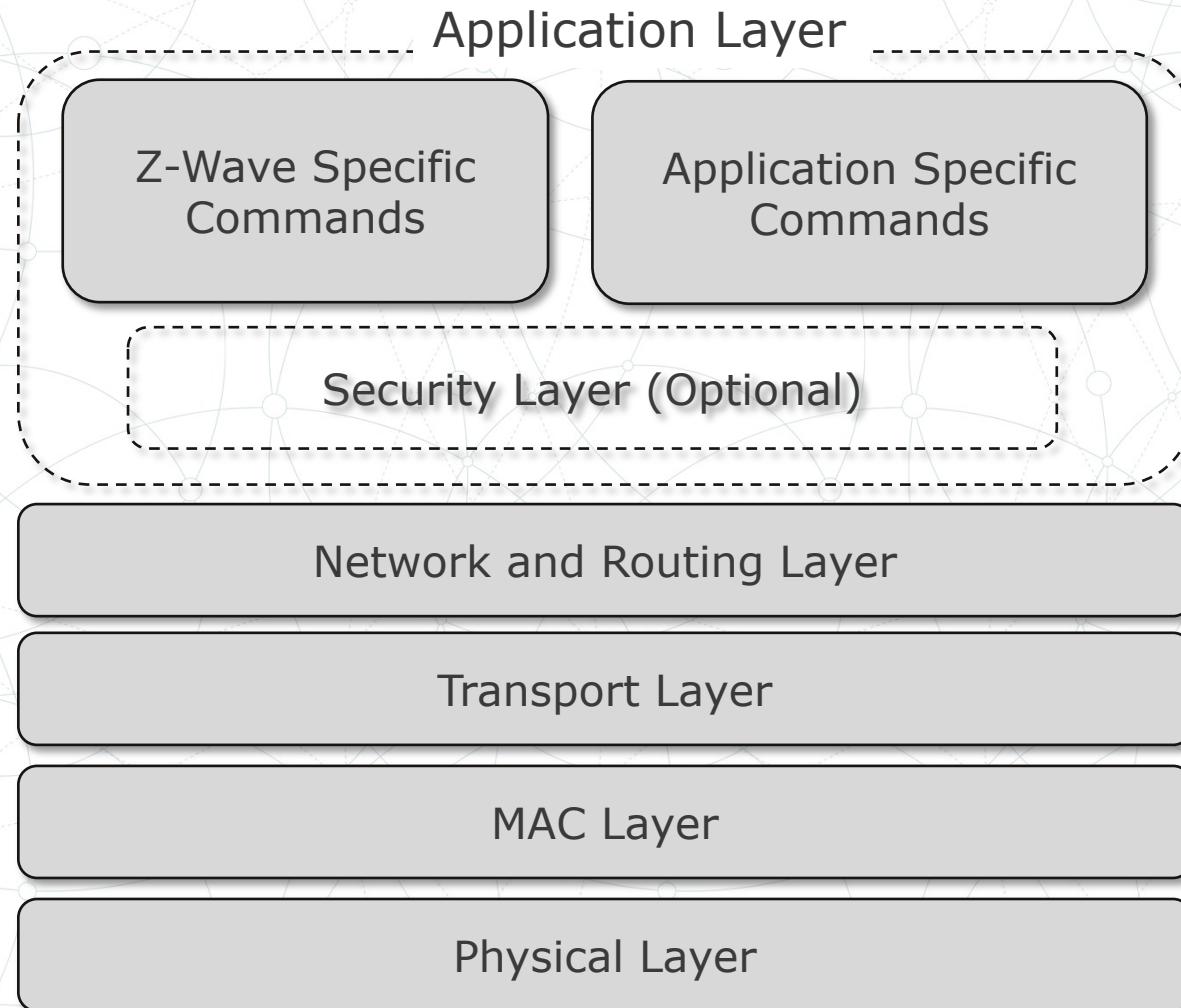
Z-wave Security

Recall: Z-Wave

- Low-energy, mesh-networking protocol
- Predominately used in home automation (locks, garage door openers, thermostats)
 - Over 100 million products in use in homes
- Proprietary design by Sigma Systems and governed by standards established by Z-Wave Alliance
 - Does not share details of protocol outside of NDA (nondisclosure agreement)
 - Controls all fabrication and delivery of Z-Wave chips to product manufacturers

Recall: Z-Wave Protocol Stack

- Uses structured protocol stack

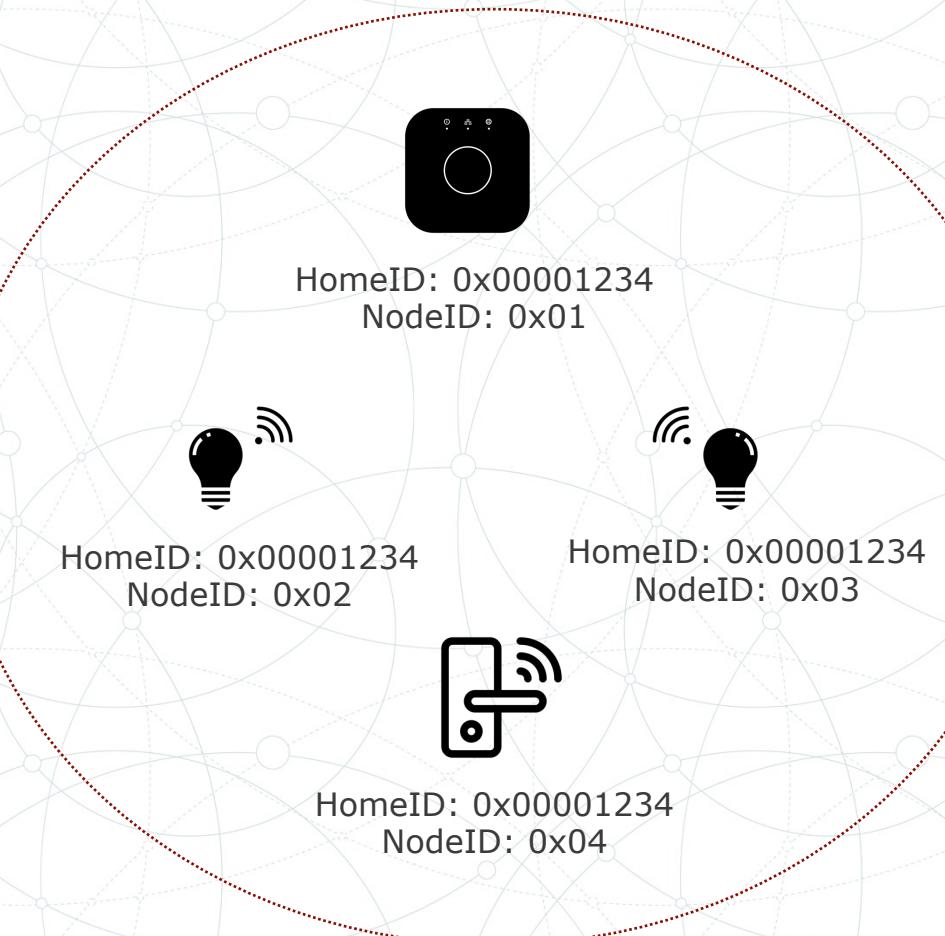


Recall: Z-Wave Network Layer: Inclusion

- Involves configuring the controller in inclusion mode (allowing it to accept new nodes) by pressing a physical button or choosing a menu item, and pressing a button on the new node to initiate an inclusion exchange
- When the new node initiates the inclusion process, it sends a Z-Wave node information frame using homeID of 0x00000000 and nodeID of 0x00 and a broadcast dest NodeID
 - Discloses the capabilities of the new device to the controller, which, in turn, allocates a NodeID to the new device for subsequent use on the network and updates routing tables to accommodate packet delivery to the new node

Z-Wave Network Topology

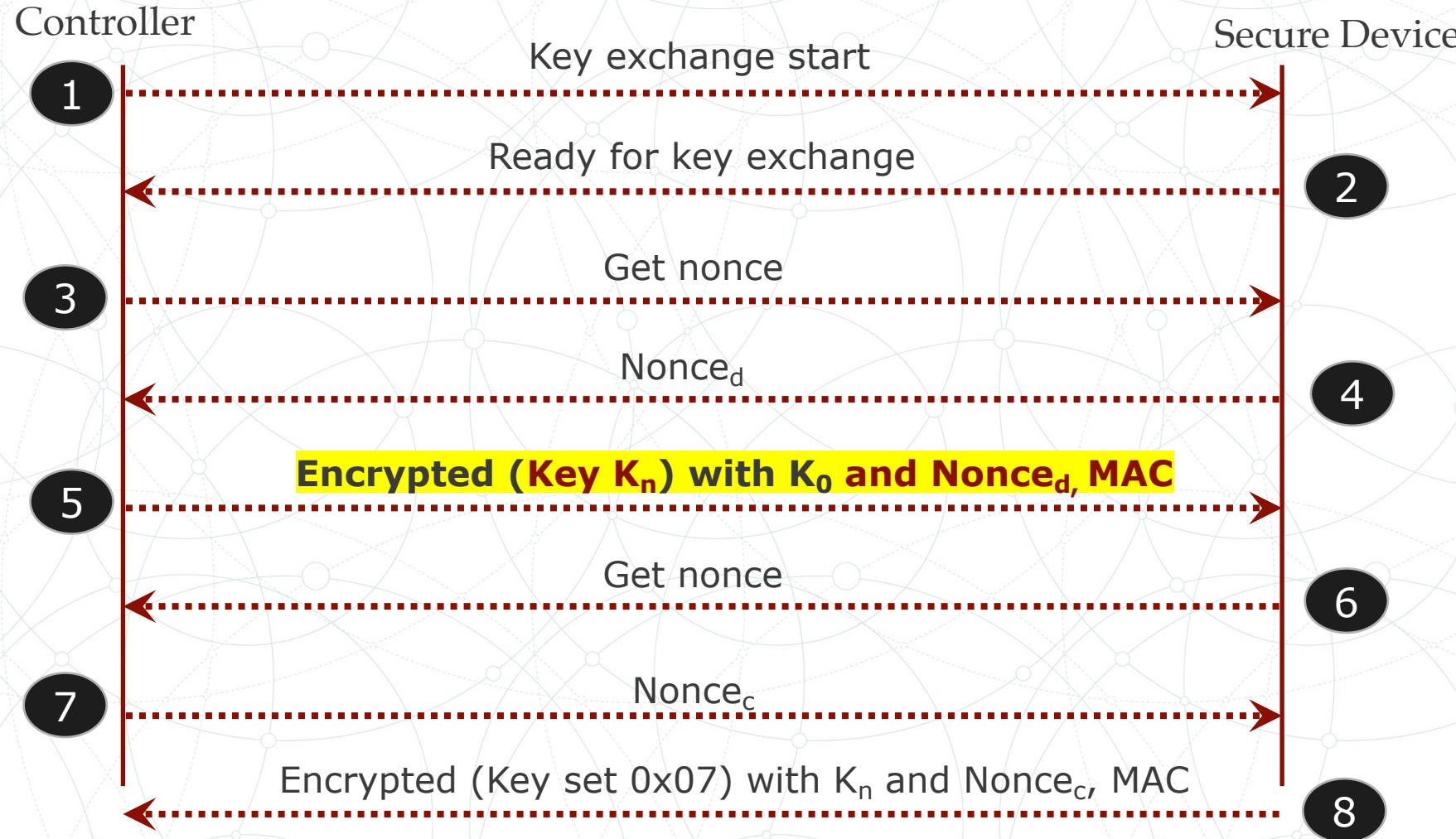
- Nodes in a Z-Wave have a 1-byte NodeID, which must be different than every other node in the network
- Nodes in a Z-Wave network have a 4-byte HomeID, assigned by the controller at the time of inclusion
- Nearby networks must have different HomeIDs



Recall: Z-Wave Security

- Uses AES-OFB (Output Feedback Mode) to provide data confidentiality on the network
 - Conserve the amount of payload content transmitted in Z-Wave frames while being NIST (National Institute of Standards and Technology) approved
- AES CBC-MAC (cipher block chaining message authentication code) for data integrity protection
- CLASS_SECURITY command; key exchange process to derive keys

Recall: Z-Wave Key Exchange Process



Recall: Z-Wave Key Exchange's Solution

- Low power inclusion mode
 - Controller and secure device transmit using minimal power capabilities
 - Require no more than 3 feet apart to complete the process
 - Also infrequent practice of adding new devices
 - Results in less opportunity for the attacker

Z-Wave Key Derivation

- After the K_n (network key) is established, the device generates two additional keys K_c (packet encryption) and K_m (message auth key)
 - K_c (packet encryption) = AES-ECB_{kn}(Password_c)
 - K_m (message auth key) = AES-ECB_{kn}(Password_m)
- Where Password_c and Password_m are static values across all Z-Wave devices
- However, if we observe Kn and we know Password_c and Password_m, we can compute the packet encryption and message auth keys

Z-Wave Key Establishment Attack

- We can force the key establishment process to establish a new K_n if we missed the original key establishment
 - Similar to an IEEE 802.11 Deauth

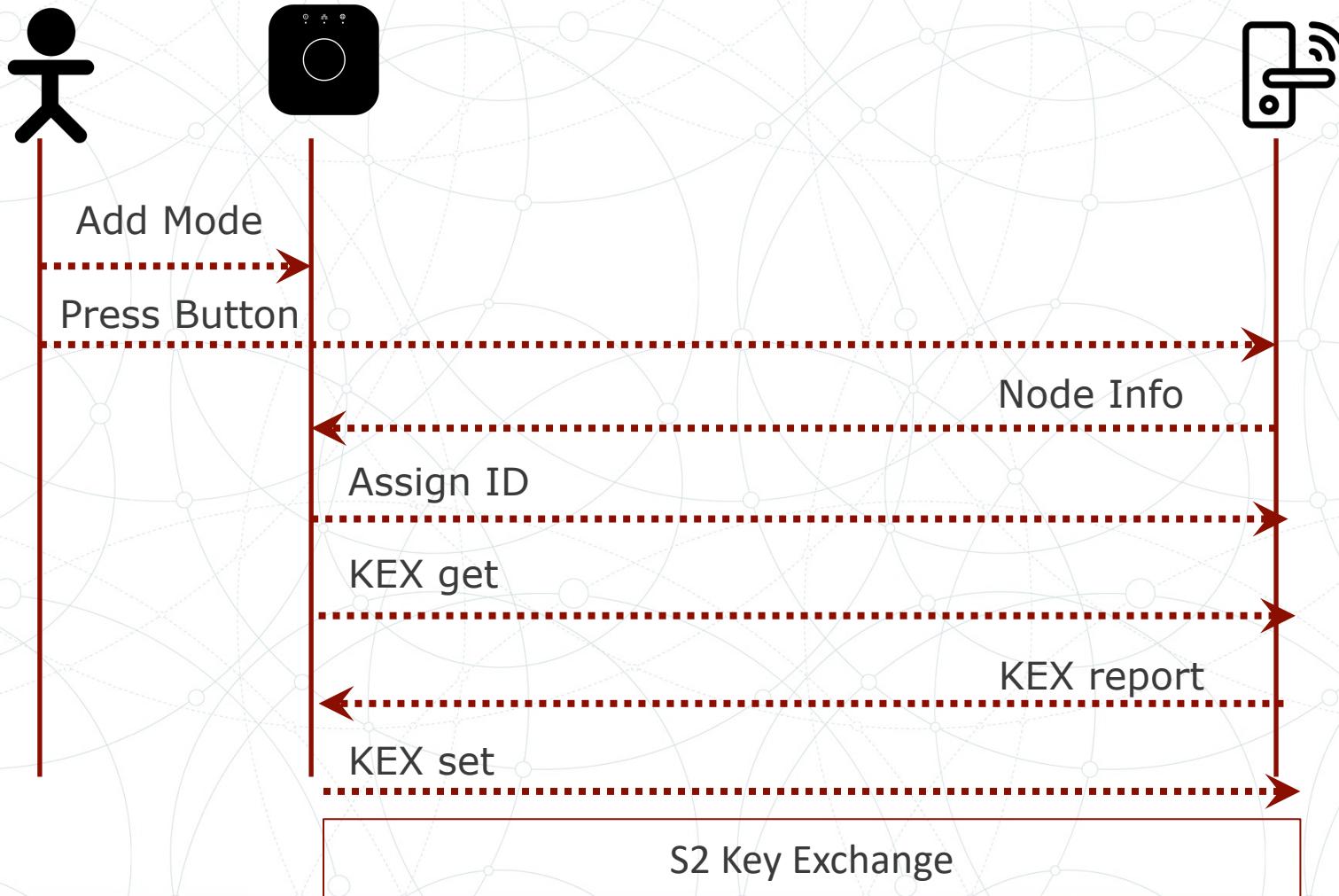


Z-Wave Key Establishment: Solution

- In response to the flawed S_0 Key Establishment and Pairing Process, SI Labs (i.e., Z-Wave Alliance) responded with a S_2 Key Establishment
- Removed the null temporal key
- Each devices has a DSK [device specific key] – 16 byte key
- Key exchanged using Diffie Hellman key exchange protocol
- Removed issues with man-in-the-middle
 - Are we safe now?

https://community.silabs.com/s/topic/0TO1M000000qHcQWAU/zwave?language=en_US&tabset-178da=2

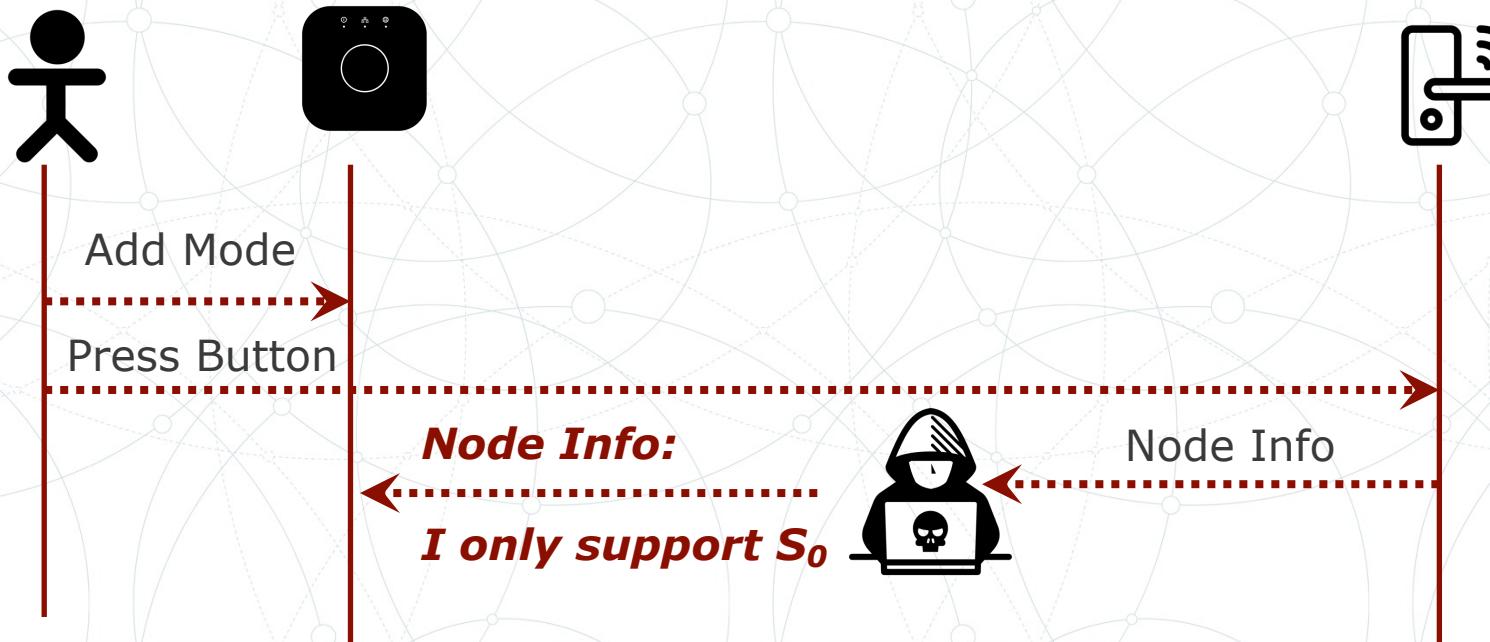
Z-Wave Key Exchange



S2 Z-Wave Rollback Attack

- An attacker jams the node info or responds prior to the node
- Provides node info that device only supports flawed S_0 pairing

mode



Mitigating Eavesdropping in Z-Wave

- As in any wireless technology, attacker can always capture the traffic (aka eavesdropping)
 - Useless if confidentiality and integrity of the data is ensured
- Unfortunately, the use of encryption in Z-Wave is optional
 - Switch to a vendor that offers network confidentiality and integrity control

Injection Attacks in Z-Wave

- If no encryption, attacker can capture and replay the packets
- Injection is also done if the device is in the Z-Wave network
 - To be in the network, Z-Wave inclusion needed
 - Does it solve this problem?
 - Attacker can spoof the address and inject any packet

<https://github.com/joswr1ght/killerzee>

Scapy-Radio Framework

- Modified version of Scapy to support
 - Zwave
 - Zigbee
 - 802.15.4
- Works with software defined radios (SDR)

| | | | | |
|-----------------------|---|-----|------------------------|--------------|
| Balint Seeber | Removed deprecated function from Wireshark dissector. | ... | f1240ab on Apr 1, 2016 | ⌚ 12 commits |
| gnuradio | Removed deprecated function from Wireshark dissector. | | | 7 years ago |
| scapy | disables xbee for the moment | | | 8 years ago |
| utils/Zwave | Add copyright stuff | | | 8 years ago |
| wireshark/scapy-radio | Removed deprecated function from Wireshark dissector. | | | 7 years ago |
| .hgignore | initial import of scapy-radio | | | 8 years ago |
| README.md | Add note about GNU Radio 3.7.5 in README | | | 8 years ago |
| install.sh | Bugfix on installation script | | | 8 years ago |

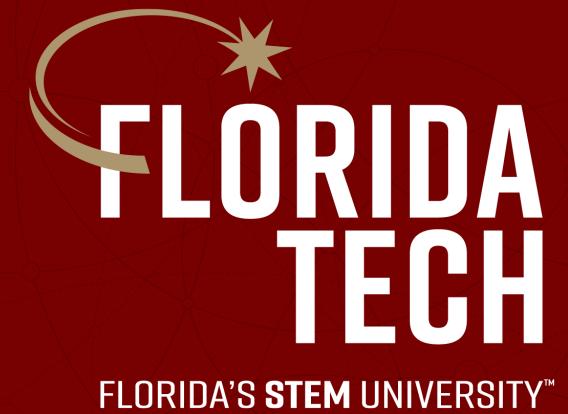
☰ README.md

Introduction

This tool is a modified version of scapy that aims at providing an quick and efficient pentest tool with RF capabilities.

It includes:

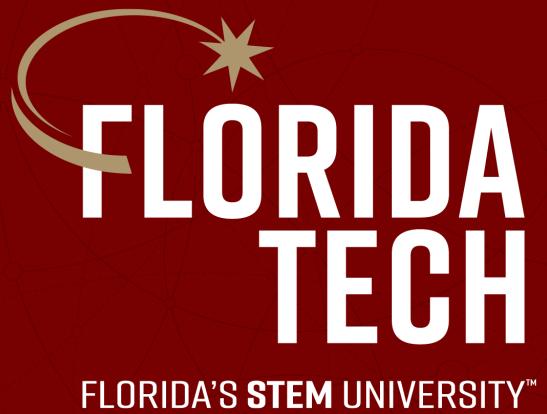
- A modified version of scapy that can leverage GNU Radio to handle a SDR card
- GNU Radio flow graphs (GRC files) we have build that allows full duplex communication
- GNU Radio blocks we have written to handle several protocols



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

18. LoRaWAN

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

LoRaWAN

As a Wireless Standard

History

Characteristics

Network Architecture

LoRaWAN

- Long Range (LoRa) Wide Area Network (WAN)
 - “A Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated ‘things’ to the internet in regional, national or global networks,
 - And support targets key Internet of Things (IoT) requirements such as bi-directional communication, end-to-end security, mobility and localization services”

LoRaWAN Protocol

- The LoRaWAN specification is open so anyone can set up and operate a LoRa network
 - Wireless audio frequency technology that operates in a license-free radio frequency spectrum
- It uses a narrow band waveform with a central frequency to send data
 - Makes it robust to interference

LoRaWAN Protocol

- LoRa is a physical layer protocol that uses spread spectrum modulation and supports long-range communication at the cost of a narrow bandwidth
 - Modulation technique derived from Chirp Spread Spectrum (CSS) technology
 - It encodes information on radio waves using chirp pulses;
 - Similar to the way dolphins and bats communicate!

LoRaWAN as Wireless Standard

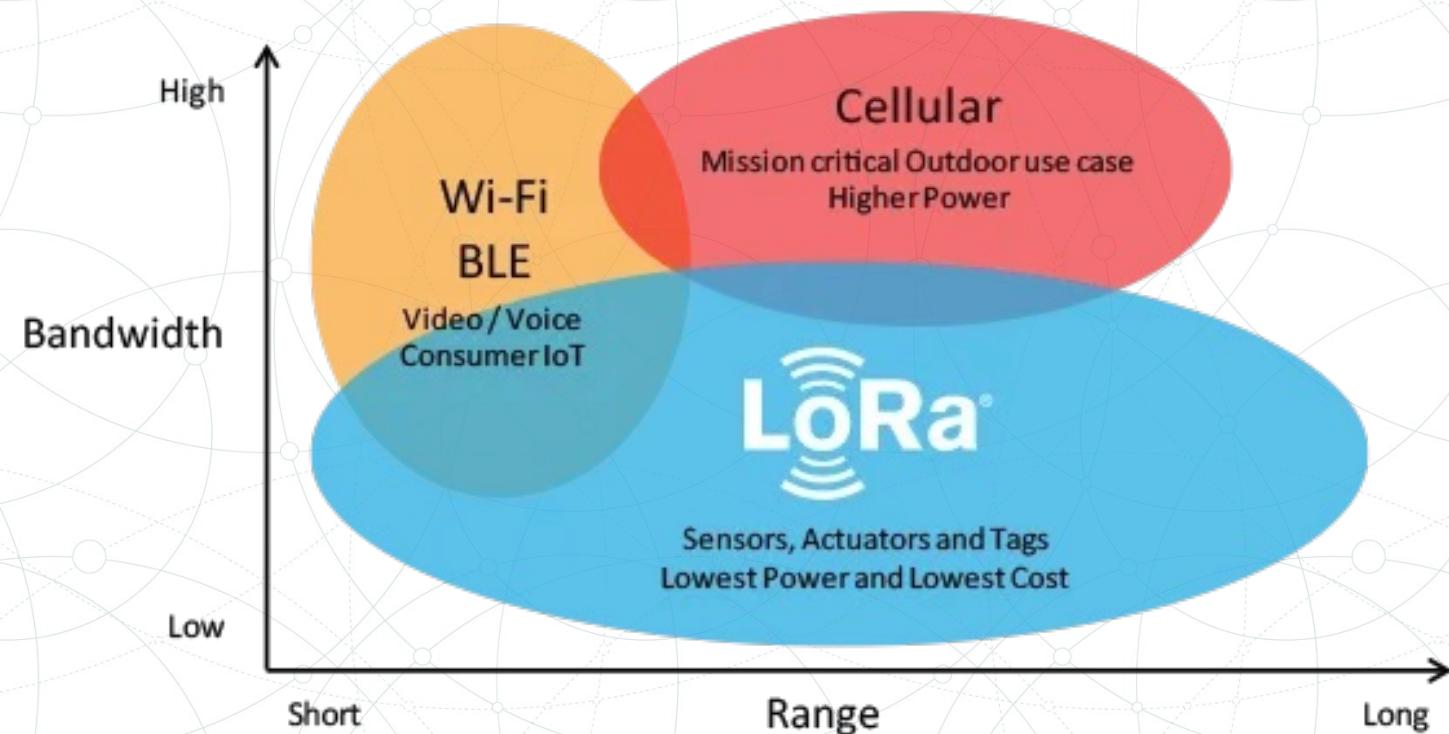
- LoRa is ideal for applications that transmit small chunks of data with low bit rates
- Data can be transmitted at a longer range compared to technologies like WiFi, Bluetooth or ZigBee
- These features make LoRa well suited for sensors and actuators that operate in low power mode

LoRaWAN as Wireless Standard

- LoRa can be operated on the license free sub-gigahertz bands, for example, 915 MHz, 868 MHz, and 433 MHz
- It also can be operated on 2.4 GHz to achieve higher data rates compared to sub-gigahertz bands, at the cost of range
- These frequencies fall into ISM bands that are reserved internationally for industrial, scientific, and medical purposes

LoRaWAN as Wireless Standard

- Suitable for transmitting small size payloads (like sensor data) over long distances
 - Greater communication range with low bandwidths than other competing wireless data transmission technologies



Why LoRaWAN?

- Ultra low power;
 - LoRaWAN end devices are optimized to operate in low power mode and can last up to 10 years on a single coin cell battery
- Long range;
 - LoRaWAN gateways can transmit and receive signals over a distance of over 10 kilometers in rural areas and up to 3 kilometers in dense urban areas
- Deep indoor penetration;
 - LoRaWAN networks can provide deep indoor coverage, and easily cover multi floor buildings

Why LoRaWAN?

- License free spectrum;
 - You don't have to pay expensive frequency spectrum license fees to deploy a LoRaWAN network
- Geolocation;
 - A LoRaWAN network can determine the location of end devices using triangulation without the need for GPS
 - A LoRa end device can be located if at least three gateways pick up its signal
- High capacity;
 - LoRaWAN Network Servers handle millions of messages from thousands of gateways

Why LoRaWAN?

- Public and private deployments;
 - It is easy to deploy public and private LoRaWAN networks using the same hardware (gateways, end devices, antennas) and software (UDP packet forwarders, Basic Station software, LoRaWAN stacks for end devices)
- End-to-end security;
 - LoRaWAN ensures secure communication between the end device and the application server using AES-128 encryption

Why LoRaWAN?

- Firmware updates over the air;
 - You can remotely update firmware (applications and the LoRaWAN stack) for a single end device or group of end devices
- Roaming;
 - LoRaWAN end devices can perform seamless handovers from one network to another
- Low cost;
 - Minimal infrastructure, low-cost end nodes and open source software

Why LoRaWAN?

- Certification program;
 - The LoRa Alliance certification program certifies end devices and provides end-users with confidence that the devices are reliable and compliant with the LoRaWAN specification
- Ecosystem;
 - LoRaWAN has a very large ecosystem of device makers, gateway makers, antenna makers, network service providers, and application developers

LoRaWAN Use Cases

- Vaccine cold chain monitoring;
 - LoRaWAN sensors are used to ensure vaccines are kept at appropriate temperatures in transit
- Animal conservation;
 - Tracking sensors manage endangered species such as Black Rhinos and Amur Leopards
- Dementia patients;
 - Wristband sensors provide fall detection and medication tracking

LoRaWAN History

- The LoRaWAN protocol is developed and maintained by the LoRa Alliance
- The first LoRaWAN specification was released in January

2015

| Version | Release date |
|---------|---------------|
| 1.0 | January 2015 |
| 1.0.1 | February 2016 |
| 1.0.2 | July 2016 |
| 1.1 | October 2017 |
| 1.0.3 | July 2018 |
| 1.0.4 | October 2020 |

Characteristics of LoRaWAN

- LoRaWAN is a Media Access Control (MAC) layer protocol built on top of LoRa modulation
 - A software layer which defines how devices use the LoRa hardware, for example when they transmit, and the format of messages
- Long range communication up to 10 miles in line of sight
- Long battery duration of up to 10 years
 - For enhanced battery life, you can operate your devices in class A or class B mode, which requires increased downlink latency

Characteristics of LoRaWAN

- Low cost for devices and maintenance
- License-free radio spectrum but region-specific regulations apply
- Has a limited payload size of 51 bytes to 241 bytes depending on the data rate
 - The data rate can be 0,3 Kbit/s – 27 Kbit/s

LoRaWAN Network Overview

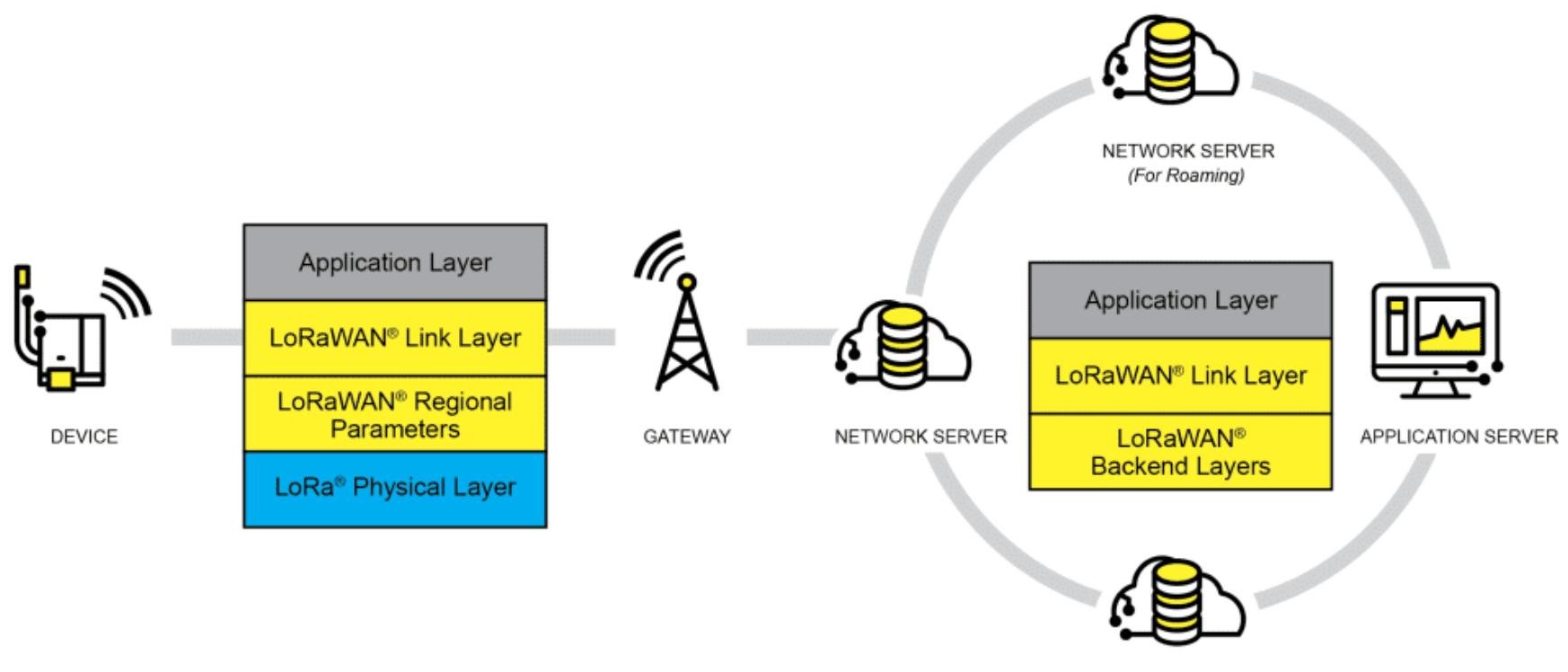
- LoRaWAN® network architecture is deployed in a star-of-stars topology in which gateways relay messages between end-devices and a central network server
- The gateways are connected to the network server via standard IP connections and act as a transparent bridge, simply converting RF packets to IP packets and vice versa

LoRaWAN Network Overview

- The wireless communication takes advantage of the Long Range characteristics of the LoRa physical layer, allowing a single-hop link between the end-device and one or many gateways
- All modes are capable of bi-directional communication,
 - And there is support for multicast addressing groups to make efficient use of spectrum during tasks such as Firmware Over-The-Air (FOTA) upgrades or other mass distribution messages

LoRaWAN Network Overview

- The specification defines the device-to-infrastructure (LoRa®) physical layer parameters & (LoRaWAN®) protocol and so provides seamless interoperability between manufacturers



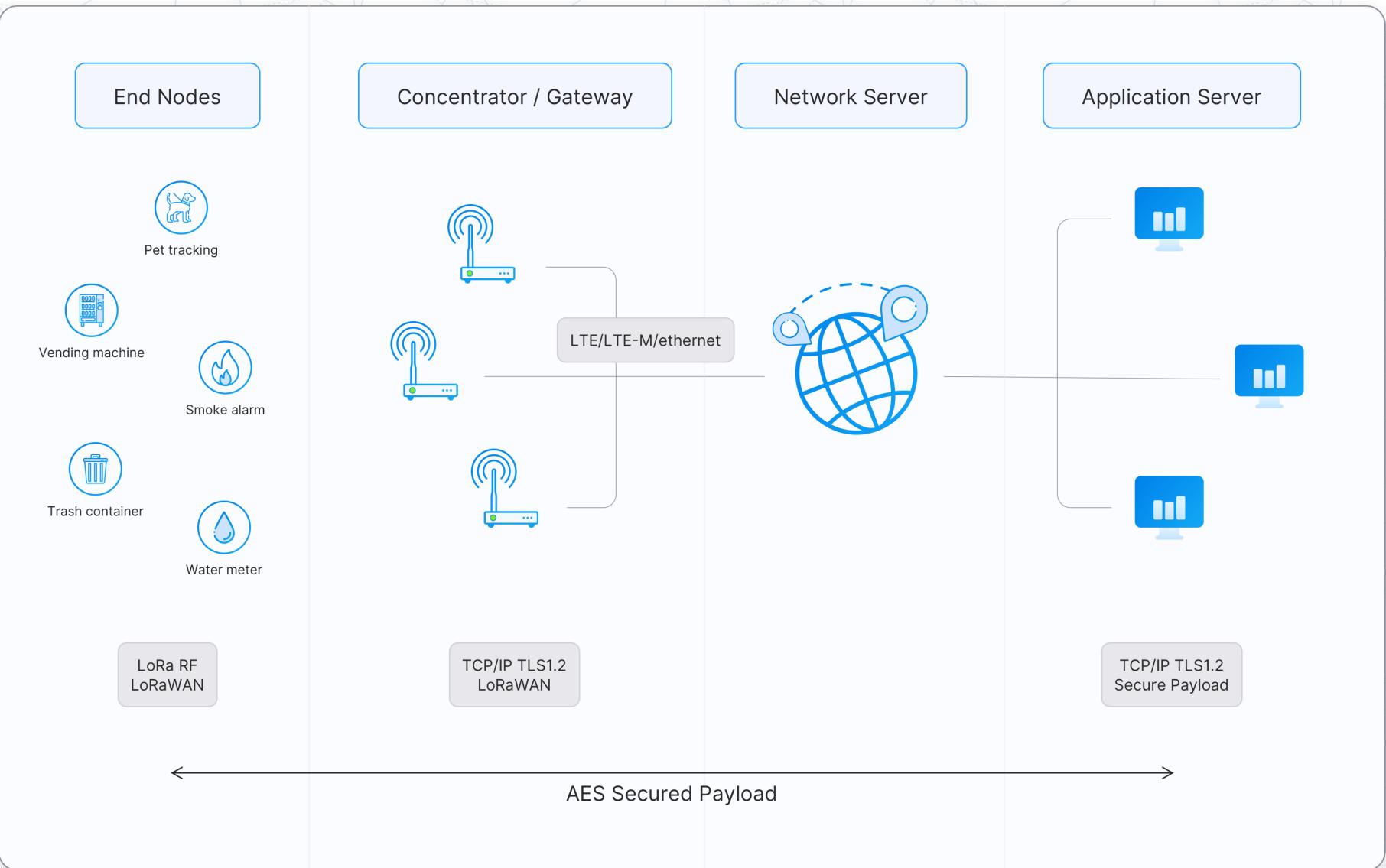
LoRaWAN Network Architecture

- While the specification defines the technical implementation, it does not define any commercial model or type of deployment (public, shared, private, enterprise) and so offers the industry the freedom to innovate and differentiate how it is used
- The LoRaWAN® specification is developed and maintained by the LoRa Alliance®: an open association

LoRaWAN Network Devices

- LoRaWAN Device Types:
 - End Devices
 - Gateways
 - Network Server
 - Application servers
 - Join Server

LoRaWAN Architecture



LoRaWAN: End Device

- A LoRaWAN end device can be a sensor, an actuator, or both
- They are often battery operated
- These end devices are wirelessly connected to the LoRaWAN network through gateways using LoRa RF modulation
- *LoRaWAN end device - The Things Industries Generic Node Sensor Edition:*
 - An end device that consists of sensors like temperature, humidity, and fall detection



LoRaWAN Device Classes

- LoRaWAN has three different classes of end-point devices to address the different needs reflected in the wide range of applications:
 - Class A – Lowest power, bi-directional end-devices
 - Class B – Bi-directional end-devices with deterministic downlink latency
 - Class C – Lowest latency, bi-directional end-devices

LoRaWAN Device Class A

- The default class which must be supported by all LoRaWAN end-devices, class A communication is always initiated by the end-device and is fully asynchronous
- Each uplink transmission can be sent at any time and is followed by two short downlink windows, giving the opportunity for bi-directional communication, or network control commands if needed

LoRaWAN Device Class A

- This is an ALOHA type of protocol
 - Multiple access protocol for transmission of data via a shared network channel
- The end-device is able to enter low-power sleep mode for as long as defined by its own application:
 - There is no network requirement for periodic wake-ups

LoRaWAN Device Class A

- This makes class A the lowest power operating mode, while still allowing uplink communication at any time
- Because downlink communication must always follow an uplink transmission with a schedule defined by the end-device application, downlink communication must be buffered at the network server until the next uplink event

LoRaWAN Device Class B

- Bi-directional end-devices with deterministic downlink latency
- In addition to the class A initiated receive windows, class B devices are synchronized to the network using periodic beacons, and open downlink ‘ping slots’ at scheduled times

LoRaWAN Device Class B

- This provides the network the ability to send downlink communications with a deterministic latency, but at the expense of some additional power consumption in the end-device
- The latency is programmable up to 128 seconds to suit different applications, and the additional power consumption is low enough to still be valid for battery powered applications

LoRaWAN Device Class C

- Lowest latency, bi-directional end-devices
- In addition to the class A structure of uplink followed by two downlink windows, class C further reduces latency on the downlink by keeping the receiver of the end-device open at all times that the device is not transmitting (half duplex)
 - Based on this, the network server can initiate a downlink transmission at any time on the assumption that the end-device receiver is open, so no latency

LoRaWAN Device Class C

- The compromise is the power drain of the receiver (up to ~50mW) and so class C is suitable for applications where continuous power is available
- For battery powered devices, temporary mode switching between classes A & C is possible and is useful for intermittent tasks such as firmware over-the-air updates

LoRaWAN: Gateways

- Each gateway is registered (using configuration settings) to a LoRaWAN network server
- A gateway receives LoRa messages from end devices and simply forwards them to the LoRaWAN network server
- Gateways are connected to the Network Server using a backhaul like Cellular (3G / 4G / 5G), WiFi, Ethernet, fiber-optic or 2.4 GHz radio links

Types of LoRaWAN Gateways

- LoRaWAN gateways can be categorized into indoor (picocell) and outdoor (macrocell) gateways
- Indoor gateways are cost-effective and suitable for providing coverage in places like deep-indoor locations
 - These gateways have internal antennas or external 'pigtail' antennas
 - Depending on the indoor physical environment some indoor gateways can receive messages from sensors located several kilometers away



The Things Indoor gateway designed to be directly plugged into an AC power outlet

Types of LoRaWAN Gateways

- Outdoor gateways provide a larger coverage than the indoor gateways
 - Are suitable for providing coverage in both rural and urban areas
- These gateways can be mounted on cellular towers, the rooftops of very tall buildings, metal pipes (masts) etc.
- Usually an outdoor gateway has an external antenna (i.e. Fiberglass antenna) connected using a coaxial cable
- You can convert some indoor gateways to outdoor gateways using water/dust proof enclosures and adding external antennas



Tektelic Enterprise
Outdoor Gateway

LoRaWAN: Network Server

- The Network Server manages gateways, end-devices, applications, and users in the entire LoRaWAN network
- A typical LoRaWAN Network Server has the following features:
 - Establishing secure 128-bit AES connections for the transport of messages between end-devices and the Application Server (end-to-end security)
 - Validating the authenticity of end devices and integrity of messages
 - Deduplicating uplink messages
 - Selecting the best gateway for routing downlink messages

LoRaWAN: Network Server

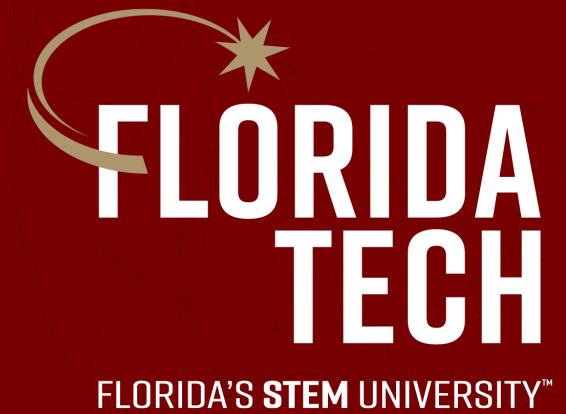
- Sending ADR commands to optimize the data rate of devices
- Device address checking
- Providing acknowledgements of confirmed uplink data messages
- Forwarding uplink application payloads to the appropriate application servers
- Routing uplink application payloads to the appropriate Application Server
- Forwarding Join-request and Join-accept messages between the devices and the join server
- Responding to all MAC layer commands

LoRaWAN: Application Server

- Processes application-specific data messages received from end devices
- It also generates all the application-layer downlink payloads and sends them to the connected end devices through the Network Server
- A LoRaWAN network can have more than one Application Server
- The collected data can be interpreted by applying techniques like machine learning and artificial intelligence to solve business problems

LoRaWAN: Join Server

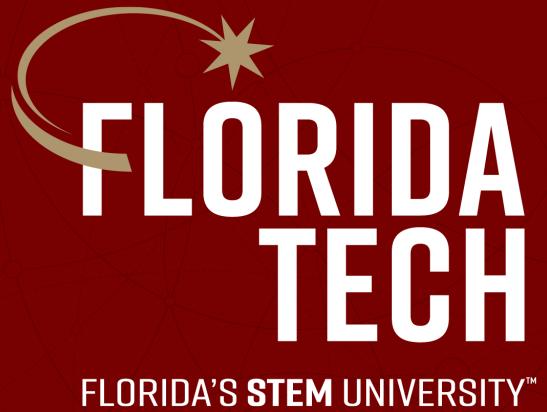
- Assists in secure device activation, root key storage, and session key generation
- The join procedure is initiated by the end device by sending the Join-request message to the Join Server through the Network Server
- The Join-server processes the Join-request message, generates session keys, and transfers NwkSKey and AppSKey to the Network server and the Application server respectively
- The Join Server was first introduced with LoRaWAN v1.1



**Thank you.
Questions?**

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

19. LoRaWAN Security

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

LoRaWAN

Message Types

Adaptive Data Rates

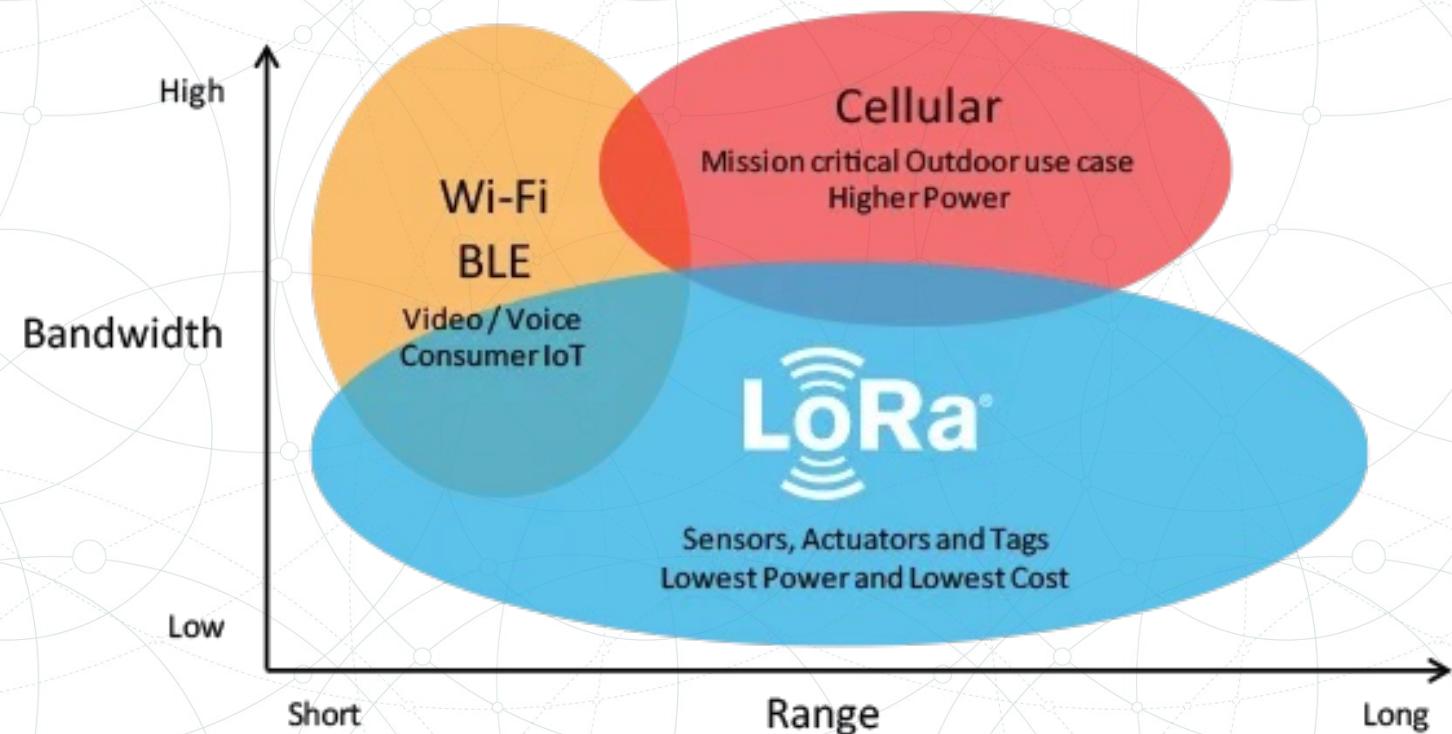
Security

Recall: LoRaWAN

- Long Range (LoRa) Wide Area Network (WAN)
 - “A Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated ‘things’ to the internet in regional, national or global networks,
 - And support targets key Internet of Things (IoT) requirements such as bi-directional communication, end-to-end security, mobility and localization services”

Recall: LoRaWAN as Wireless Standard

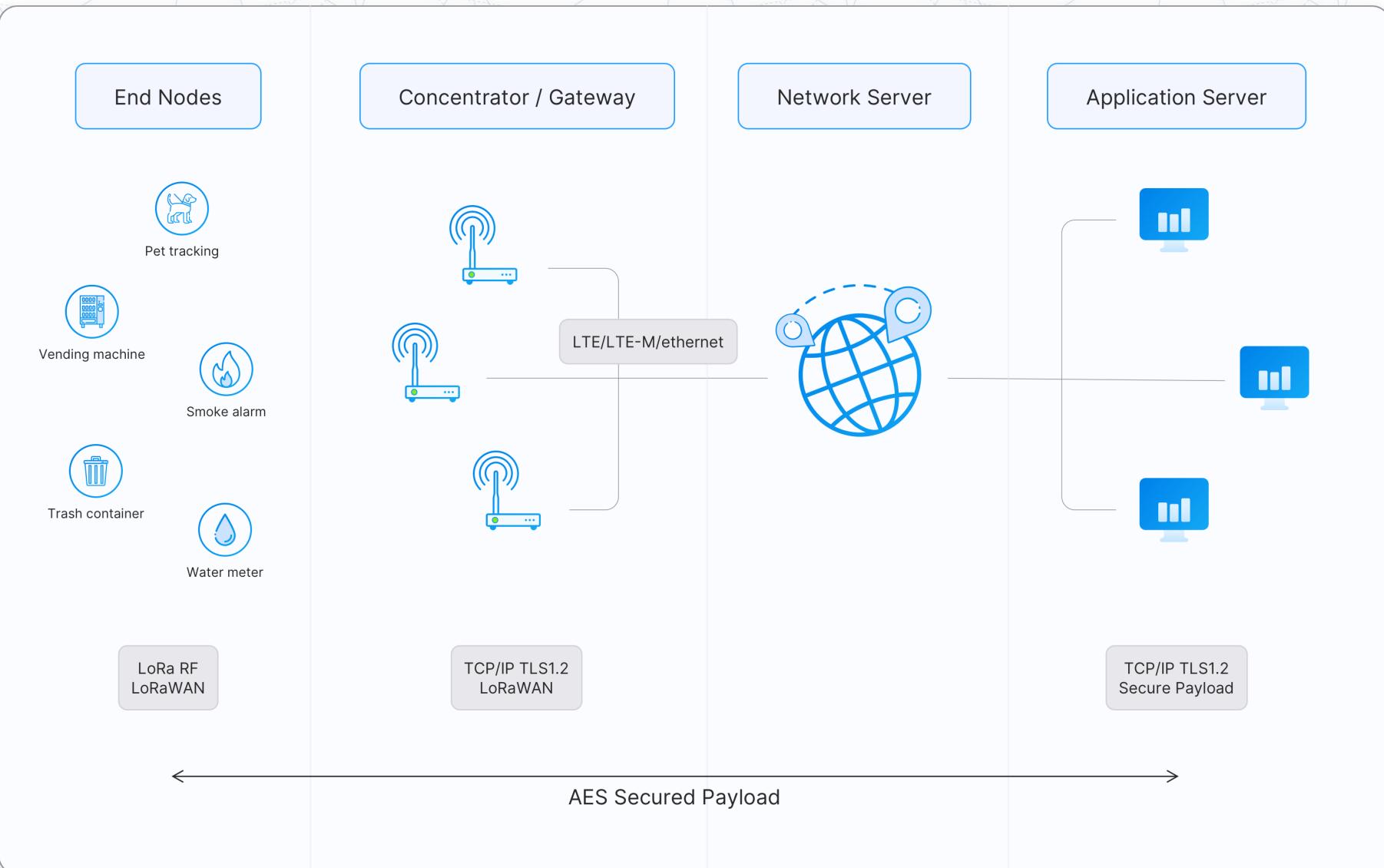
- Suitable for transmitting small size payloads (like sensor data) over long distances
 - Greater communication range with low bandwidths than other competing wireless data transmission technologies



Recall: LoRaWAN Network Devices

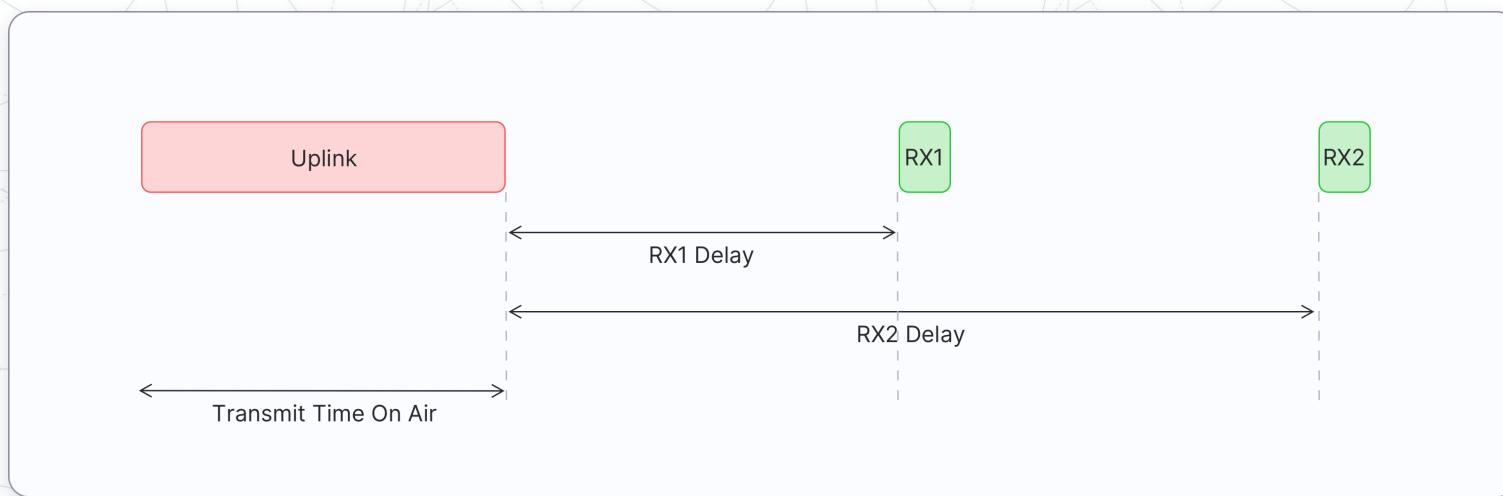
- LoRaWAN Device Types:
 - End Devices
 - Gateways
 - Network Server
 - Application servers
 - Join Server

Recall: LoRaWAN Architecture



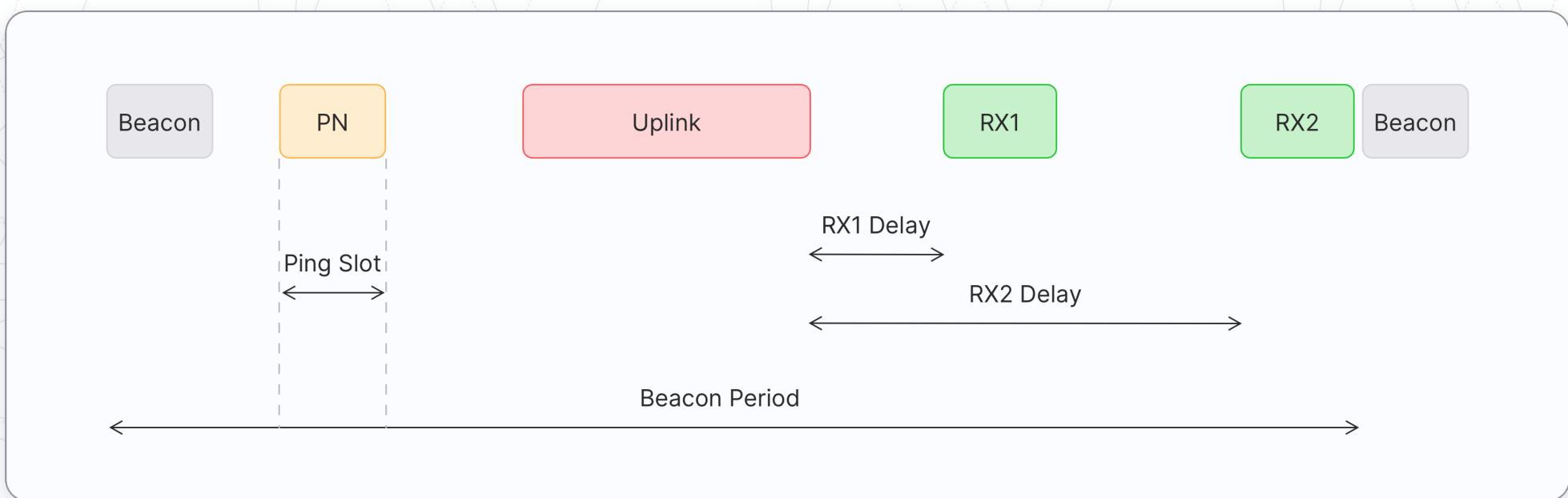
Recall: LoRaWAN Device Class A

- Communication is always initiated by the end-device
- A device can send an uplink message at any time
 - Once the uplink transmission is completed the device opens two short receive (downlink) windows



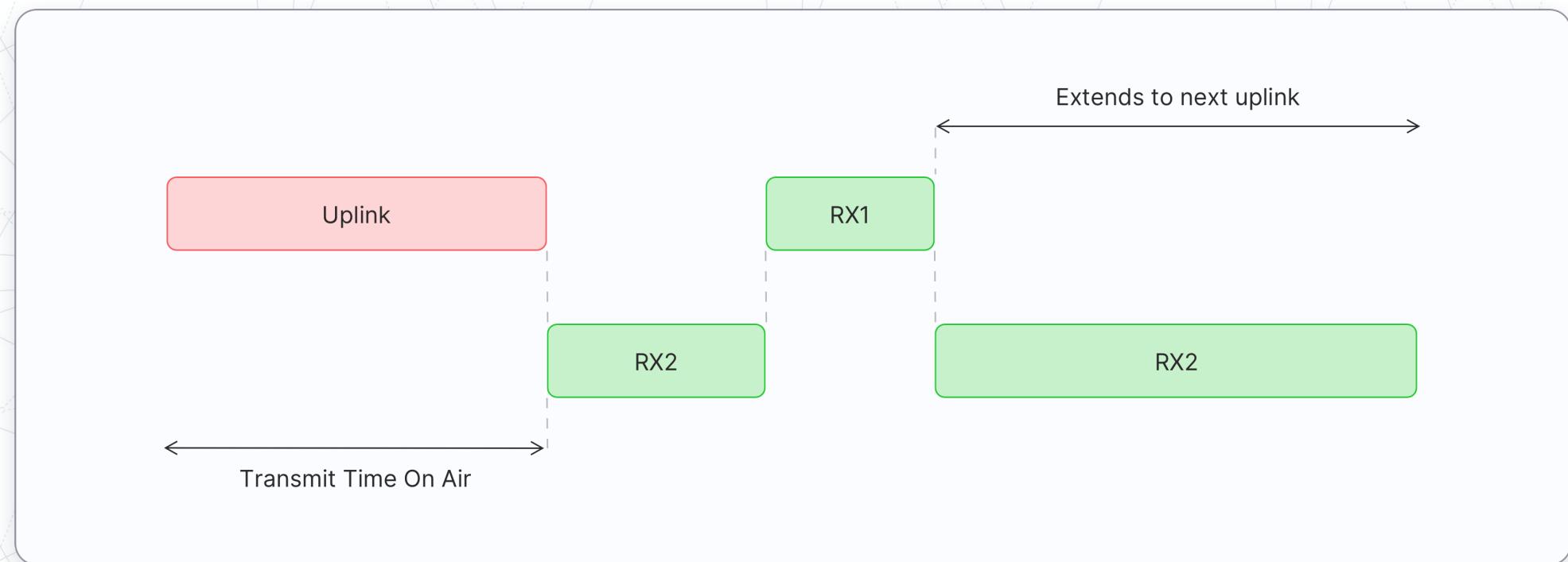
Recall: LoRaWAN Device Class B

- Open scheduled receive windows for receiving downlink messages from the network server



Recall: LoRaWAN Device Class C

- Extend Class A by keeping the receive windows open unless they are transmitting



LoRaWAN MAC Message Types

| LoRaWAN 1.0.x | LoRaWAN 1.1 | Description |
|-----------------------|-----------------------|--|
| Join-request | Join-request | An uplink message, used by the over-the-air activation (OTAA) procedure |
| Join-accept | Join-accept | A downlink message, used by the over-the-air activation (OTAA) procedure |
| Unconfirmed Data Up | Unconfirmed Data Up | An uplink data frame, confirmation is not required |
| Unconfirmed Data Down | Unconfirmed Data Down | A downlink data frame, confirmation is not required |
| Confirmed Data Up | Confirmed Data Up | An uplink data frame, confirmation is requested |
| Confirmed Data Down | Confirmed Data Down | A downlink data frame, confirmation is requested |
| RFU | Rejoin-request | 1.0.x - Reserved for Future Usage 1.1 - Uplink over-the-air activation (OTAA) Rejoin-request |
| Proprietary | Proprietary | Used to implement non-standard message formats |

LoRaWAN: Join-Request

- The Join-request message is always initiated by an end device and sent to the Network Server
- In LoRaWAN versions earlier than 1.0.4 the Join-request message is forwarded by the Network Server to the Application Server
 - In LoRaWAN 1.1 and 1.0.4+, the Network Server forwards the Join-request message to the device's Join Server
- The Join-request message is not encrypted

LoRaWAN: Join-Accept

- In LoRaWAN versions earlier than 1.0.4 the Join-accept message is generated by the Application Server
 - In LoRaWAN 1.1 and 1.0.4+ the Join-accept message is generated by the Join Server
 - In both cases the message passes through the Network Server
- Then the Network Server routes the Join-accept message to the correct end-device

LoRaWAN: Join-Accept

- The Join-accept message is encrypted as follows
 - In LoRaWAN 1.0, the Join-accept message is encrypted with the AppKey
 - In LoRaWAN 1.1, the Join-accept message is encrypted with different keys:

| If triggered by | Encryption Key |
|---------------------------------|----------------|
| Join-request | NwkKey |
| Rejoin-request type 0, 1, and 2 | JSEncKey |

Message Integrity Code (MIC)

- The Message Integrity Code (MIC) ensures the integrity and authenticity of a message
- The message integrity code is calculated over all the fields in the message and then added to the message itself

| Message Type | Fields |
|--------------------------------|--|
| Join-request | MHDR JoinEUI DevEUI DevNonce |
| Join-accept | MHDR JoinNonce NetID DevAddr DLSettings RxDelay CFList |
| Rejoin-request Type 0 and 2 | MHDR Rejoin Type NetID DevEUI RJcount0 |
| Rejoin-request Type 1 | MHDR Rejoin Type JoinEUI DevEUI RJcount1 |
| Data messages (up and down) | MHDR FHDR FPort FRMPayload |

LoRaWAN Data Messages

- Are used to transport both MAC commands and application data which can be combined together in a single message

- Data messages can be confirmed or unconfirmed



LoRaWAN End Device Activation

- Every end device must be registered with a network before sending and receiving messages
 - This procedure is known as activation
- There are two activation methods available:
 - Over-The-Air-Activation (OTAA)
 - Activation By Personalization (ABP)

OTAA

- The most secure and recommended activation method for end devices
- Devices perform a join procedure with the network, during which a dynamic device address is assigned and security keys are negotiated with the device
- The join procedure for LoRaWAN 1.0.x and 1.1 is slightly different

OTAA in LoRaWAN 1.1

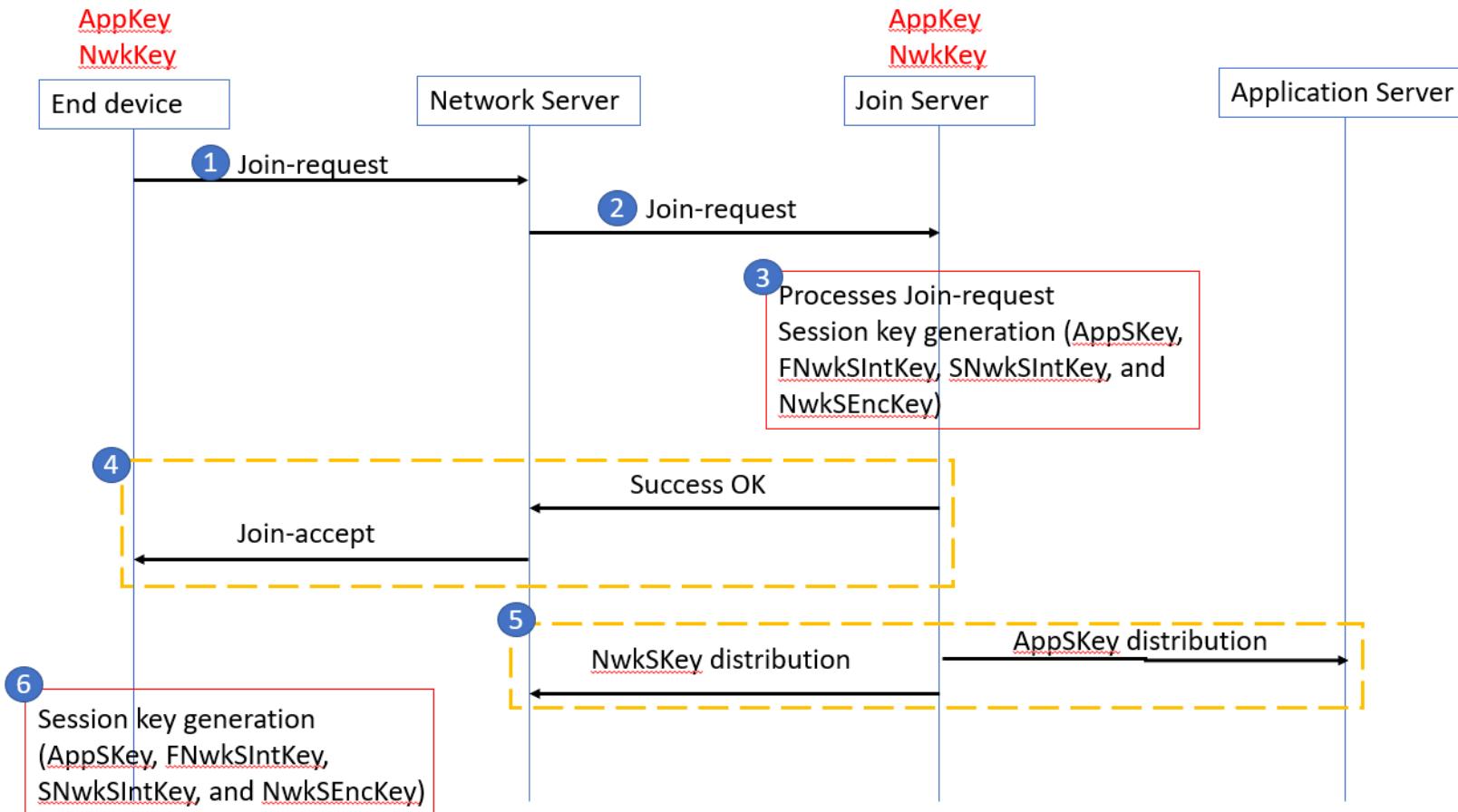
- The join procedure requires two MAC messages to be exchanged between the end device and the Join Server:
 - Join-request - from end device to the Join Server
 - Join-accept - from Join Server to the end device
- Before activation, the JoinEUI, DevEUI, AppKey, and NwkKey should be stored in the end device

OTAA in LoRaWAN 1.1

- The AppKey and NwkKey are AES-128 bit secret keys known as root keys
- The matching AppKey, NwkKey, and DevEUI should be provisioned onto the Join Server that will assist in the processing of the join procedure and session key derivation
- The JoinEUI and DevEUI are not secret and visible to everyone
- **Note:** The AppKey and NwkKey are never sent over the network

OTAA

- The join procedure is always initiated by the end device
- The end device sends the Join-request message to the network that is going to be joined



OTAA: Join Request #1

| | | |
|---------|---------|----------|
| 8 bytes | 8 bytes | 2 bytes |
| JoinEUI | DevEUI | DevNonce |

- The Join-request message consists of the following fields
 - JoinEUI – a 8 bytes global application identifier that uniquely identifies the Join Server that can assist in the processing of the Join-request and derivation of the session keys
 - DevEUI – a 8 bytes global device identifier that uniquely identifies the end-device
 - DevNonce – a 2-byte counter, starting at 0 when the device is initially powered up and incremented with every Join-request
 - The DevNonce value is used to prevent replay attacks

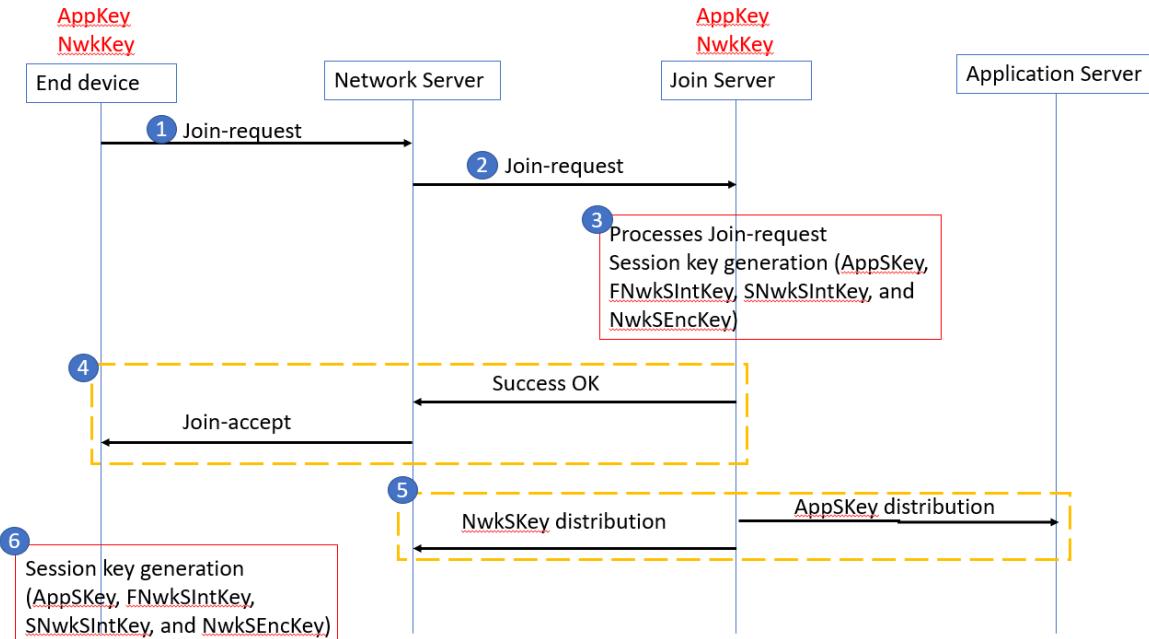
OTAA: Join Request #1

| | | |
|---------|---------|----------|
| 8 bytes | 8 bytes | 2 bytes |
| JoinEUI | DevEUI | DevNonce |

- MIC is calculated over all the fields in the Join-request message using the NwkKey
 - The calculated MIC is then added to the Join-request message
- **Note:** The NwkKey is not sent with the Join-request message, and the Join-request message is not encrypted but sent as plain text
- The Join-request message can be transmitted using any data rate and using one of the region-specific join channels and travels through one or more gateways to the Network Server
- **Note:** No response is given to the end-device if the Join-request is not accepted

OTAA Steps

- Step 2:
 - The Network Server forwards the Join-request message to the corresponding Join Server
- Step 3:
 - The Join Server processes the Join-request message
 - The Join Server will generate all the session keys (AppSKey, FNwkSIntKey, SNwkSIntKey, and NwkSEncKey) if the end-device is permitted to join the network



OTAA Steps #4

- If the above step 2-3 gets success, the Network Server generates the Join-accept message
- The Join-accept message consists of the following fields:

| 1 byte | 3 bytes | 4 bytes | 1 bytes | 1 bytes | 16 bytes |
|-----------|---------|---------|------------|---------|----------|
| JoinNonce | NetID | DevAddr | DLSettings | RXDelay | CFList |

- JoinNonce – a device specific counter value provided by the Join Server and used by the end device to derive the session keys, FNwkSIntKey, SNwkSIntKey, NwkSEncKey, and AppSKey

OTAA Steps: #4

| | | | | | |
|-----------|---------|---------|------------|---------|----------|
| 1 byte | 3 bytes | 4 bytes | 1 bytes | 1 bytes | 16 bytes |
| JoinNonce | NetID | DevAddr | DLSettings | RxDelay | CFList |

- NetID – a 24-bit unique network identifier
- DevAddr – a 32-bit device address assigned by the Network Server to identify the end device within the current network
- DLSettings – a 1-byte field consisting of downlink settings which the end device should use
- RxDelay – contains delay between TX and RX
- CFList – an optional list of channel frequencies for the network the end-device is joining
 - These frequencies are region-specific

OTAA Steps #4

- The Message Integrity Code (MIC) is calculated over all the fields in the Join-accept message using NwkKey (for LoRaWAN 1.0 devices) or JSIntKey (for LoRaWAN 1.1 devices)
 - The calculated MIC is then added to the Join-accept message
- The Join-accept message itself is then encrypted with the NwkKey
 - The Network Server uses an AES operation in ECB mode to encrypt the join-accept message

OTAA Steps #4

- The Join-accept message is encrypted with the NwkKey (if triggered by Join-request) or JSEncKey (if triggered by Rejoin-request)
- Then the Network Server sends the encrypted Join-accept message back to the end device as a normal downlink
- **Note:** No response is given to the end-device if the Join-request message is not accepted by the Network Server

OTAA Step #5-6

- Step 5:
 - The Join Server sends the AppSKey to the Application Server and the three network session keys (FNwkSIntKey, SNwkSIntKey, and NwkSEncKey) to the Network Server
- Step 6
 - The end-device decrypts the Join-accept message using AES operation
 - The end device uses AppKey, NwkKey, and JoinNonce to generate session keys

OTAA Step #6

- For LoRaWAN 1.1 devices,
 - AppSKey is derived from AppKey
 - FNwkSIntKey, SNwkSIntKey, and NwkSEncKey are derived from the NwkKey
- The end device is now activated on the Network

OTAA Step #6

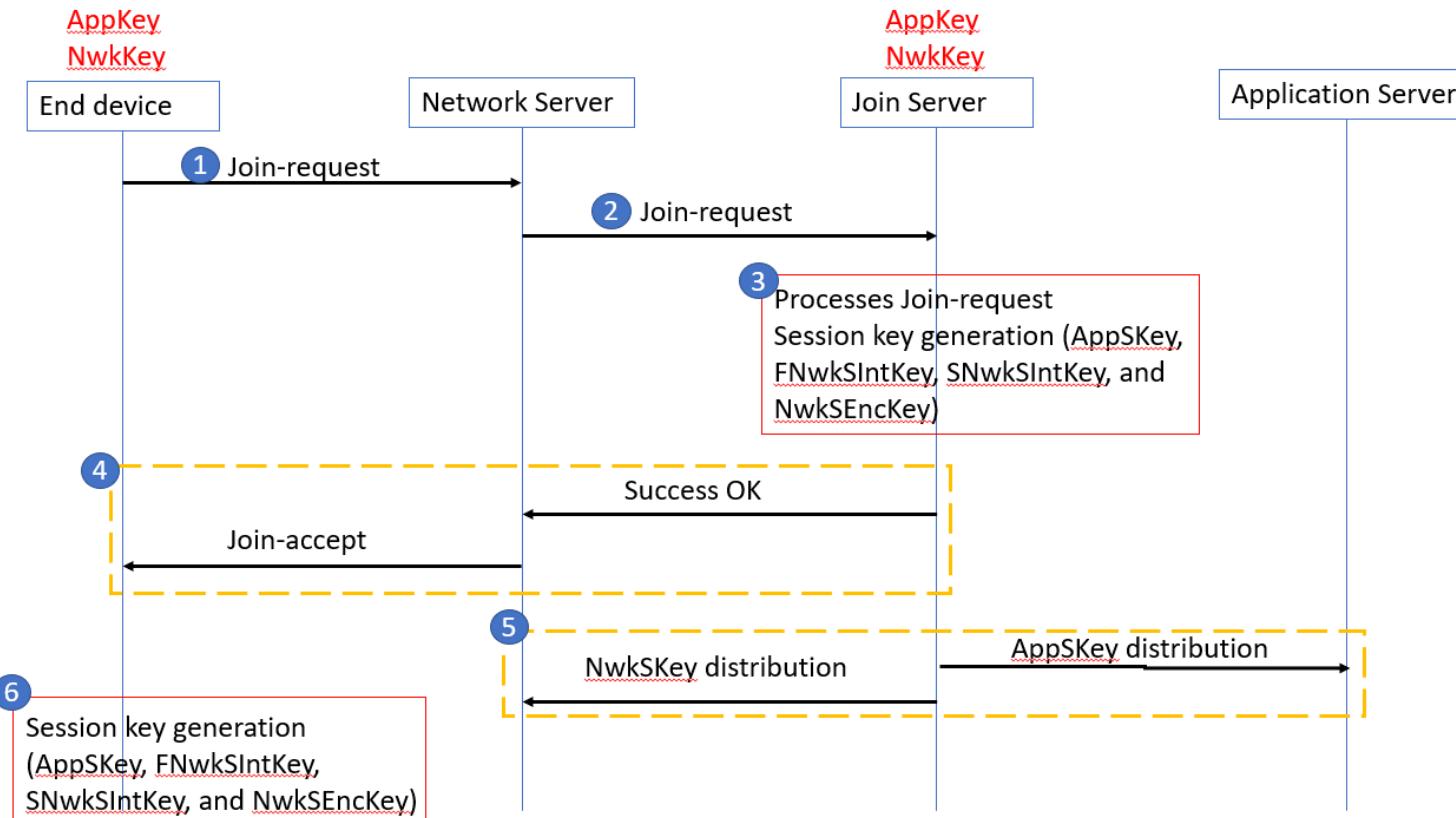
- After activation, the following additional information is stored in the end device:
- DevAddr:
 - 32-bit device address assigned by the Network Server to identify the end device within the current network
- FNwkSIntKey:
 - Network session key that is used by the end device to calculate the MIC (partially) of all uplink data messages for ensuring message integrity

OTAA Step #6

- SNwkSIntKey:
 - Network session key that is used by the end device to calculate the MIC (partially) of all uplink data message and calculate the MIC of all downlink data messages for ensuring message integrity
- NwkSEncKey:
 - Network session key that is used to encrypt and decrypt the payloads with MAC commands of the uplink and downlink data messages for ensuring message confidentiality

OTAA Step #6

- AppSKey:
 - Session key used by both the Application Server and the end device to encrypt and decrypt the application data in the data messages for ensuring message confidentiality



Activation By Personalization (ABP)

- ABP directly ties an end-device to a pre-selected network, bypassing the over-the-air-activation procedure
 - Requires hardcoding the device address as well as the security keys in the device
- ABP is the less secure activation method, and also has the downside that devices can not switch network providers without manually changing keys in the device
 - A Join Server is not involved in the ABP process
- An end device activated using the ABP method can only work with a single network and keeps the same security session for its entire lifetime

ABP in LoRaWAN 1.1

- The DevAddr and the four-session keys FNwkSIntKey, SNwkSIntKey, NwkSEncKey, and AppSKey are directly stored into the end device instead of the DevEUI, JoinEUI, AppKey, and NwkKey
- The same DevAddr, FNwkSIntKey, SNwkSIntKey, and NwkSEncKey should be stored in the Network Server and the AppSKey should be stored in the Application Server



LoRaWAN Data Rates

- In addition to frequency hopping, all communication packets between end-devices and gateways also include a variable 'Data rate' (DR) setting
- The selection of the DR allows a dynamic trade-off between communication range and message duration
- Also, due to the spread spectrum technology, communications with different DRs do not interfere with each other and create a set of virtual 'code' channels increasing the capacity of the gateway

LoRaWAN Data Rates

- To maximize both battery life of the end-devices and overall network capacity, the LoRaWAN network server manages the DR setting and RF output power for each end-device individually by means of an Adaptive Data Rate (ADR) scheme
- LoRaWAN baud rates range from 0.3 kbps to 50 kbps

LoRaWAN Adaptive Data Rate

- Adaptive Data Rate (ADR) is a mechanism for optimizing data rates, airtime and energy consumption in the network
- The ADR mechanism controls the following transmission parameters of an end device:
 - Spreading factor
 - Bandwidth
 - Transmission power

LoRaWAN ADR

- ADR can optimize device power consumption while ensuring that messages are still received at gateways
- When ADR is in use, the network server will indicate to the end device that it should reduce transmission power or increase data rate
- End devices which are close to gateways should use a lower spreading factor and higher data rate, while devices further away should use a high spreading factor because they need a higher link budget

LoRaWAN ADR

- ADR should be enabled whenever an end device has sufficiently stable RF conditions
 - This means that it can generally be enabled for static devices
- If the static end device can determine that RF conditions are unstable (for example, when a car is parked on top of a parking sensor), ADR should (temporarily) be disabled
- Mobile end devices should be able to detect when they are stationary for a longer times, and enable ADR during those times
- End devices decide if ADR should be used or not, not the application or the network

LoRaWAN Security

- LoRaWAN 1.0 specifies a number of security keys: NwkSKey, AppSKey and AppKey
- All keys have a length of 128 bits
- The algorithm used for this is AES-128, similar to the algorithm used in the 802.15.4 standard

LoRaWAN Session Keys

- When a device joins the network (this is called a join or activation), an application session key AppSKey and a network session key NwkSKey are generated
 - The NwkSKey is shared with the network, while the AppSKey is kept private
 - These session keys will be used for the duration of the session

LoRaWAN Session Keys

- The Network Session Key (NwkSKey) is used for interaction between the Node and the Network Server
- This key is used to validate the integrity of each message by its Message Integrity Code (MIC check)
- This MIC is similar to a checksum, except that it prevents intentional tampering with a message

LoRaWAN Session Keys

- For this MIC, LoRaWAN uses AES-CMAC
 - This validation is also used to map a non-unique device address (DevAddr) to a unique DevEUI and AppEUI

The MIC of the Join-accept message is computed using the NwkKey:

```
cmac = aes128_cmac(NwkKey, MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay  
| CFList)  
MIC = cmac[0..3]
```

LoRaWAN Session Keys

- The Application Session Key (AppSKey) is used for encryption and decryption of the payload
- The payload is fully encrypted between the Node and the Handler / Application Server component of The Things Network, which you can run on your own server
 - This means that nobody except you is able to read the contents of messages you send or receive

```
AppSKey = aes128_encrypt(AppKey, 0x02 | JoinNonce | JoinEUI | DevNonce | pad16)
```

LoRaWAN Session Keys

- These two session keys (NwkSKey and AppSKey) are unique per device, per session
- If you dynamically activate your device (OTAA), these keys are re-generated on every activation
- If you statically activate your device (ABP), these keys stay the same until you change them

```
FNwkSIntKey = aes128_encrypt(NwkKey, 0x01 | JoinNonce | JoinEUI | DevNonce | pad16)
```

```
SNwkSIntKey = aes128_encrypt(NwkKey, 0x03 | JoinNonce | JoinEUI | DevNonce | pad16)
```

```
NwkSEncKey = aes128_encrypt(NwkKey, 0x04 | JoinNonce | JoinEUI | DevNonce | pad16)
```

LoRaWAN Application Keys

- The application key (AppKey) is only known by the device and by the application
- Dynamically activated devices (OTAA) use the Application Key (AppKey) to derive the two session keys during the activation procedure
- In The Things Network you can have a default AppKey which will be used to activate all devices or customize the AppKey per device

```
AppSKey = aes128_encrypt(AppKey, 0x02 | JoinNonce | JoinEUI | DevNonce | pad16)
```

LoRaWAN Security

- By providing these two keys, it becomes possible to implement 'multi-tenant' shared networks without the network operator having visibility of the users payload data
- The keys can be Activated By Personalisation (ABP) on the production line or during commissioning, or can be Over-The-Air Activated (OTAA) in the field
 - OTAA allows devices to be re-keyed if necessary

LoRaWAN: Frame Counters

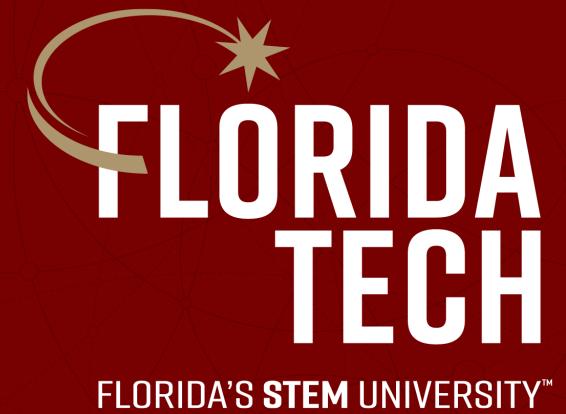
- Anyone will be able to capture and store messages
 - It's not possible to read these messages without the AppSKey, because they're encrypted
 - Nor is it possible to tamper with them without the NwkSKey, because this will make the MIC check fail
 - It is however possible to re-transmit the messages
- These so-called replay attacks can be detected and blocked using frame counters

LoRaWAN: Frame Counters

- When a device is activated, these frame counters (FCntUp and FCntDown) are both set to 0
- Every time the device transmits an uplink message, the FCntUp is incremented and every time the network sends a downlink message, the FCntDown is incremented
- If either the device or the network receives a message with a frame counter that is lower than the last one, the message is ignored

LoRaWAN: Frame Counters

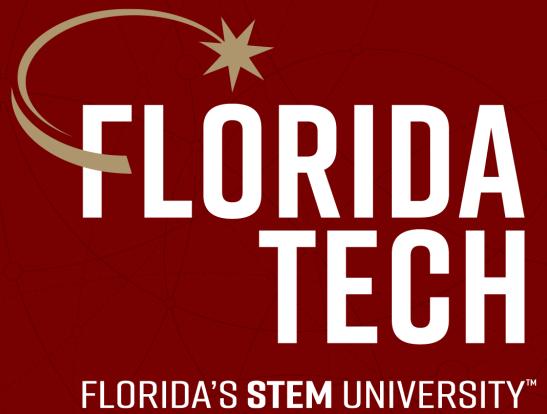
- These frame counters reset to 0 every time the device restarts (when you flash the firmware or when you unplug it)
- As a result, The Things Network will block all messages from the device until the FCntUp becomes higher than the previous FCntUp
- Therefore, you should re-register your device in the backend every time you reset it



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

20. Cellular Networks

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

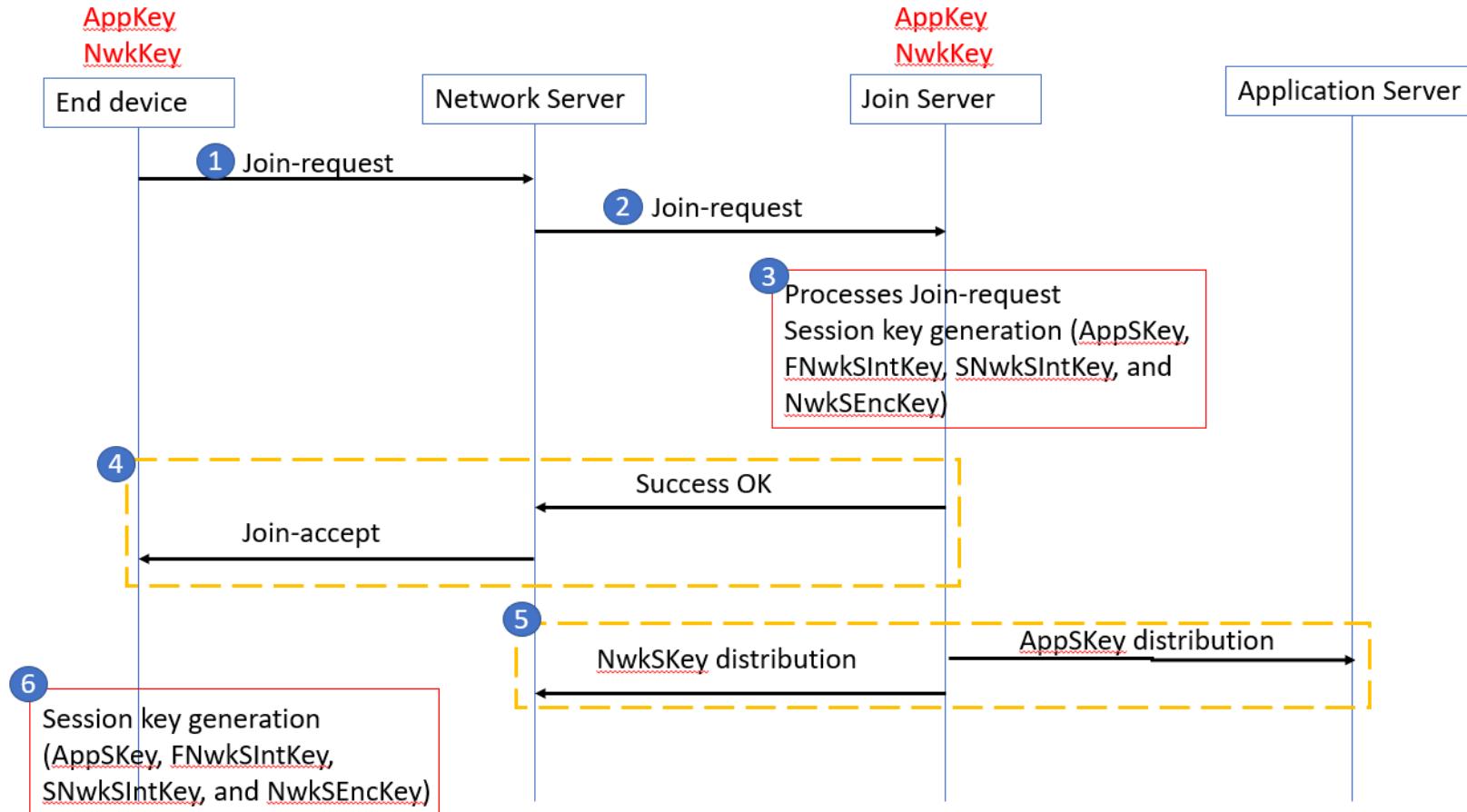
Outline

Cellular Networks

2G - Edge

Recall: OTAA

- The join procedure is always initiated by the end device
- The end device sends the Join-request message to the network that is going to be joined



Recall: LoRaWAN: Frame Counters

- Anyone will be able to capture and store messages
 - It's not possible to read these messages without the AppSKey, because they're encrypted
 - Nor is it possible to tamper with them without the NwkSKey, because this will make the MIC check fail
 - It is however possible to re-transmit the messages
- These so-called replay attacks can be detected and blocked using frame counters

Cellular Network Radio Frequencies

- Modern cell phones use a number of different frequencies to transmit data
 - Depends on the country and mobile operator network
- Some countries provide government-owned cellular services
 - Some lease spectrum to mobile operators to provide cellular access

Cellular Network Standards

- Responsible body for defining global cellular standards is 3GPP;
 - Composed of Association of Radio Industries and Businesses, Japan
 - The Alliance for Telecommunications Industry Solutions, US
 - Standards Association, China
 - The European Telecommunications Standards Institute, EU
 - Telecommunications Technology Association, Korea
 - Telecommunication Technology Committee, Japan

Cellular Network Standards

- “Release 98” comprising the majority of 3G
- “Release 8” in 2008 -> the first 4G LTE
- “Release 10” in 2011 -> advanced LTE
- 3GPP defines radio interfaces and backend network infrastructure components and protocols used

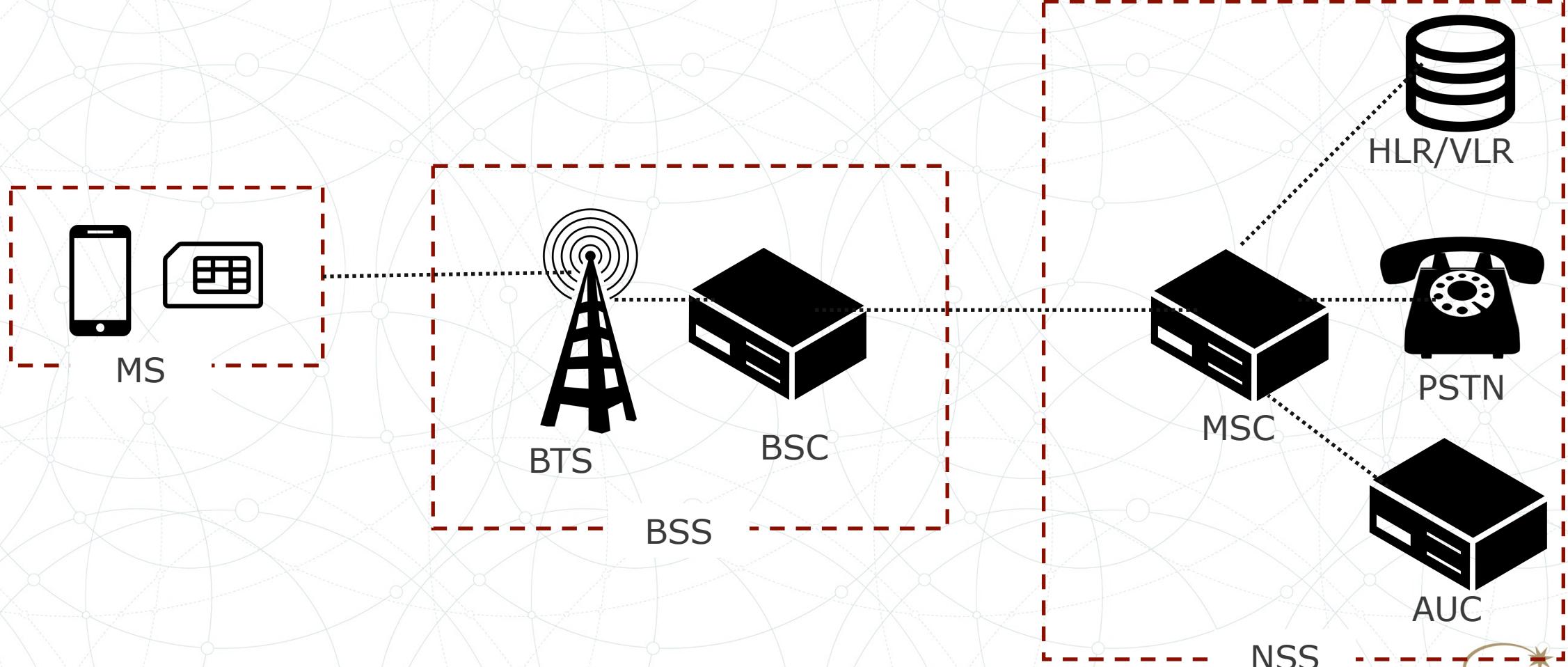
2G

- 2.5G Global System for Mobile (GSM), 2.75G General Packet Radio Service (GPRS), and Enhanced Data Rates for GSM Evolution (EDGE)
- Has been the most widespread cellular protocol used worldwide
 - Some devices still use it for backward-compatibility or back-up connection in the absence of other access opportunities

GSM Protocol

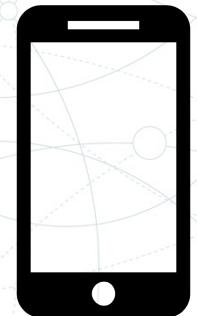
- Global System for Mobile Communications (GSM)
- First digital, circuit-switched network for carrying voice
- Expanded for carrying data (GPRS)
- Implemented cryptographic algorithms for security
- By mid-2010s, had over 90% of market share
- Obsoleted by LTE (4G); reached end-of-life for AT&T in 2017

GSM Network Model



GSM Mobile Station

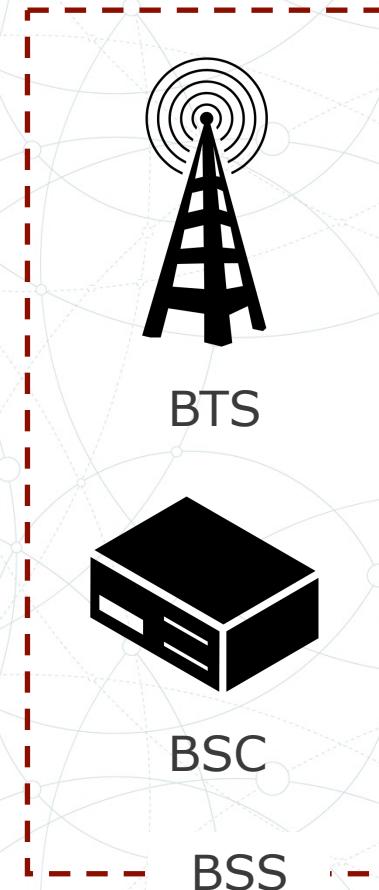
- Mobile Station (MS):
 - User handset/device that connects to the GSM network
- Subscriber Identity Mobile (SIM):
 - Removable media that identifies the unique International Mobile Subscriber Identifier (IMSI) for the mobile station and the 128-bit Authentication Key (K_i)
- International Mobile Subscriber Identifier (IMSI):
 - Unique identifier consisting of Mobile Country Code (3 digits), Mobile Network Code (2 | 3 digits), Mobile Subscriber Identification (10 | 15 digits)



MS

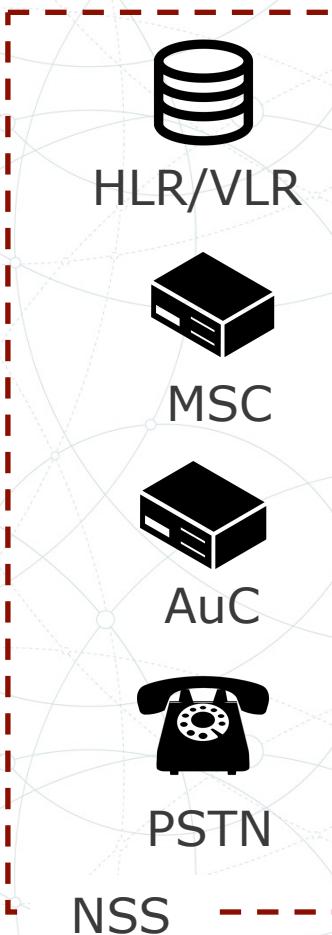
GSM Base Station Subsystem

- Base Transceiver Station (BTS):
 - Devices that facilitates radio connectivity for the mobile station (e.g., cell tower)
- Base Station Controller (BSC):
 - Facilitates the management of several BTSSs;
 - Handles radio allocation, BTS handovers, etc.
- Base Station Subsystem (BSS):
 - Consists of BTS/BSC;
 - GSM network component that connects mobile station and network switching system



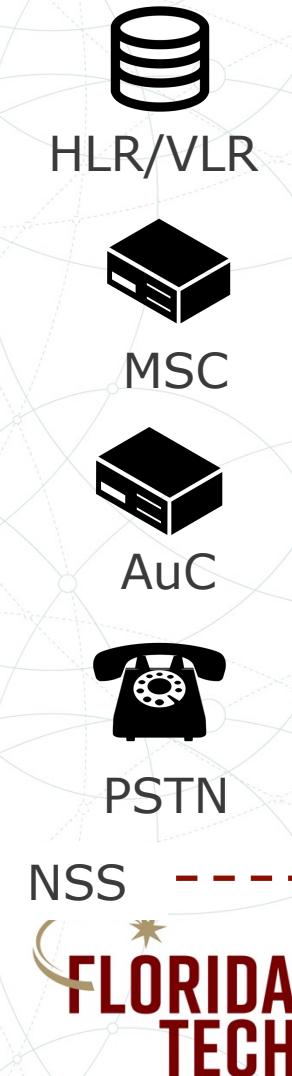
GSM Network Switching Subsystem

- Network Switching Subsystem (NSS):
 - Back-end network for GSM provider and services
- Home Location Register (HLR):
 - Database that contains the IMSI for each MS on the network
- Visitor Location Register (VLR):
 - Database that facilitates roaming operations for MS devices

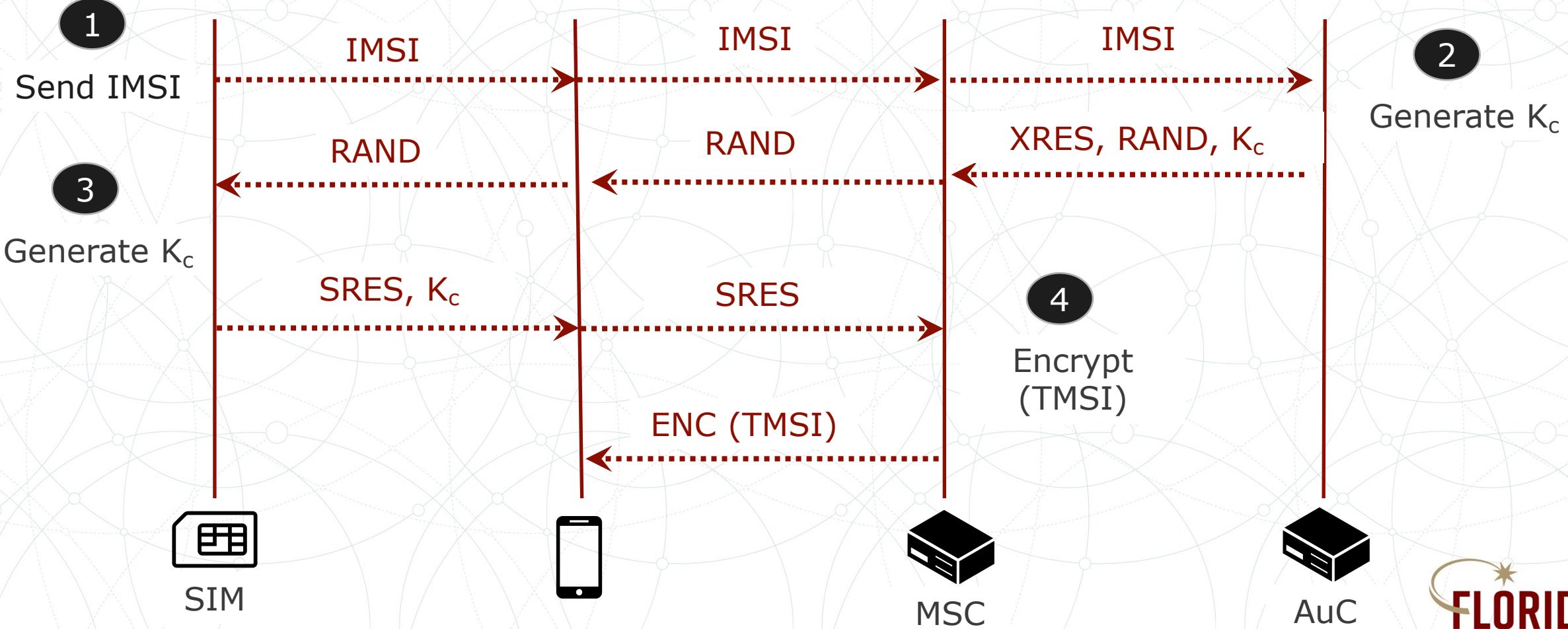


GSM Network Switching Subsystem

- Mobile Switching Center (MSC):
 - Responsible for routing
- Authentication Center (AuC):
 - Facilitates user identification and authentication for SIM cards
- Public Switched Telephone Network (PSTN):
 - Public interface to GSM network and other network providers



GSM Authentication



GSM Authentication

- Exchange validates the identity of the subscriber through the use of the IMSI and the associated subscriber key, K_i , stored on the SIM card
 - Involves the SIM, the MS, the MSC, and the AuC
- AuC and SIM have knowledge of IMSI and Ki as part of the device registration process
 - When the MS connects to network, SIM shares IMSI info which is forwarded to AuC

GSM Authentication

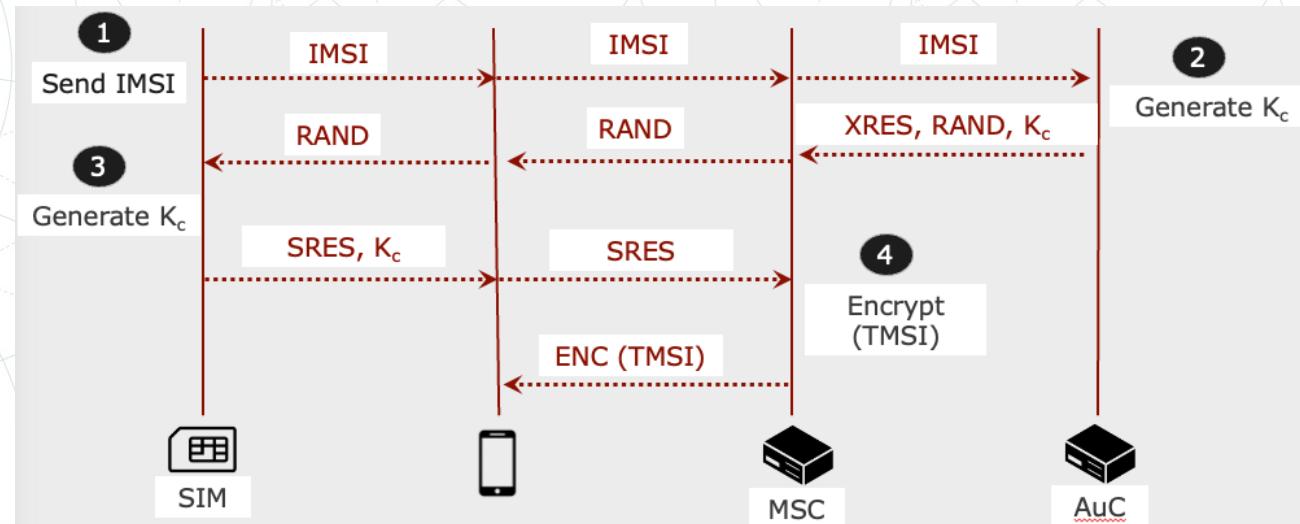
- The AuC retrieves K_i linked to IMSI and selects a random value RAND
- The RAND is used with two algorithms, A3 and A8, with the subscriber key K_i to generate the temporary cipher key K_c and the expected response (XRES)
- AuC shares K_c , RAND and XRES with the MSC, ending its role in the authentication process

GSM Authentication

- Next, MSC shares the RAND with the MS, which sends it to the SIM
 - Similar to AuC, the SIM uses the RAND to generate Kc and signed response SRES
 - SRES is delivered to MS, which forwards it over the air interface to the MSC for validation

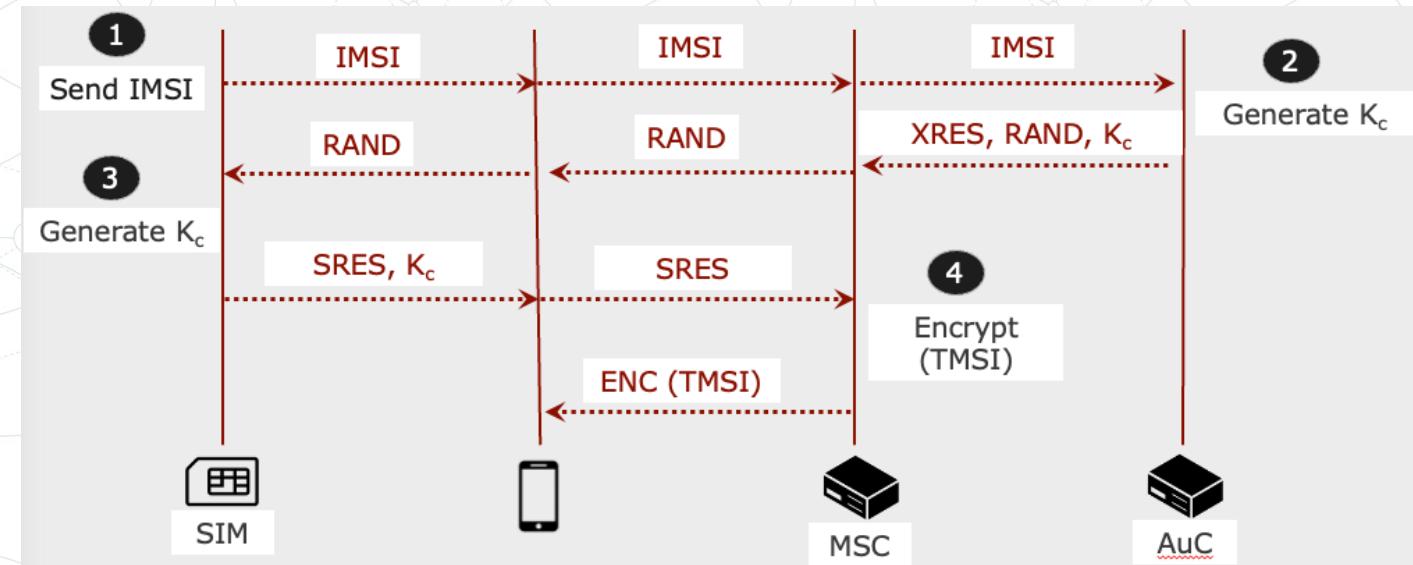
GSM Authentication

- The MSC compares SRES to XRES;
 - If match, validating the ME's identity
 - Then, MSC generates and encrypts TMSI for the ME to use and delivers it over the air interface



GSM Authentication: Vulnerability

- SIM card (and so the ME) is authenticated to the AuC
 - Prevent unauthorized devices accessing network services
- The authentication exchange does not validate the identity of the provider to the ME

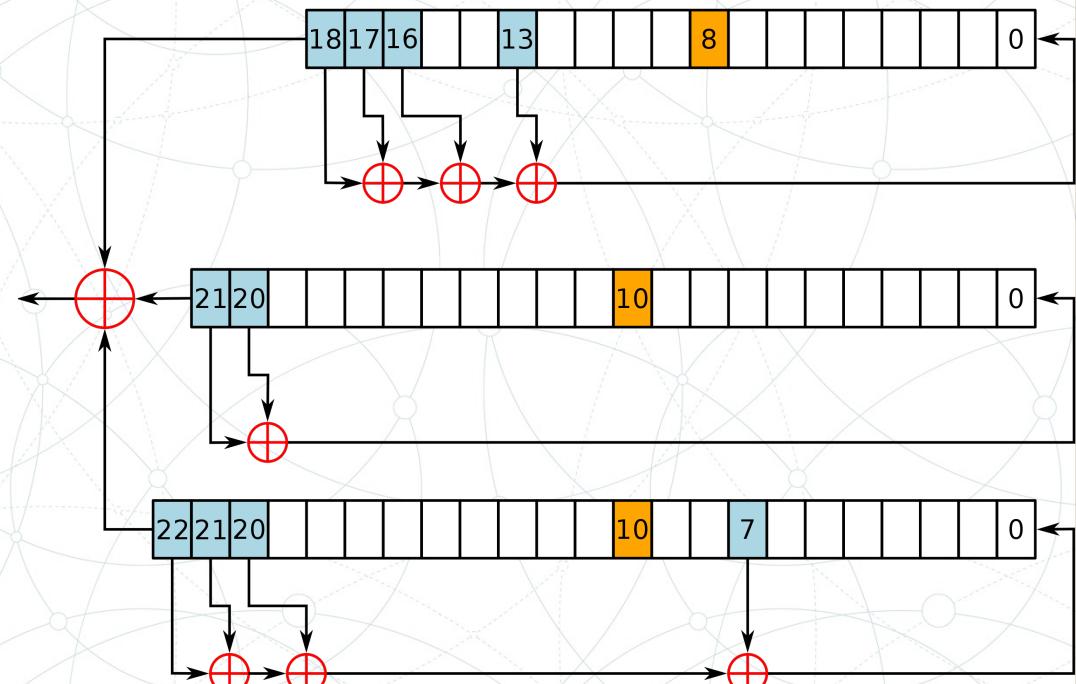


GSM Encryption

- Use the A5/1 cipher to provide confidentiality controls of traffic delivered over the GSM air interface between MS and BSC
- A5/1 is a stream cipher implemented using a Linear Feedback Shift Register (LFSR) mechanism
- Temporary cipher key K_c is used as the A5/1 input to generate keystream data
 - Ease of implementation in hardware and reduced implementation cost

A5/1

- Developed in 1987 (before GSM)
- 114 bits of GSM are XORed with 114 bits of Keystream material to produce encrypted data
- Uses three Linear Feedback Shift Registers (LFSR)



GSM Encryption

- The plaintext data is XOR'd with the keystream data to generate ciphertext
- Ciphertext is similarly XOR'd with matching keystream data at the recipient to decrypt

GSM Attacks

- Privacy attacks;
 - Leading to the disclosure of IMSI
- Confidentiality attacks;
 - Disclosure of voice and data communications over the GSM
- Integrity attacks;
 - Adversary manipulates or impersonates back-end GSM services to modify the content being delivered

GSM Eavesdropping

- Commercial tools are expensive
 - But, you can build your own GSM sniffer using inexpensive hardware
 - For ex. RTL-SDR
- GSM packet captures disclose basic information about the network, but do not reveal the contents of phone call/SMS etc. due to being encrypted

A5/1 Key Recovery

- Precomputed reference attack for full key recovery
- In 2008, gsm-tvoid; keystream data to known keystream
state information in lookup tables
 - Using set of precomputed 288 quadrillion possible entries (apprx 2tb storage), adversary recovers K_i in approx. 30mins
 - It was taken offline without explanation
 - Possible government intervention

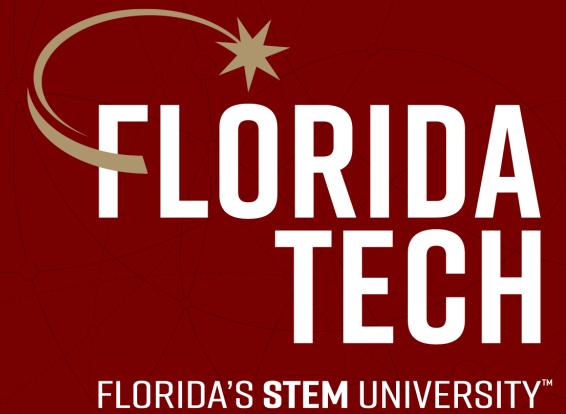
A5/1 Key Recovery

- In 2009, A5/1 cracking project reproduced the work previously published
 - As a plus, they distribute key recovery lookup tables through peer-to-peer networks to prevent them from being taken offline
- In 2011, ‘Kraken’, practical tool that integrates with AirProbe for effective capture and decryption of GSM traffic with the A5/1 tables
 - Later, ‘Pytacle’ tool improved features of the tool by adding play back capabilities for full and simple passive GSM decryption attack

<https://github.com/0xh4di/kraken>

Running Pytacle Requirements

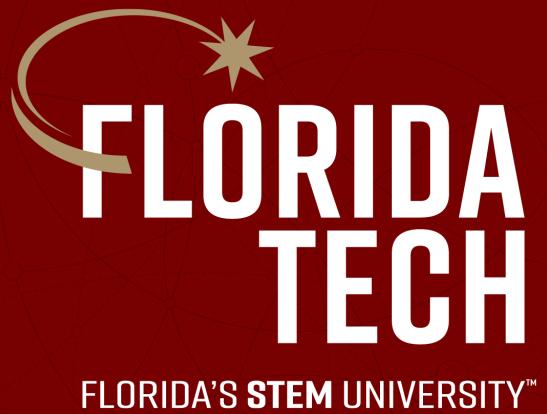
- RTL/SDR with AirProbe software for GSM capture
- Kraken software
- Pytacle software
- A5/1 tables on a temporary storage drive, apprx 1.6tb
- A5/1 tables written to one or more lookup drives, apprx 3tb



Thank you. Questions?

Dr. Abdullah Aydeger

Department of Computer Science



CSE 4820: Wireless and Mobile Security

21. Cellular Networks Ctd

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

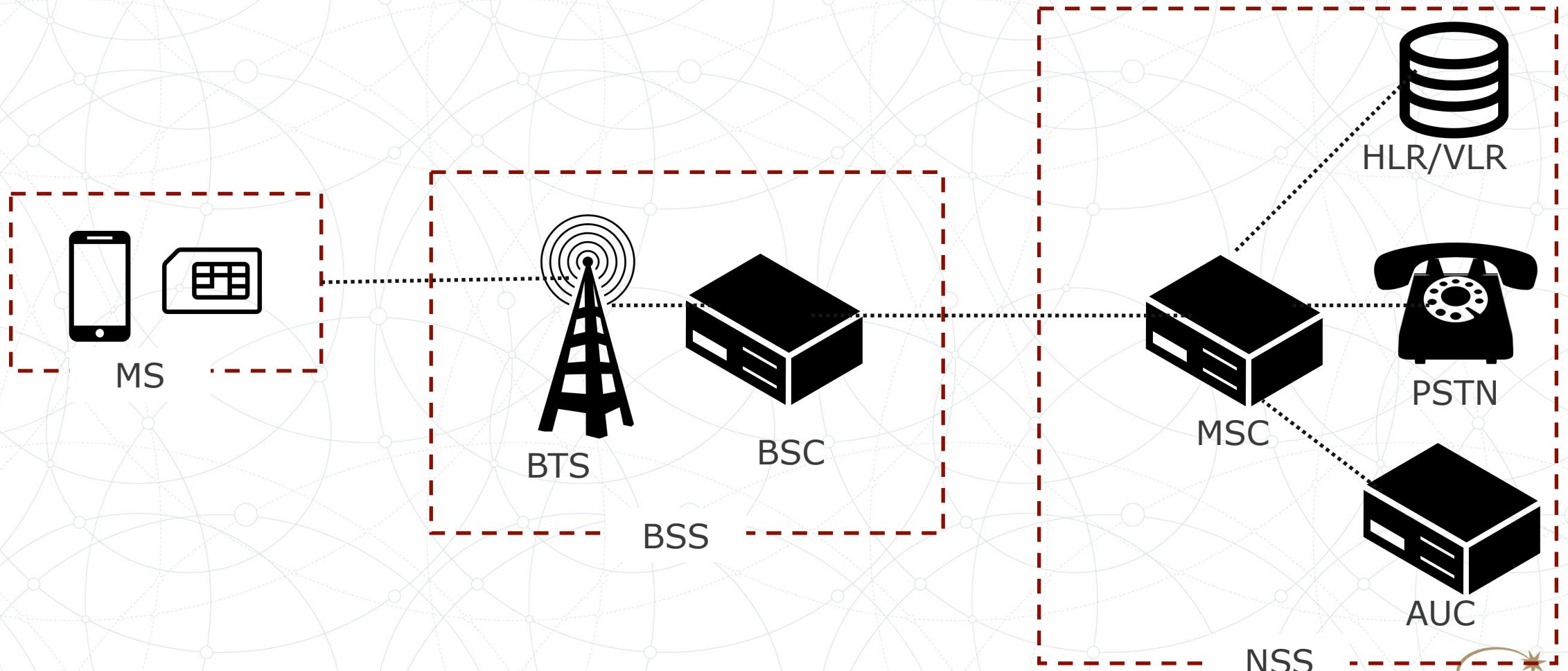
Outline

Cellular Networks

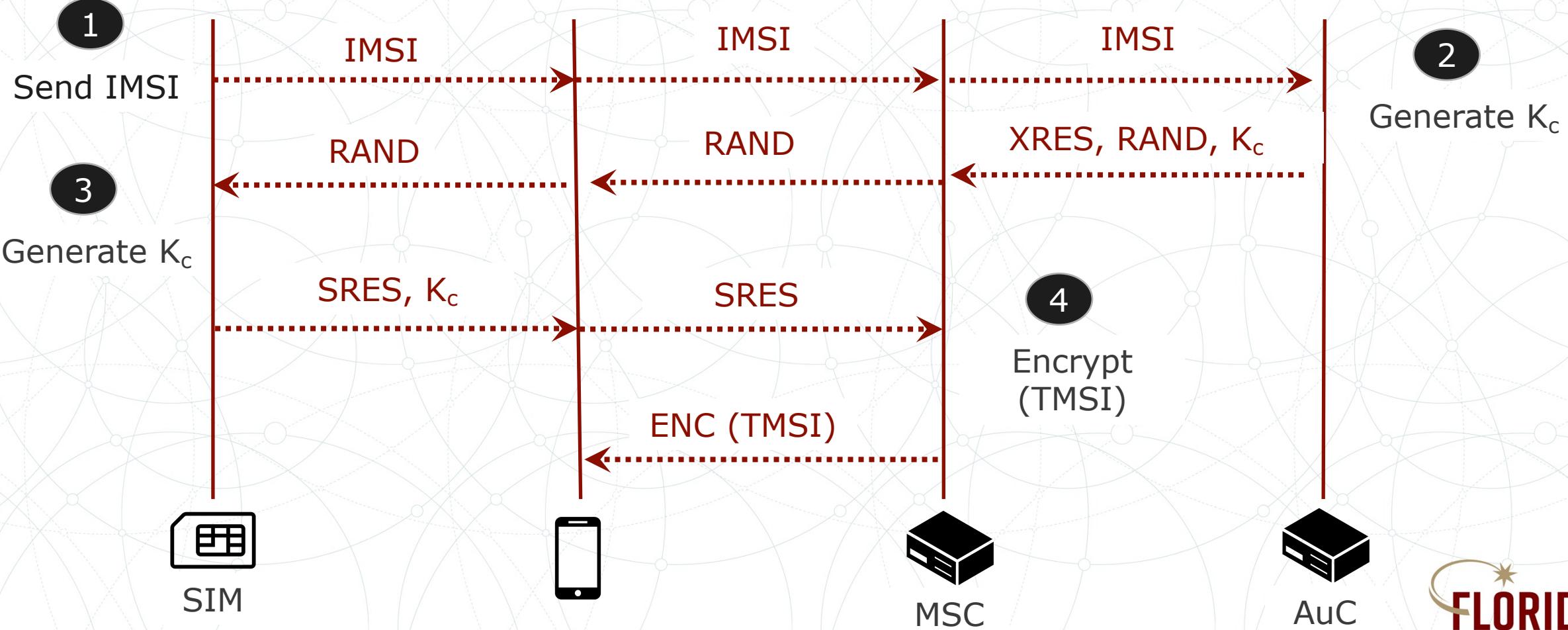
Femtocell

4G/LTE

Recall: GSM Network Model



Recall: GSM Authentication



Recall: A5/1 Key Recovery

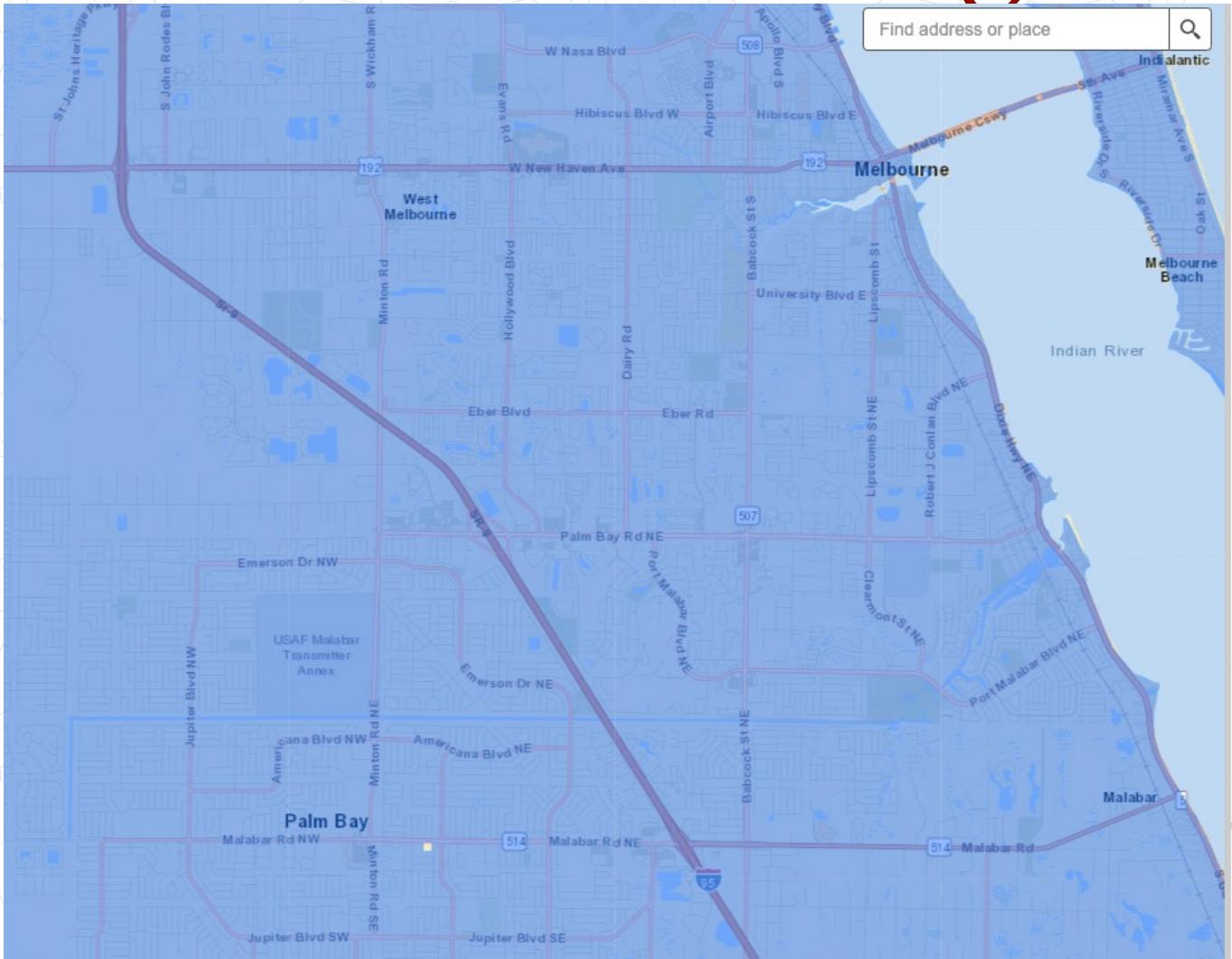
- Precomputed reference attack for full key recovery
- In 2008, gsm-tvoid; keystream data to known keystream
state information in lookup tables
 - Using set of precomputed 288 quadrillion possible entries (apprx 2tb storage), adversary recovers K_i in approx. 30mins
 - It was taken offline without explanation
 - Possible government intervention

GSM Attacks: IMSI Catcher

- An IMSI Catcher is a fake cell phone tower used to surreptitiously eavesdrop on mobile phones
- Sting-Ray phone tracker, manufactured by L3 Harris, is an example of an IMSI catcher distributed to law-enforcement/ military
- IMSI Catchers can work passively (by advertising MCC/MNC) or actively by (disrupting channel and forcing disconnect)



Network Coverage Map: AT&T

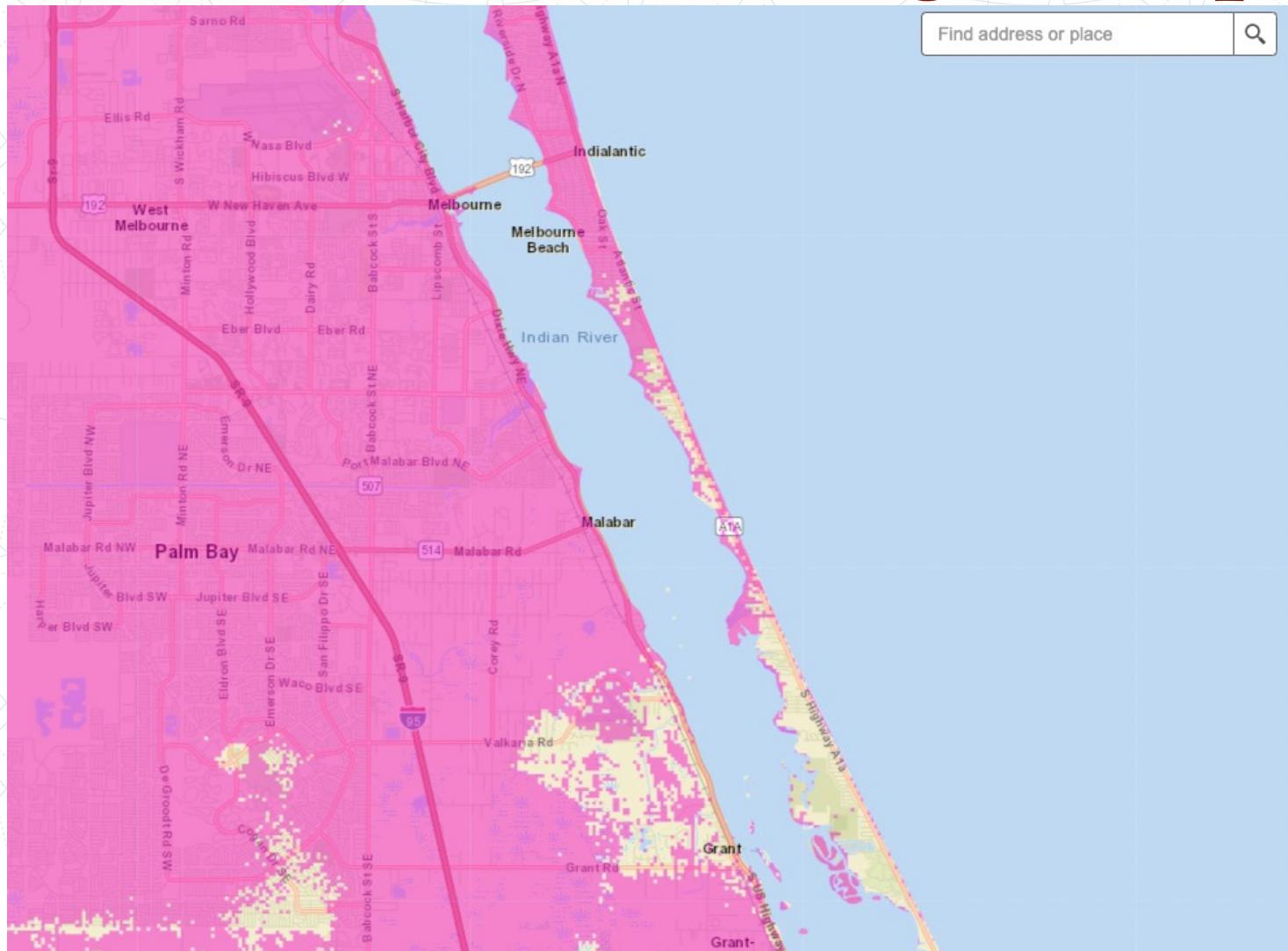


Data download links:

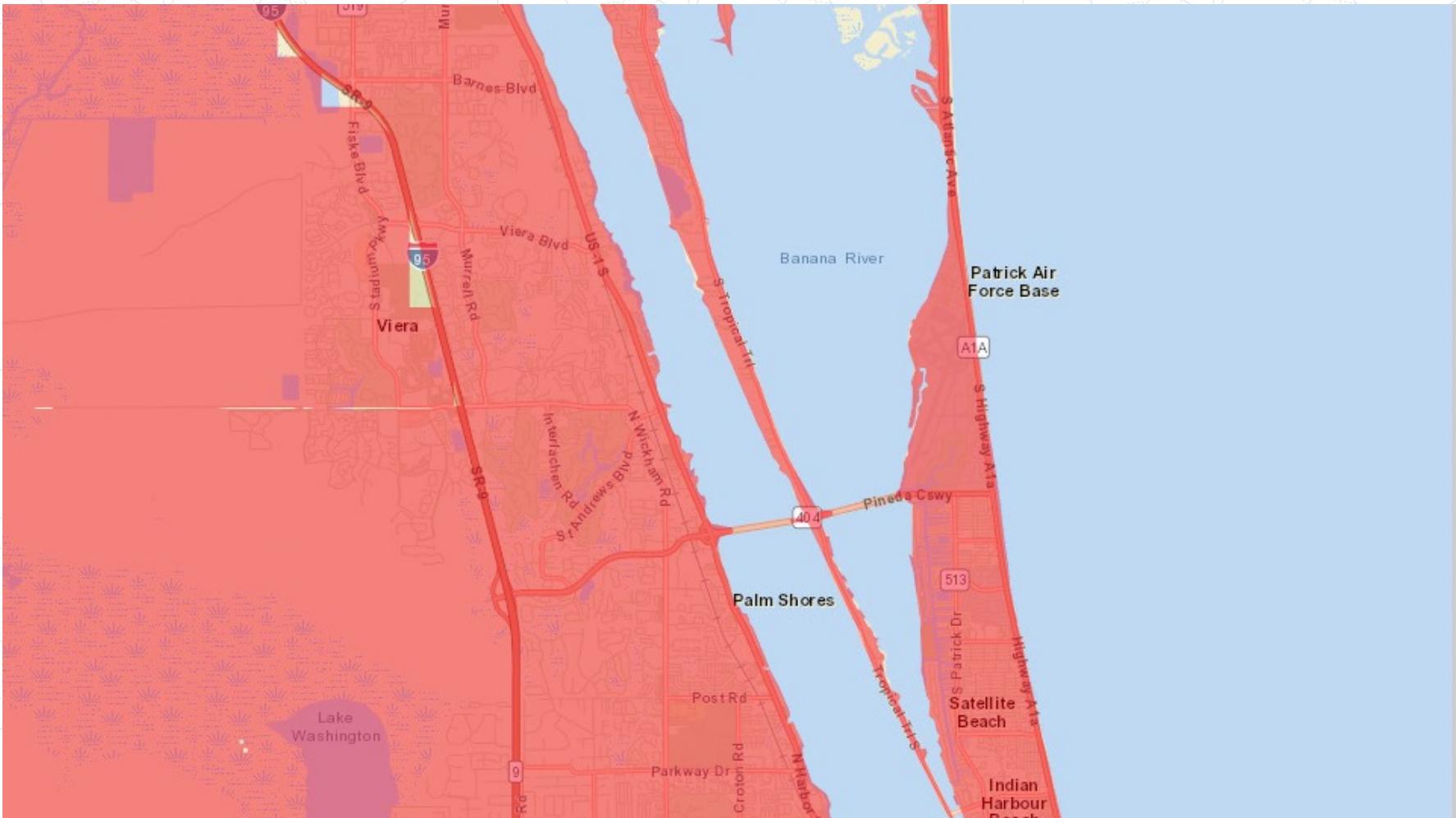
[Download Broadband Mapping Files \(Shapefiles\)](#)

<https://fcc.maps.arcgis.com/apps/webappviewer/index.html?id=6c1b2e73d9d749cdb7bc88a0d1bdd25b>

Network Coverage Map: T-Mobile



Network Coverage Map: Verizon



- ▶ AT&T Mobility LTE Voice
- ▶ T-Mobile LTE Data
- ▶ T-Mobile LTE Voice
- ▶ UScellular LTE Data
- ▶ UScellular LTE Voice
- ▶ Verizon LTE Data
- ▶ Verizon LTE Voice

Femtocell

- Extend the carrier network, leveraging the consumer's broadband connection for uplink connectivity
- Femtocell devices (e.g., Home NodeB or HNB) allow consumers to establish a relatively short-range extension of the carrier network that provides similar connectivity services (e.g., voice, data, SMS / MMS)
 - Also offers attackers new opportunities to attack the carrier infrastructure, as well as User Equipment devices

Femtocell

- HNB devices use IPsec to connect to the carrier network and provides strong confidentiality and integrity support over the untrusted broadband connection
- UE is responsible for encrypting/decrypting the 3G voice, data, and messaging services locally before forwarding to the UE or to the carrier over IPsec
 - The opportunity to mount attacks against unsuspecting UE devices

Femtocell Attack

- HNB is authorized device on the carrier network and has access to dynamic key information used to encrypt/decrypt the 3G connection
 - MiTM to manipulate and intercept phone calls
- They found a way to have root access to femtocell device and run their codes in them to sniff the traffic
 - Presented at Defcon 21

<https://www.youtube.com/watch?v=gfcq8clu1RI>

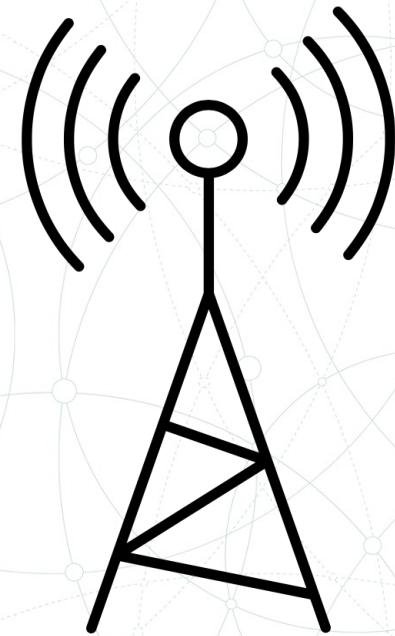
WiFi Based IMSI Catcher

- Features
 - Tracking: IMSI, Location
- Operates in unlicensed ISM Bands: WiFi
 - Fake Access Points
 - Redirect/Spoofs mobile packet data gateway
 - Exploits protocol & configuration weaknesses
- Based on two separate techniques [3GPP TS33.234]
 - WiFi Network Authentication ('WLAN direct IP access')
 - WiFi-Calling Authentication ('WLAN 3GPP IP access')

<https://www.youtube.com/watch?v=7ZBDfxSdnD4>

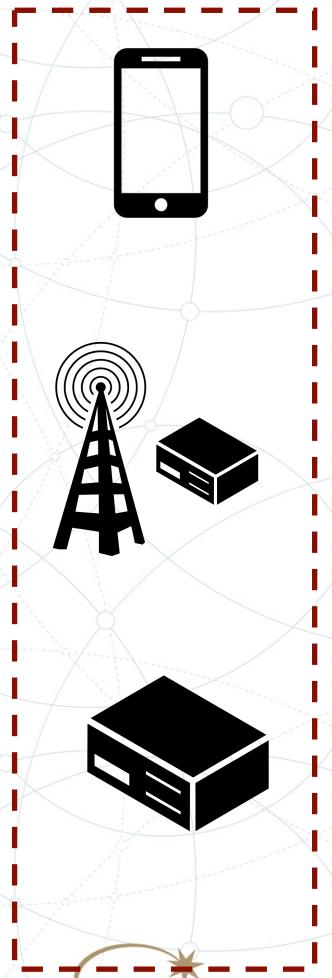
4G / LTE

- Long-Term Evaluation (LTE) Protocol
- Predecessor to GSM
- Marketed as 4G LTE or Advanced 4G
- In addition to higher speeds
 - Offers improvements for privacy
 - Introduces new encryption schemes



LTE Network Elements

- Universal Subscriber Identity Mobile (USIM):
 - An application that resides within the mobile devices;
 - Implements mutual authentication, encryption, and the address book functionality
- Evolved Node B (eNodeB):
 - Provides the radio element access mechanism for the network
- Mobile Management Entity (MME):
 - Key-control node for LTE access network;
 - Responsible for encryption/decryption of network traffic after mutual authentication and key setup/exchange

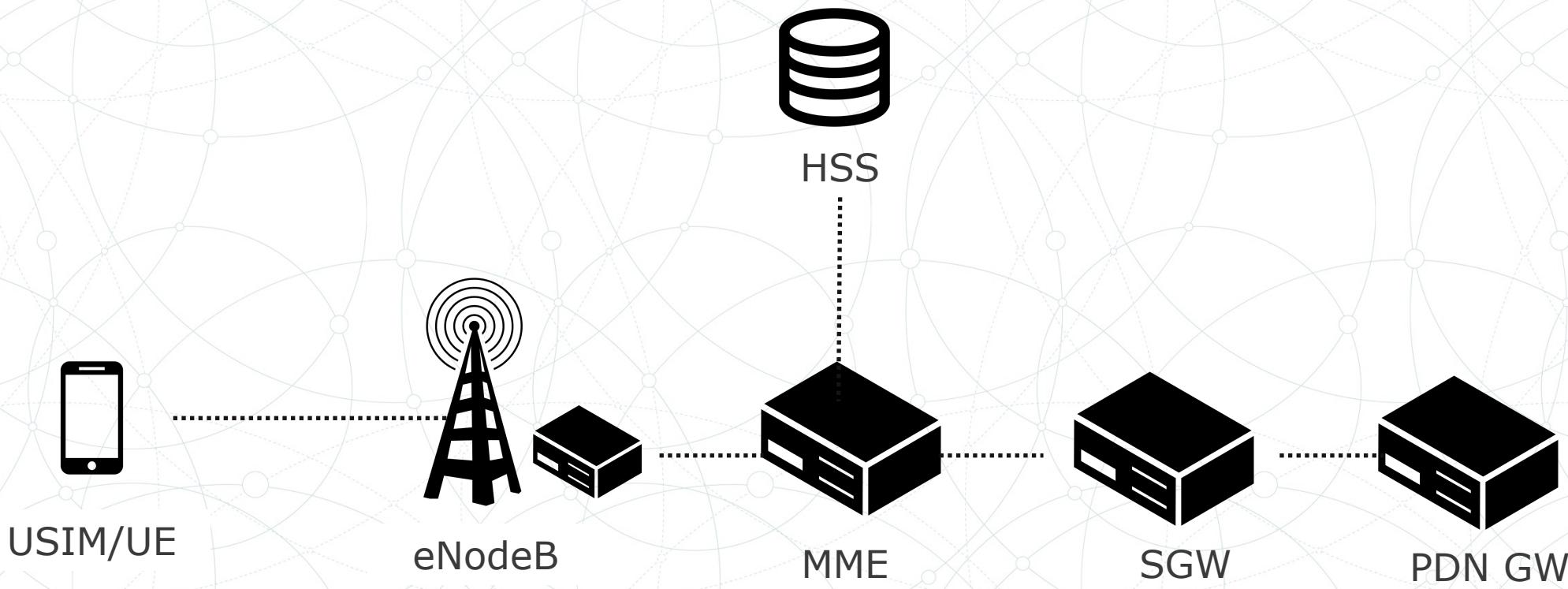


LTE Network Elements

- Home Subscriber Server (HSS):
 - Subscriber database that provides MME with records to establish mutual authentication between USIM and MME
- Serving Gateway (SGW):
 - Establishes routing and packet forwarding within network
 - Interacts with MME to grant/deny access to the UE
- Packet Data Network Gateway (PDN-GW):
 - Provides connectivity to external packet networks

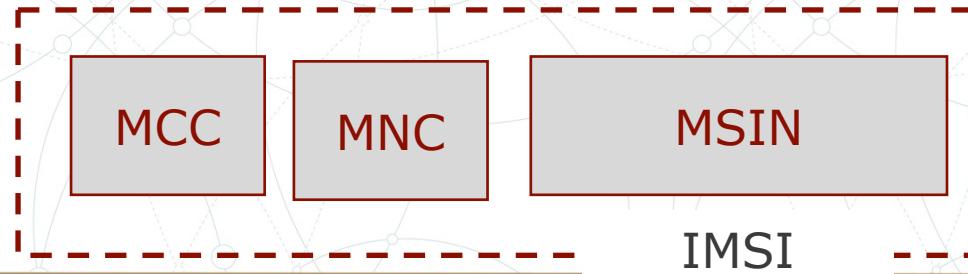


LTE Network Model



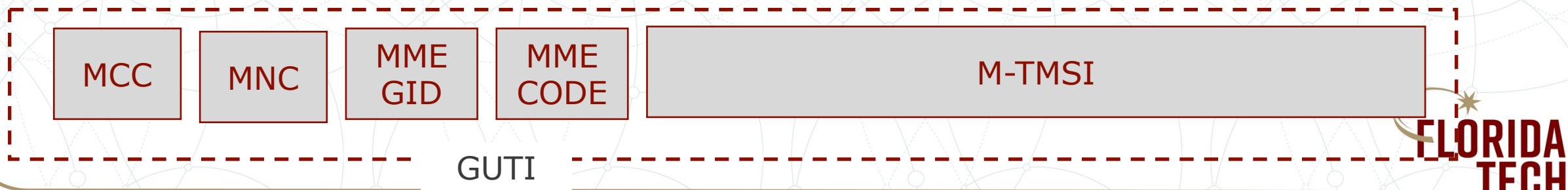
LTE Addressing Scheme for IMSI

- IMSI is made up of three components:
 - MCC: Mobile Country Code, identifying the country of the end-user
 - MNC: Mobile Network Code, identifying the home network
 - MSIN: Mobile Subscriber Identification Number, identifying the user within MCC and MNC context



LTE Addressing Scheme for IMSI

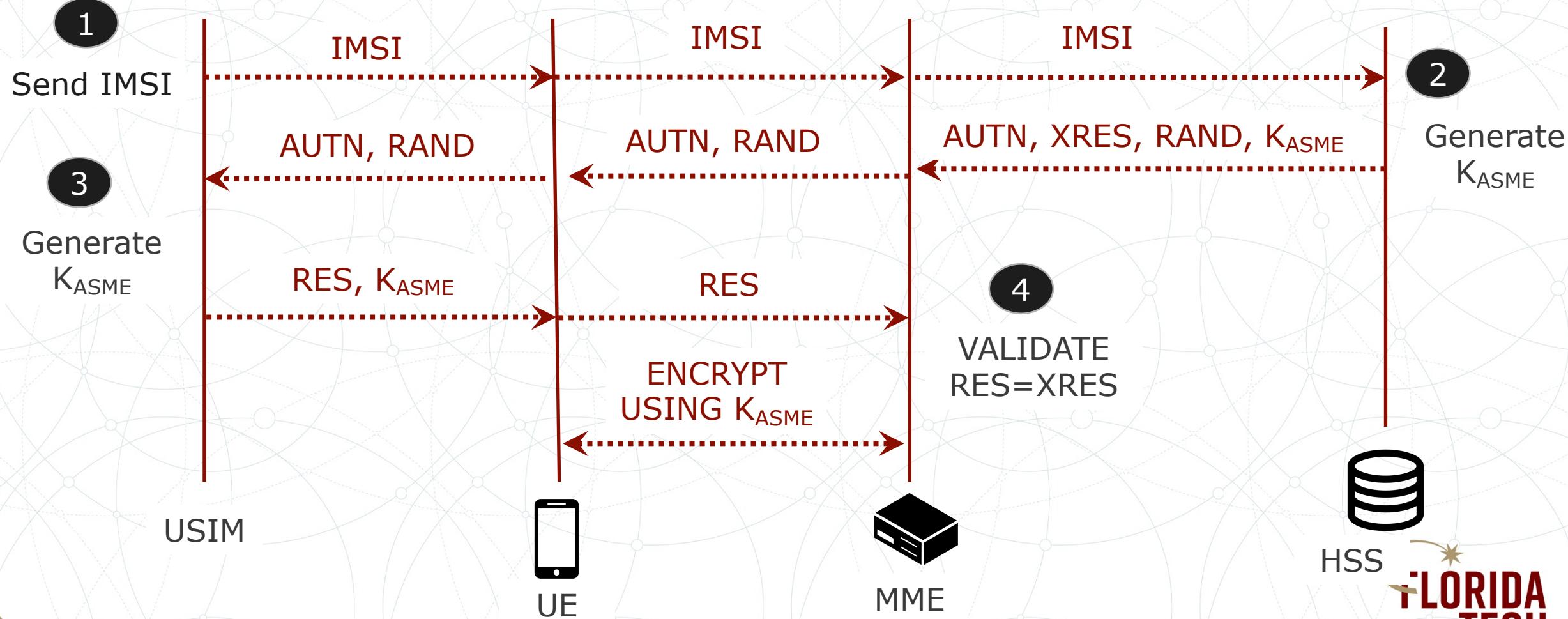
- IMSI value is stored on the USIM and a fixed value
 - Acts as a shared identifier for UE (e.g., LTE phone) and the HSS for the associated authentication key "K"
- To ensure privacy of the unique handset (such as identifying an IMSI to an individual), LTE introduces a Globally Unique Temporary ID (GUTI) which consists of the MCC, MNC MME info and the Temporary Mobile Subscriber ID (TMSI)



LTE Authentication

- Mutual auth of the handset and the network infrastructure through the Evolved Packet System Authentication and Key Agreement (EPS-AKA)
- Similar to GSM/3G, authentication in LTE relies on the identification function and shared key content provided by the IMSI (International Mobile Subscriber Identity)

LTE Authentication



LTE Authentication Steps

- 1. USIM shares IMSI with the UE
 - USIM never discloses the secret key K to the UE or over any network interface
- 2. UE forwards IMSI to the MME
- 3. MME forwards IMSI to the HSS
 - With IMSI, HSS can identify the secret key K (that is never shared with MME)
 - With secret key K, HSS selects a random value (RAND) and derives the Access Secure Management Entity Key (K_{ASME}), an authentication value (AUTN), and the Expected Response (XRES) values

LTE Authentication Steps

- 4. HHS shares K_{ASME} , AUTN, XRES, and RAND values with the MME
 - HHS is finished with the exchange at this point, leaving identity validation to the MME
- 5. MME retains the K_{ASME} and XRES values as local secrets, sharing the AUTN and RAND values with the UE
- 6. UE shares the AUTN and RAND with the USIM

LTE Authentication Steps

- 7. The USIM, who, like the HHS, knows the secret key K, calculates its own AUTN value, comparing it to that of AUTN originally from the HSS
 - If the AUTN values match, the USIM has validated the identity of the HSS as having the same shared key K
 - Next, USIM calculated its own response value (RES) and intermediate key values ultimately used to derive the K_{ASME} sent to the UE
- 8. The UE saves the K_{ASME} for later use, forwarding the RES value to the MME

LTE Authentication Steps

- 9. The MME compares the RES to the XRES previously delivered from the HSS
 - By comparing them, MME validates that the USIM has the correct secret key K
 - Mutually authenticated
- 10. Using the derived K_{ASME} values, UE and MME can encrypt and decrypt traffic over the wireless medium

LTE Authentication Vulnerability

- The IMSI is sent in plaintext
 - Rogue LTE network can get IMSI
 - Privacy threat to IMSI
- Yet, the secret K never is disclosed to the UE from USIM, preventing rogue applications from stealing the value and limiting attacker's ability to clone the value onto another USIM

LTE Encryption

- LTE supports algorithm flexibility
- 3GPP systems were limited to a handful algorithms and these could not be replaced without changes to the network infrastructure
- Yet, LTE networks could adapt to new algorithm option to mitigate any flow
 - Let's say there is a flaw found in AES

LTE Supported Encryption Algorithms

- NULL Algorithm:
 - Does not provide confidentiality of network traffic
 - In some cases, need to provide service outweighs the desire for security in LTE
 - Provides network access for devices lacking USIM card for situations such as emergency services (e.g., 911 in US)
 - May create opportunity for attacker to impersonate a legitimate carrier network without the need for cryptographic attacks

LTE Supported Encryption Algorithms

- The Kasumi Algorithm:
 - The first ciphering algorithm for the LTE standard, the Kasumi algorithm, is mainly a block cipher algorithm that uses a key size of 128 bits
 - The algorithm utilizes two mapping functions to produce the ciphertext, which are called S-boxes
 - Kasumi was specifically designed as a building block for the UMTS encryption algorithms (UEA1) and integrity algorithms (UIA1)

LTE Supported Encryption Algorithms

- SNOW 3G (128-EEA1):
 - Word-based synchronous stream ciphers implemented with a LFSR and Finite State Machine
 - Brought forward from 3G networks and reintroduced as a well-known option for carriers that have used the algorithm for many years prior
 - Helped LTE by reusing an algorithm well understood and readily available

LTE Supported Encryption Algorithms

- Milenage:
 - AES-128-bit Based algorithm in CTR (Counter) mode
 - AES encryption can be accelerated in hardware using parallelism and has already been proven in other well-known deployment scenarios (e.g., 802.11/WPA2 security)

LTE Supported Encryption Algorithms

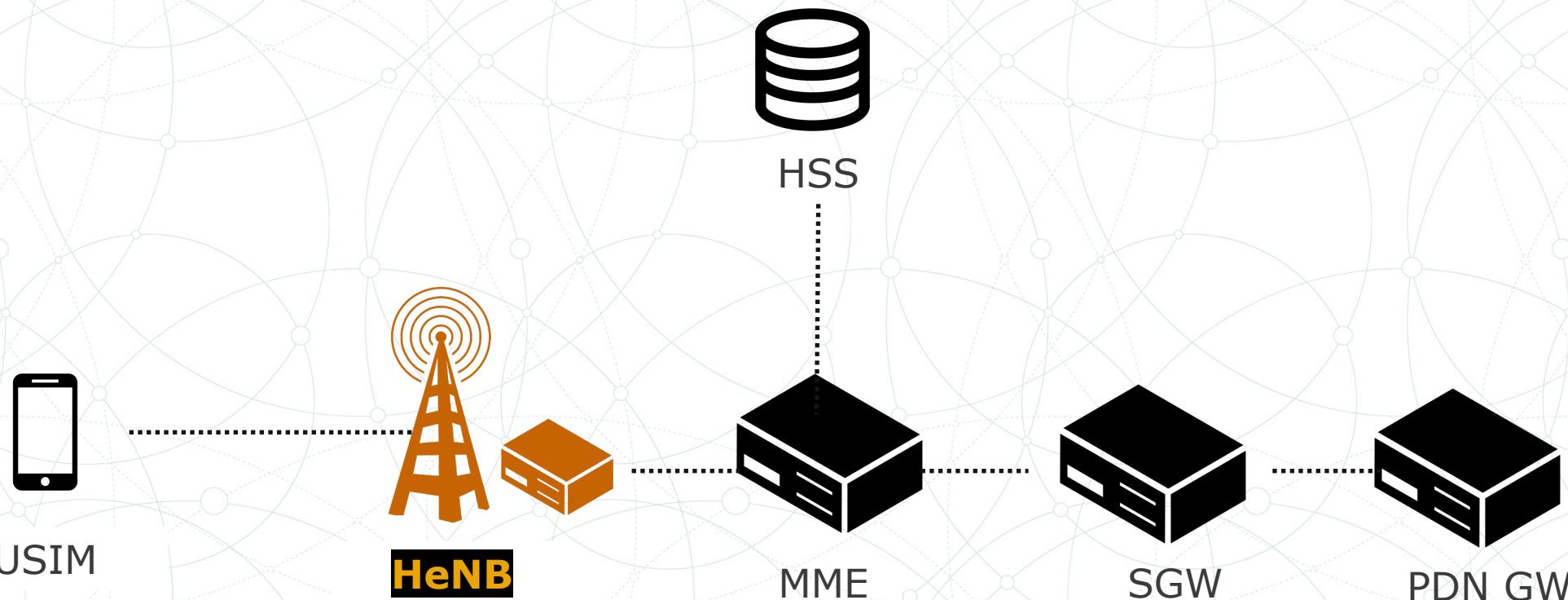
- ZUC:
 - Cryptographic algorithm for LTE
 - Combines block + steam cipher approaches, using:
 - Non-Linear Function (e.g., S-Boxes)
 - Linear Feedback Shift Register (LFSR)
 - Uses Bit Reorganization (BR)
 - Strong resistance to algebraic attacks

```
void GenerateKeystream(u32*  
pKeystream, int KeystreamLen)  
{  
int i; {  
BitReorganization();  
F(); /* discard the output of F */  
LFSRWithWorkMode();  
}  
for (i = 0; i < KeystreamLen; i ++)  
{  
BitReorganization();  
pKeystream[i] = F() ^ BRC_X3;  
LFSRWithWorkMode();  
}  
}
```

LTE Encryption Algorithm Tradeoffs

| Scheme | Advantages | Disadvantages |
|----------|---|---|
| Kasumi | Offers strong encryption via 128-bit keys Optimized for hardware implementation Offers resistance to block cipher attacks | Vulnerable to algebraic attacks |
| SNOW 3G | Fits 3G security requirements Offers protection against algebraic attacks | Computationally complicated |
| Milenage | Fits 3G security requirements Offers strong encryption via 128-bit keys Protects against side-channel attacks | Does not require standard algorithm Some interoperability issues |
| ZUC | Fits 3G security requirements Offers strong encryption via 128-bit keys Built on sound design principles | Still under scrutiny |

Home eNodeB: AKA Femtocell

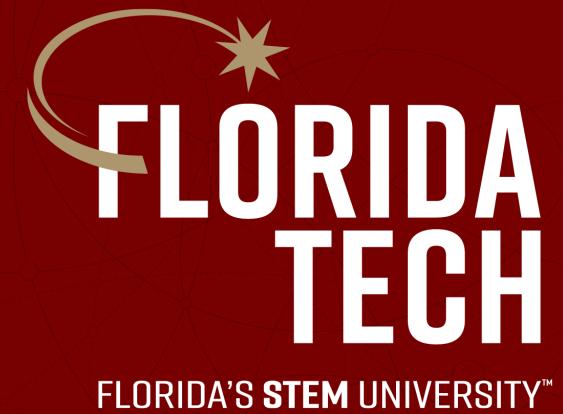


HeNB Device Requirements

- Physical security requirements
- Root of trust and Trusted Execution Environment:
 - Utilize root of trust that is subsequently used to verify the Trusted Execution Environment (TEE)
 - Through this mechanism, all code must pass signature validation tests based on the root of trust to thwart malicious code attacks
 - Specifically, TEE must extend the boot process and all OS and other executables used on the HeNB

HeNB Device Requirements

- Device and data integrity check:
 - Must provide these check functionalities to identify tampering attacks that could threaten the security of the HeNB, user data, and the carrier network
- Geolocation:
 - To make sure it is using the frequency that it is allowed to in a given location that is obtained by GPS receiver
- Time synchronization
 - Maintain an accurate clock system to ensure validity of certificate expiration used by IPsec



Thank you. Questions?

Dr. Abdullah Aydeger

LoFin: LoRa-based UAV Fingerprinting Framework



Maryna Veksler¹, David Langus Rodríguez², Ahmet Aris², Kemal Akkaya¹, A. Selcuk Uluagac²

¹Advanced Wireless and Security Lab., ²Cyber-Physical Systems Security Lab.

Florida International University

{mveks001, dlang029, aaris, suluagac, kakkaya} @fiu.edu

Overview

- Introduction
- Related Work
- Background
- LoFin Architecture
- Performance Evaluation
- Conclusions & Future Work



Introduction

- Unmanned Aerial Vehicles (UAVs) have become a world phenomenon:
 - ❖ Used for surveillance, reconnaissance, search & rescue missions
 - Intelligent decision making, mobility, and sensing capabilities
- UAV applications often involve long distance communications with the controller (GCS):
 - ❖ Reliable network channel for security and large transmission radius
- Long Range (LoRa) communication protocol:
 - ❖ Long-range and low-power technology
 - ❖ Consistent coverage across urban and rural areas



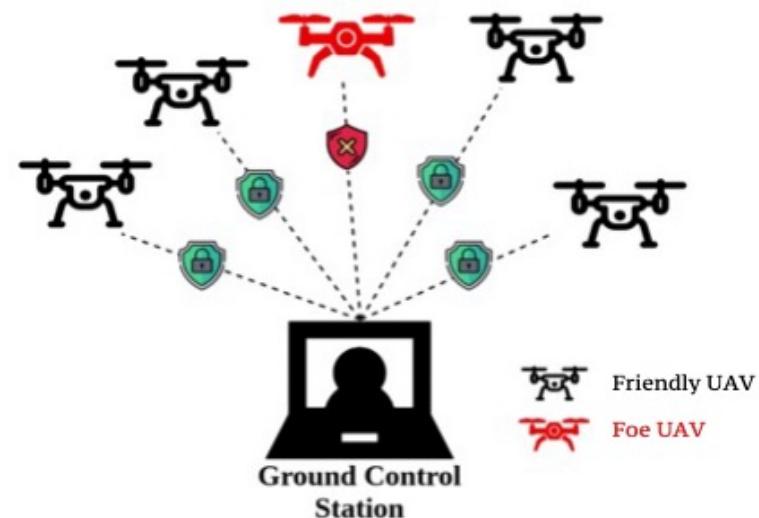
Motivation

❑ UAV applications require strong security:

- ❖ Impersonation attacks
- ❖ Attack may steal sensitive information
- ❖ Mission infiltration
- ❖ Communication disruption

❑ Intrusion Detection system is required:

- ❖ No sniffing tools developed for UAV communicating via LoRa
- ❖ Radio frequency (RF) analysis not effective due to channel interference and reconfiguration of LoRa channel



Related Work

□ Device Fingerprinting is an effective technique to detect the device impersonation attack

❖ LoRa fingerprinting:

- Analysis of physical (PHY) layer radio frequency (RF) using deep learning (DL) [1, 2]
- Challenges: varying configurations of LoRa protocol

❖ UAV fingerprinting:

- RF statistical analysis [3,4]
- Challenges: affected by signal-to-noise (SNR) ratio

□ Our Contribution → LoFin:

❖ Do not require analysis of physical layer communications:

- Resistant to changes in (1) RF signals due to external factors, (2) LoRa parameters

❖ Passive fingerprinting:

- No processing overhead

❖ Encryption immunity with preserved data security



Background

□ LoRa Stack



- ❖ LoRa physical layer – chirp spread spectrum (CSS) radio frequency modulation system
- ❖ LoRaWAN communication protocol and network architecture of upper layers
- ❖ LoRa provides flexibility to configure multiple transmission parameters for improved data rate:
 - Spreading factor (SF), bandwidth (BW), coding rate (CR) and carrier frequency (CF)
 - Adjusted based on the payload size and transmission range
- ❖ LoRaWAN provides two methods for device activation:
 - Over-the-air-activation (OTAA) provides dynamic assignment of device addresses and session keys.
 - Activation-by-Personalization (ABP) requires hardcoding of device addresses and session keys

□ LoRa-based UAV Communications:

- ❖ Real-time quality monitoring system [5]
- ❖ Marine coastal environment monitoring [6]
- ❖ UAVs as end-node and gateway devices in LoRa network [7]



Threat Model & Assumptions

□ Assumptions:

- ❖ Network consists of multiple LoRa-based UAVs and sensors
- ❖ LoFIN set up on the centralized server to passively capture network traffic

□ Attacks & scenarios considered:

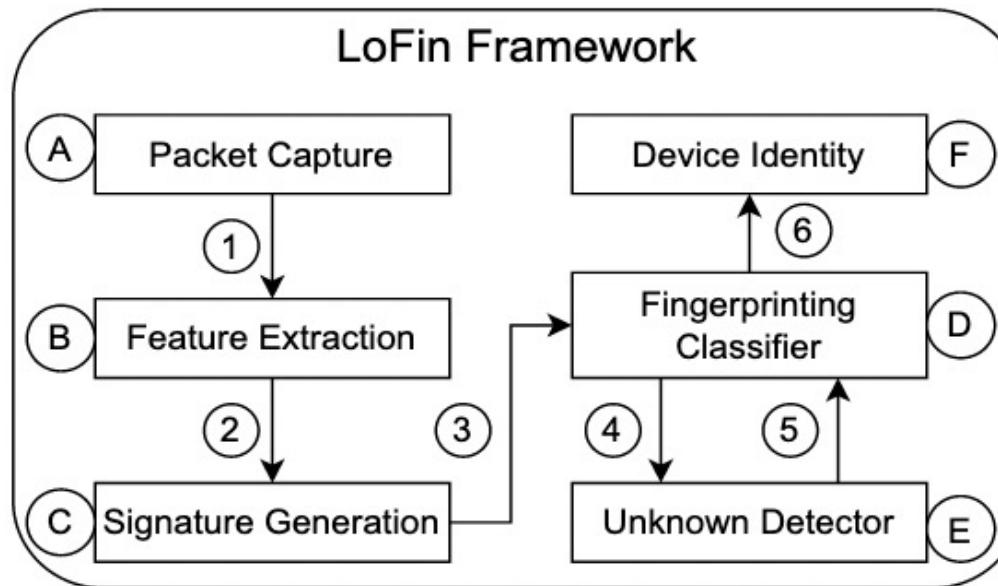
- ❖ Passive Impersonation:
 - Network infiltration to mimic the behavior of legitimate UAV
- ❖ Active attack
 - Disrupt functionality of the device identification tool and network communications
- ❖ LoRa Configurations:
 - Periodic changes of the protocol configuration may impact accuracy of proposed detection mechanism



LoFin Architecture - Overview

□ Overview:

- ❖ Passively collects network traffic
- ❖ Extract device-specific features
- ❖ Generate device signature
- ❖ Classification process & detection of unknown device signatures
- ❖ Devices identify – a foe or a friend



LoFin Architecture - Components

□ 4 major components:

- ❖ *feature extraction, signature generation, classifier, and unknown detector*

□ Feature Extraction:

❖ Considered features:

- Payload length, device address, SNR, and combination of LoRa configuration parameters

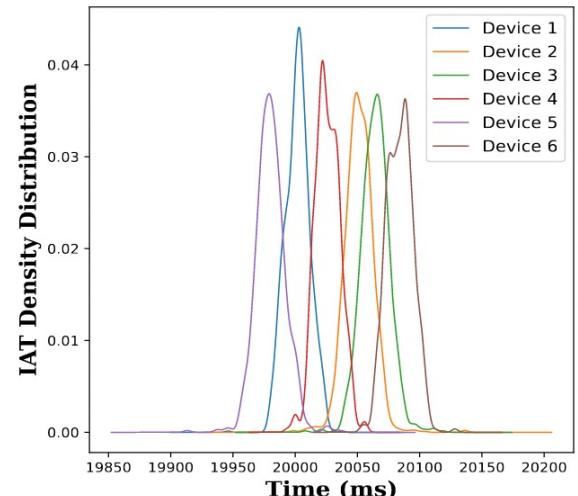
❖ Selected inter-arrival time (IAT) for device signature generation

- Manufacturing defects introduce unique noises to the data transmission process, resulting in fixed time overhead

❖ IAT for LoRa is affected by Time on Air (ToA):

- Represent the time it takes a signal to travel from sender to receiver
- ToA is directly affected by spreading factor (SF) and bandwidth (BW) LoRa parameters

$$\text{❖ } \text{IAT} = (t_i - t_{i-1}) - \text{ToA}_i$$



LoFIN Architecture – Components contd.

❑ Signature Generation

❖ Time-series extraction

- Extract time-series of length l for each set of IAT values
- Extracted features split into N shorter series

$$\text{time_series} = [IAT_1, IAT_1, IAT_2, \dots, IAT_n].$$

❖ Statistical Analysis

- Using tsfresh obtained 816 distinctive features
- Selected 367 significant features based on p-score and relevance table

❑ Fingerprinting Classifier

- ❖ Machine Learning classifiers: KNN, RF, GaussianNB, and SVM)
- ❖ Produces probability vector, V_p indicating similarity value for known device
- ❖ V_p is passed to ***unknown detector***, to filter out unknown devices:
 - Then classification results for known devices are interpreted to determine device identity



LoFIN Architecture – Components contd.

□ Unknown Detector

- ❖ Responsible for identifying potentially adversarial devices within LoRa network
- ❖ Intermediate stage of ***fingerprinting classifier***
 - Applies probability measures to filter out suspicious devices
 - Threshold approach to spot samples yielding low closeness in regard to known devices

□ Effectiveness Metrics

- ❖ Accuracy (ACC) to measure overall system performance

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

- ❖ Precision & Recall

- Accuracy for specific device class and indication of the number of mishits for a given class

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$



Performance Evaluation - Testbed

□ Devices Summary

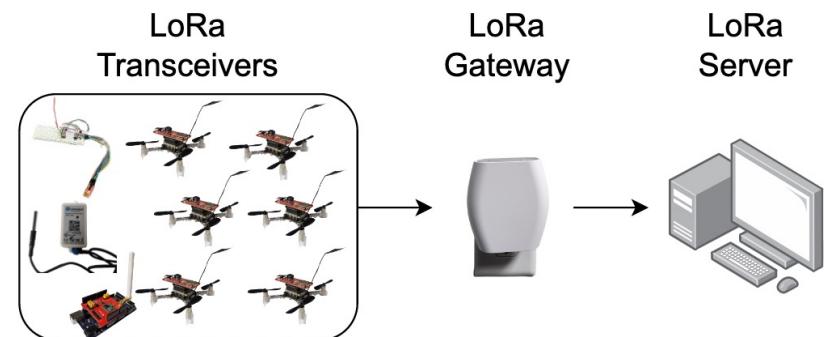
| Device | Device Type | Quantity |
|-----------------------------------|--------------------|----------|
| CrazyFlee with Arduino expLoRaBLE | Drone Telemetry | 6 |
| Arduino Mega | LoRa Transceiver 1 | 1 |
| Arduino Mini | LoRa Transceiver 2 | 1 |
| Dragino LHT65 | Temperature Sensor | 1 |

□ Testbed Setup:

- ❖ CrazyFlie drones programmed using Bitcraze library [8]
- ❖ Arduino expLoRaBLE as telemetry
- ❖ MAVLink protocol

□ Data Collected:

- ❖ Total of $\cong 24,000$ network packets
- ❖ 1,200 packets for each device
- ❖ Additional data collected to implement adversarial random delay scenario
- ❖ 70% of data used to train LoFin classifier



Performance Evaluation - Experiments

❑ Devices setup:

- ❖ Drone Telemetry devices configured for a synchronized mission of data collection
- ❖ Rest of the devices transmit independently

❑ LoFin Configurations:

- ❖ Extracted 40 time series vectors consisting of 30 consecutive IAT values for each device
- ❖ Selected optimal ML algorithm for fingerprinting classifier
 - RF with 10 trees, *entropy function* as quality measure of a split, and random split of 42
 - KNN with “ball_tree” algorithm for nearest neighbors computation
- ❖ 5 K-Fold cross validations
 - Optimal model fitting



Performance Evaluation - Experiments

❑ Experiment 1 – Isolated Environment

- ❖ All devices are known and isolated from the adversary
- ❖ Accuracy: **100%** for RF classifier and **99.2%** for KNN

❑ Experiment 2 – Different Configuration Scenarios (Table 1)

- ❖ LoFin is trained for devices transmitting using **SF7**
- ❖ Set LoRa SF configuration to **SF10** for 4 devices and apply LoFin framework

| Device | RF | | KNN | |
|-------------------------|-----------|--------|-----------|--------|
| | Precision | Recall | Precision | Recall |
| Drone Telemetry 1* | 1.0 | 1.0 | 1.0 | 1.0 |
| Drone Telemetry 2* | 1.0 | 1.0 | 1.0 | 1.0 |
| Drone Telemetry 3 | 1.0 | 1.0 | 0.89 | 1.0 |
| Drone Telemetry 4 | 1.0 | 1.0 | 0.88 | 1.0 |
| Drone Telemetry 5 | 1.0 | 1.0 | 1.0 | 1.0 |
| Drone Telemetry 6* | 1.0 | 1.0 | 1.0 | 1.0 |
| LoRa Transceiver 1* | 1.0 | 1.0 | 1.0 | 0.67 |
| LoRa Transceiver 2 | 1.0 | 1.0 | 1.0 | 1.0 |
| Temperature Sensor | 1.0 | 1.0 | 1.0 | 1.0 |
| Average Accuracy | 1.0 | | 0.972 | |

Table 1. Results for experiment 2

Performance Evaluation - Experiments

☐ Experiment 3 – Impersonation Attack (Table 2)

- ❖ 6 identical drone telemetry devices;
 - 1 is selected to represent an **adversary**
- ❖ Evaluated unknown detector ability to detect foe's traffic
- ❖ ***Unknown detector*** cannot be used with KNN:
 - Requires probability vector
- ❖ **100%** classification and true negative rate;
 - False negative rate **below 10%**

| Devices | Precision | Recall |
|----------------------------|-----------|--------|
| Drone Telemetry 1 | 1.0 | 1.0 |
| Drone Telemetry 2 | 1.0 | 1.0 |
| Drone Telemetry 3 | 1.0 | 1.0 |
| Drone Telemetry 4 | 1.0 | 1.0 |
| Drone Telemetry 5 | 1.0 | 1.0 |
| Drone Telemetry 6* | N/A | N/A |
| Average Accuracy | 1.0 | |
| True Negative Rate | 1.0 | |
| False Negative Rate | 0.082 | |

Table 2. Results for experiment 3

Performance Evaluation - Experiments

□ Experiment 4 – Random Delay Attack (Table 3)

❖ Implemented artificial delay to 1 drone telemetry device

- Used built-in probabilistic random() library to design random delay algorithm

❖ Overall accuracy is **95%** for RF-based fingerprinting classifier

❖ Unknown detector trade-off

- Benign traffic may be marked as potentially adversarial

```
p=0; d=0, counter=0;  
while(True):  
    select p from [30,70] and d from [1,5]  
    if counter == p:  
        wait d and transmit  
        p=0; d=0, counter=0;  
    else:  
        transmit  
        counter += 1
```

| Device | RF | | KNN | |
|--------------------|-----------|--------|-----------|--------|
| | Precision | Recall | Precision | Recall |
| Drone Telemetry 1* | 1.0 | 0.86 | 1.0 | 0.57 |
| Drone Telemetry 2 | 1.0 | 1.0 | 1.0 | 1.0 |
| Drone Telemetry 3 | 1.0 | 1.0 | 1.0 | 1.0 |
| Drone Telemetry 4 | 1.0 | 1.0 | 0.86 | 1.0 |
| Drone Telemetry 5 | 1.0 | 1.0 | 1.0 | 1.0 |
| Drone Telemetry 6 | 0.92 | 1.0 | 0.91 | 0.91 |
| LoRa Transceiver 1 | 1.0 | 1.0 | 1.0 | 0.67 |
| LoRa Transceiver 2 | 1.0 | 1.0 | 1.0 | 1.0 |
| Temperature Sensor | 1.0 | 1.0 | 1.0 | 1.0 |
| Average Accuracy | 0.991 | | 0.964 | |

Table 3. Results for experiment 4



Conclusion & Future Work

□ LoFin:

- ❖ 1st framework to passively fingerprint LoRa devices using MAC layer information
- ❖ 100% precision and recall for majority of the scenarios
- ❖ Resistant to changes in LoRa configurations
- ❖ 100% detection of unknown devices with false-negative rate below 10%
- ❖ Not influenced by changes in PHY layer

□ Future Work:

- ❖ Increase number of devices and its diversity
- ❖ Improve LoFin robustness against random delay attack



Acknowledgements

This work is partially supported by the US National Science Foundation Awards: NSF-CAREER-CNS-1453647, NSF-1718116, NSF-1663051, Microsoft, and US Army Research Office (Grant Number W911NF-21-1-0264). The views expressed are those of the authors only, not of the funding agencies.



References

1. G. Shen, J. Zhang, A. Marshall, L. Peng, and X. Wang, "Radio frequency fingerprint identification for lora using spectrogram and cnn," in IEEE INFOCOM 2021 - IEEE Conference on Computer Communications.
2. A. Elmaghbub and B. Hamdaoui, "Lora device fingerprinting in the wild: Disclosing rf data-driven fingerprint sensitivity to deployment variability," IEEE Access, vol. 9, pp. 142 893–142 909, 2021.
3. N. Soltani, G. Reus-Muns, B. Salehi, J. Dy, S. Ioannidis, and K. Chowdhury, "Rf fingerprinting unmanned aerial vehicles with non-standard transmitter waveforms," IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 15 518–15 531, 2020.
4. M. Ezuma, F. Erden, C. K. Anjinappa, O. Ozdemir, and I. Guvenc, "Micro-uav detection and classification from rf fingerprints using machine learning techniques," in 2019 IEEE Aerospace Conference.
5. A. Rahmadhani, Richard, R. Isswandhana, A. Giovani, and R. A. Syah, "Lorawan as secondary telemetry communication system for drone delivery," in 2018 IEEE International Conference on Internet of Things and Intelligence System (IOT AIS), pp. 116–122.
6. C. A. Trasviña-Moreno, R. Blasco, Marco, R. Casas, and A. Trasviña-Castro, "Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring," Sensors, vol. 17, no. 3, 2017.
7. M. H. M. Ghazali, K. Teoh, and W. Rahiman, "A systematic review of real-time deployments of uav-based lora communication network," IEEE Access, vol. 9, pp. 124 817–124 830, 2021.
8. "Bitcraze: Documentation," <https://www.bitcraze.io/documentation/start/>.



Thank you!

Q&A



Department of Computer Science



CSE 4820: Wireless and Mobile Security

23. Final Review

Dr. Abdullah Aydeger

Location: Harris Inst #310

Email: aaydeger@fit.edu

Outline

SDR

Zigbee

Z-Wave

LoRaWAN

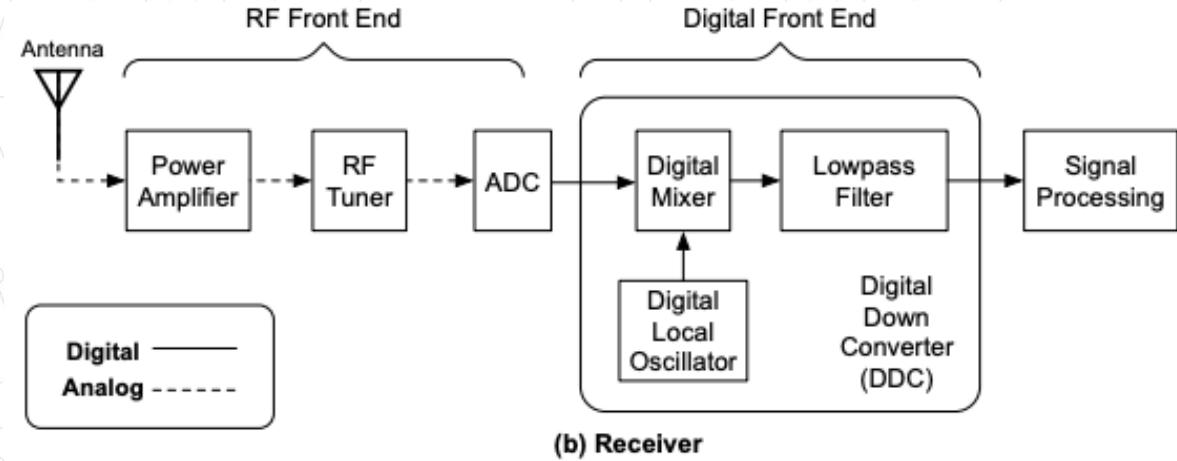
Cellular Networks (2G / 3G / LTE)

Software Defined Radios

- The process of reverse engineering a radio signal to understand elements such as frequency, modulation, encoding, and protocol in order to intercept or replicate the signal
 - For ex. replace sound waves with radio waves
 - By using special ‘radio soundcard’, you can receive and transmit arbitrary signals
- Has redefined wireless hacking
 - Instead of being limited by black box radios, unfettered access to RF
 - Access to Radio modules/protocols that were obscured

SDR Architecture

- Three main parts:
 - Radio Frequency (RF) amplifier, the tuner, and the Analog-to-Digital Converter (ADC)
 - RF amplifier is responsible for boosting weak signals
 - Tuner; select portion of radio spectrum to analyze
 - Like tuning on an old radio
 - ADC; sampling (analog waveform to stream of digital numbers)
 - Antennas; isotropic (any direction) or directional



RTL-SDR

- Digital Video Tuner converted into SDR
- Receiver only
- Frequency Range: 50MHz to 1.7GHz
- ADC: 8 bits
- Popular among hobbyist due to low cost
- RTL-SDR Source block available in Gnuradio

<https://amzn.to/321mYwB>



RTL-SDR Source

Sync: Unknown PPS
Number Channels: 1
Sample Rate (sps): 32k
Ch0: Frequency (Hz): 100M
Ch0: Frequency Correction (ppm): 0
Ch0: DC Offset Mode: 0
Ch0: IQ Balance Mode: 0
Ch0: Gain Mode: False
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

HackRF

- Wide Band Software Defined Radio (SDR)
- Half-duplex transceiver
- Frequency Range: 10MHz to 6GHz, ADC: 8 bits
- Power: 30 mW - 1 mW (depending on band)
- Popular among researchers due to low cost
- Open sourced hardware

<https://greatscottgadgets.com/hackrf/one/>



Software: GNU Radio

- Gnuradio's modular design allows us to process and prepare signals to test methods of intercepting and replicating the signal
- Helpful blocks: hardware sinks / sources, file sink / sources, modulators, signal multipliers, signal sources, vector sources, variables, and QT Gui sinks

Software: Universal Radio Hacker

- Discussed in Pohl, Johannes, and Andreas Noack. "Universal radio hacker: a suite for analyzing and attacking stateful wireless protocols." 12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18). 2018.
- Complete suite for wireless protocol investigations with native support for many common SDRs
- The URH allows easy demodulation of signals combined with an automatic detection of modulation parameters to identify the bits and bytes that fly over the air

<https://github.com/jopohl/urh>

How to try this in wild?

- Finding a target
 - For ex., key fob, garage opener, wireless mouse, and RC car
- Device reconnaissance;
 - Figure out what frequency it uses
- Finding and capturing signal
- Replaying/ changing



Device reconnaissance

- All RF devices subject to FCC (Federal Communications Commission) Certificate require registration with the FCC
- They are given an FCC ID
 - Usually it is printed somewhere on the device
- Looking up the FCC ID yields information about the RF signal to include the frequency

2 results were found that match the search criteria:

Grantee Code: **B8Q** Product Code: **ACSCT**

Displaying records 1 through 2 of 2.

| View Form | Display Exhibits | Display Grant | Display Correspondence | Applicant Name | Address | City | State | Country | Zip Code | FCC ID | Application Purpose | Final Action Date | Lower Frequency In MHz | Upper Frequency In MHz |
|---|----------------------------------|---|---|---|-------------------------|----------------------|-----------------------|-------------------------|--------------------------|------------------------|-------------------------------------|-----------------------------------|--|--|
|  | Detail Summary |  |  | The Genie Company a Division of Overhead Door Corporation | 1 Door Drive | Mt. Hope | TX | United States | 44660 | B8QACSCT | Original Equipment | 01/08/1997 | 390.0 | 390.0 |

<https://www.fcc.gov/oet/ea/fccid#:~:text=FCC%20ID%20numbers%20consists%20of,string%20representing%20the%20Grantee%2FApplicant.>

Zigbee Outline

Zigbee

History

Layers

Security

Zigbee

- Set of standards for low-power wireless networking
 - Devices with up to 5 years battery life
- Low data-rate transfers, short-range, persistent-powered network coordinators/routers, and simple protocol stack
- Found in industrial and home applications
 - For ex., home theater remote controls to hospital patient monitoring systems

Zigbee as Wireless Standard

- Strong position to be the wireless tech for IoT
 - Already used in Google Nest Smart Thermostat, Philips Hue led light bulbs, Comcast Xfinity home security router to connect home light switches and other peripherals to public internet
- Why do we need Zigbee?
 - Compared to Wifi and BLE, Zigbee is much simpler protocol with a fully functional stack implemented in 120kb of NVRAM where some vendors claim to make reduced-functionality stacks as small as 40kb

Zigbee as Wireless Standard

- Wireless networks transmit at least 54mbps, BLE 1-3mbps, and Zigbee 20-250kbps
 - Not the right protocol for high-speed data transfers
- Wifi devices; relatively short battery life
 - Bluetooth relatively comparable to Zigbee
- Deployed mostly for the home automation;
 - Connectivity among home control systems such as electrical appliances, lighting controls, home security, etc.

ZigBee Layers

Defined by Zigbee Alliance

Defined in IEEE 802.15.4
(Low-rate wireless personal area network)

Application Layer (APL)

App.
Framework

App Support
(APS)

Zigbee Device
Option
(ZDO)

Network Layer
(NWK)

Medium Access Control Layer
(MAC)

Physical Layer
(PHY)

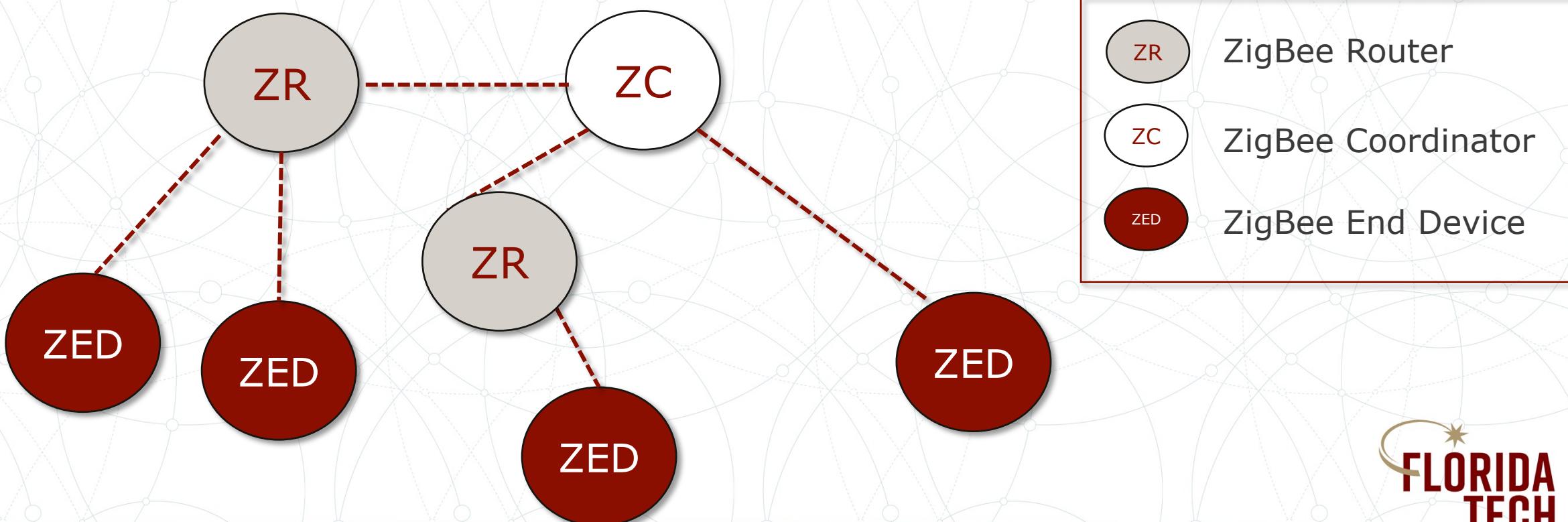
<https://csa-iot.org/all-solutions/zigbee/>

Zigbee: MAC Layer

- Includes functionality needed to build extensive Zigbee networks
 - Including the design of device interconnect topologies, device roles, packet framing, and network association / disassociation
 - Each device has set of capabilities defined by operational roles:
 - Trust Center, Coordinator, Router, and End Device

Example Zigbee Network

- One ZC for the network, additional ZRs



IEEE 802.15.4 MAC Frame Format

| Frame Control | Seq. No. | Dest. Pan ID | Dest. Addr | Source Pan ID | Source Addr | Aux Sec Hdr | Payload | FCS |
|---------------|----------|--------------|------------|---------------|-------------|-------------|---------|-----|
|---------------|----------|--------------|------------|---------------|-------------|-------------|---------|-----|

- Specified in IEEE 802.15.4 Spec
- Frame Control tells what type of frame (beacon, command, data, ack...)
- Sequence Numbers enables in-order delivery
- Dest/Src Pan ID/ Addr handles delivery of frame
- Auxiliary Security Header optionally implements security
- FCS is CRC 16-bit checksum of the mac layer frame

Zigbee MAC Frame Types

- Beacon Frames – used for network discovery; scan the network for ZC or ZR
- Command Frames – same as 802.11 management (association/disassociation)
- Data Frames – same as 802.11 data; exchange data b/w devices
- Acknowledgement Frames – acknowledge frames that were received

ZigBee Network Layer (NWK)

- ZigBee Network Layer (NWK) defined by ZigBee Alliance
 - Responsible for network formation, address allocation, and routing
- Network formation; FFD establishes itself as network coordinator
- Through device discovery, the coordinator must;
 - Select suitable channel to avoid interference
 - Choose random Pan ID that doesn't conflict with nearby Pans
 - Select and issue 16-bit device addresses for devices

ZigBee Application Layer (APL)

- Specifying operation and interface for application objects that define Zigbee device's functionality
- Application objects are developed by Zigbee Alliance as standard functionality profiles, or
 - By manufacturers for proprietary device functionality using the APL as mechanism to communicate with lower layers of Zigbee stack

Zigbee Security

- AES encryption, device and data authentication using a network key
- Two operational modes:
 - Standard; TC authorizes devices through the use of ACL (Access control list), each device uses a single shared key
 - High Security; TC keeps track of all encryption and authentication keys used on the network, enforcing policies for network authentication and key updates
 - If TC fails, no device will be permitted to join the network

Zigbee Encryption

- Uses 128-bit AES
 - Assumed to be strong security
 - Could be leveraging AES in an insecure manner
 - How?
- Three types of keys to manage security;
 - Master key, network key, and link key

Zigbee Keys

- Master key; optional except the Zigbee Pro stack
 - Used in conjunction with Zigbee symmetric key-key establishment (SKKE) process to derive other keys
- Network key; protect broadcast and group traffic, as well as authenticating to the network
 - Common key among all nodes
 - Can be distributed to a device in plaintext when it joins the network
- Link key: protect unicast traffic between two devices

Zigbee Encryption Keys

- Global Link Key: used by all nodes on the network
- Unique Link Key: used to encrypt communication between a pair of nodes
 - Preconfigured Link Key: used between trust center and a node;
 - Derived prior to joining
 - Trust Center Link Key: used between trust center and a node; distributed to node
 - Application Link Key: used between pair of nodes; distributed to node

Zigbee: Key Provisioning

- Significant challenge; process of provisioning, rotating, and revoking keys on devices
- Zigbee Pro; Administrator can use the SKKE method to derive the network and link keys on devices
 - Requires devices to have master key provisioned on the TC and device joining the network

Zigbee: Key Provisioning

- Key transport; network and link keys are sent in plaintext over the wireless network to the device when it joins
 - Can Easily be intercepted
- Pre-installation; administrator preconfigures all devices with the desired encryption keys at the manufacturing process
 - How to accommodate key revocation and rotation methods?
 - Manual changes to each device to change keys

Zigbee Authentication

- MAC address validation through ACL;
 - A list of authorized devices is maintained on each node
 - Challenging to keep the list up-to-date (memory req.)
- Standard mode; TC grants access by issuing a network key
 - Sent in plaintext
- High security mode; SKKE method to derive the network key
 - 4-way handshake

Zigbee Authentication's Vulnerability

- No mutual authentication is used in standard Zigbee (except high security mode with SKKE)
 - Authenticating node accepts the identity of TC for the delivery of network key without any validity check to verify the identity of the network
 - Easy to impersonate a legitimate network by using the same PAN ID as target, potentially on a different channel

Zigbee Attacks

- KillerBee:
 - Python-based framework for manipulating and penetration testing Zigbee and IEEE 802.15.4 networks
 - Written and tested on Linux, free and open-source
 - Includes support for Scapy
 - Includes a variety of tools including zbwreshark, zbdump, and zbreplay

<https://github.com/riverloopsec/killerbee>

IEEE 802.15.4/ZigBee Security Research Toolkit

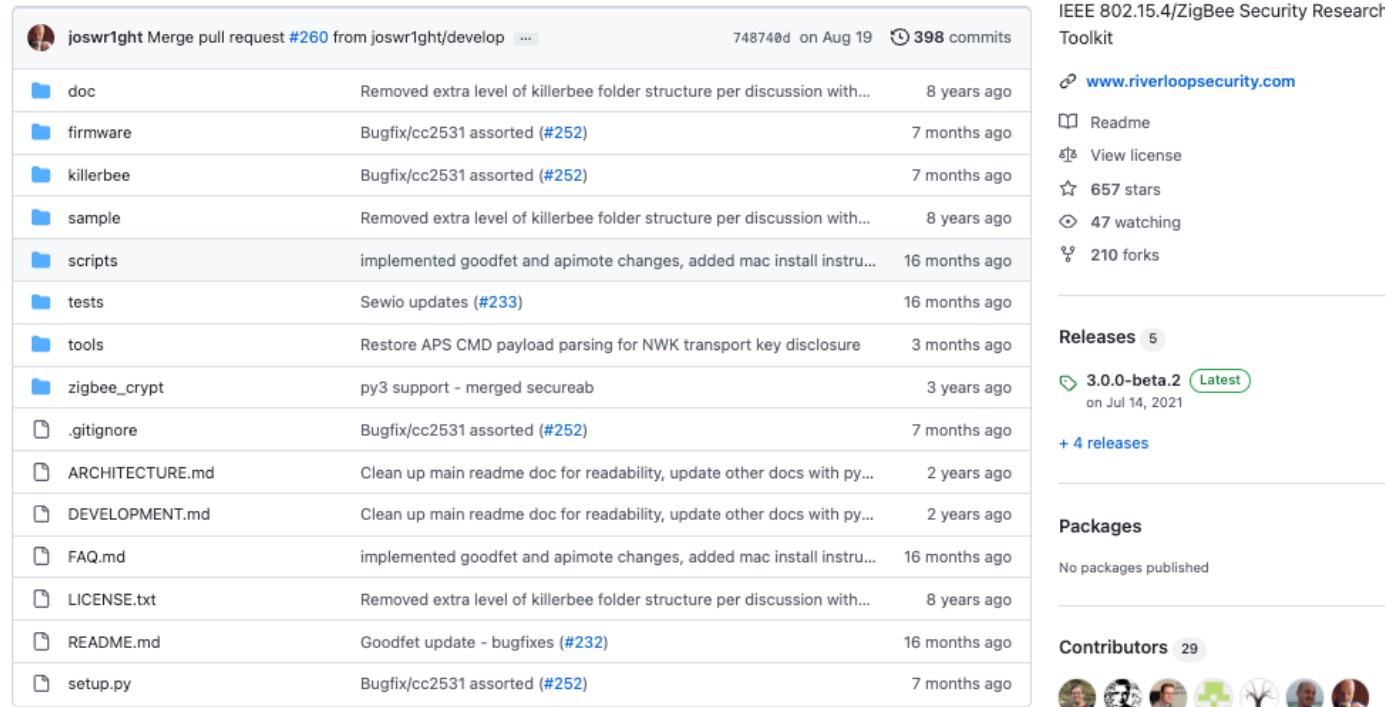
www.riverloopsecurity.com

Readme
View license
657 stars
47 watching
210 forks

Releases 5
[3.0.0-beta.2](#) (Latest) on Jul 14, 2021
+ 4 releases

Packages
No packages published

Contributors 29



The screenshot shows the GitHub repository page for 'killerbee'. At the top, it displays a merge pull request from 'joswr1ght' with 398 commits. Below this is a list of files and their commit history, including 'doc', 'firmware', 'killerbee', 'sample', 'scripts', 'tests', 'tools', 'zigbee_crypt', '.gitignore', 'ARCHITECTURE.md', 'DEVELOPMENT.md', 'FAQ.md', 'LICENSE.txt', 'README.md', and 'setup.py'. To the right of the file list, there are sections for 'Releases' (with 5 releases, the latest being '3.0.0-beta.2' from July 14, 2021), 'Packages' (no packages published), and 'Contributors' (29 contributors shown with small profile icons). The background of the slide features a faint network graph.

Z-Wave Outline

Z-Wave

Overview

Protocol Stack

Security

Z-Wave

- Low-energy, mesh-networking protocol
- Predominately used in home automation (locks, garage door openers, thermostats)
 - Over 100 million products in use in homes
- Proprietary design by Sigma Systems and governed by standards established by Z-Wave Alliance
 - Does not share details of protocol outside of NDA (nondisclosure agreement)
 - Controls all fabrication and delivery of Z-Wave chips to product manufacturers

Z-Wave

- Aggressive power conservation for long battery life
 - Like Zigbee
- Using a mesh networking model to accommodate greater device range
- Relatively simple protocol
- Support for positive acknowledgement and frame retransmission, self-forming and dynamic routing topology updates, and application-specific profiles

Z-Wave Overview

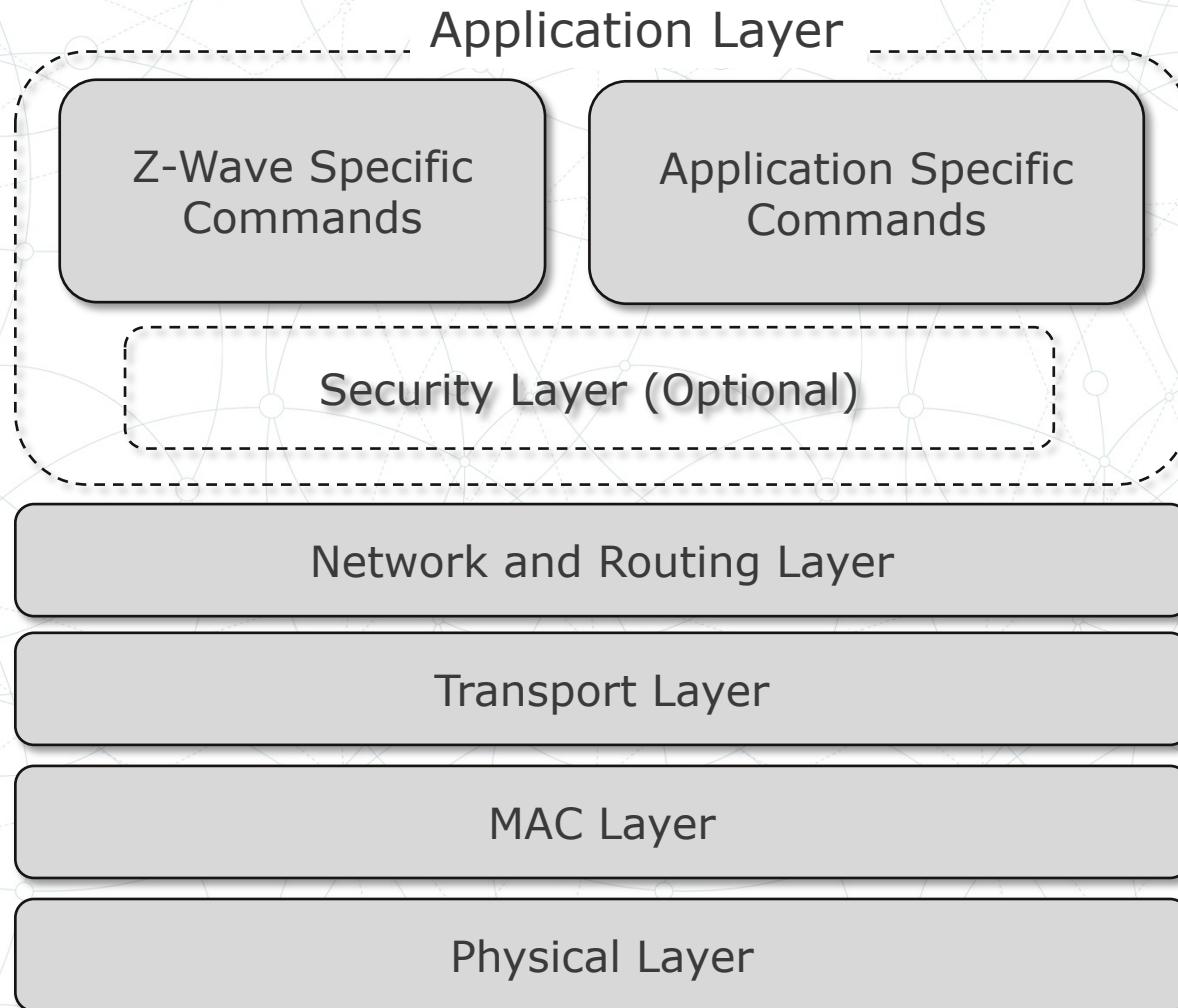
- Z-Wave is based on a mesh network topology
 - Each (non-battery) device installed in the network becomes a signal repeater
 - As a result, the more devices you have in your home, the stronger the network becomes
- While Z-Wave signals easily travel through most walls, floors and ceilings, the devices can also intelligently route themselves around obstacles to attain seamless, robust, whole-home coverage

Z-Wave Overview

- While Z-Wave has a range of 100 meters (or 328 feet) in open air, building materials reduce that range (roughly every 30 feet)
 - Operates at 908.42 MHz in the United States and Canada
- The Z-Wave networks can be linked together for even larger deployments
- Each Z-Wave network can support up to 232 Z-Wave devices

Z-Wave Protocol Stack

- Uses structured protocol stack



Z-Wave PHY Layer

- Uses sub-1-GHz band to accommodate frequency variations in different countries
- Offers 3 different RF Profiles (R1, R2, and R3) with unique data rates, encodings, modulation, and packet frame sizes
 - R1 is deprecated by Z-Wave alliance but might be in use for some
 - Range from 3 – 75 feet; depending on transmit power

| Profile | Data Rate | Encoding | Modulation | Packet Size |
|---------|-----------|------------|------------|-------------|
| R1 | 9.6 Kb/s | Manchester | FSK | 64 bytes |
| R2 | 40 Kb/s | NRZ | FSK | 64 bytes |
| R3 | 100 Kb/s | NRZ | FSK | 170 bytes |

Z-Wave MAC Layer

- Responsible for several attributes;
 - Packet framing and formatting
 - Positive ACK
 - Error detection
 - Retransmission of packets
 - Unicast, broadcast, and multicast processing
 - Address selection and allocation functions

Z-Wave MAC Layer

- Basic architecture of Z-Wave network: Controller device and slave devices
- Single primary controller device is responsible for establishing the network and selecting unique network identifier (i.e., HomeID)
- Controller devices are able to initiate a transmission on the network (polling or updating target devices) and responsible for maintaining network routing information
- Slave devices follow the instructions of controllers without dealing with how

Z-Wave MAC Layer Frame Format

- HomeID: Randomly selected 4-byte value chosen by controller
- NodeID: 1-byte value assigned to the node by the controller
 - Max 232 nodes (22 left for reserved, 1 for broadcast, 1 uninitialized)
- Frame Control: 16-bit field with several subfields

Z-Wave R1/R2 Frame Format

| | | | | | | |
|--------|----------------|---------------|--------|---------------------|---------|-----|
| HomeID | Source Node ID | Frame Control | Length | Destination Node ID | Payload | FCS |
|--------|----------------|---------------|--------|---------------------|---------|-----|

Z-Wave R3 Frame Format

| | | | | | | | |
|--------|----------------|---------------|--------|-----------------|---------------------|---------|-----|
| HomeID | Source Node ID | Frame Control | Length | Sequence number | Destination Node ID | Payload | FCS |
|--------|----------------|---------------|--------|-----------------|---------------------|---------|-----|

Z-Wave Network Layer

- Defines device responsibilities and responsible for other network components such as HomeID selection and NodeID allocation process as well as network route establishment
- Z-Wave inclusion and exclusion for network connections

Z-Wave Network Layer: Inclusion

- Involves configuring the controller in inclusion mode (allowing it to accept new nodes) by pressing a physical button or choosing a menu item, and pressing a button on the new node to initiate an inclusion exchange
- When the new node initiates the inclusion process, it sends a Z-Wave node information frame using homeID of 0x00000000 and nodeID of 0x00 and a broadcast dest NodeID
 - Discloses the capabilities of the new device to the controller, which, in turn, allocates a NodeID to the new device for subsequent use on the network and updates routing tables to accommodate packet delivery to the new node

Z-Wave Network Layer: Exclusion

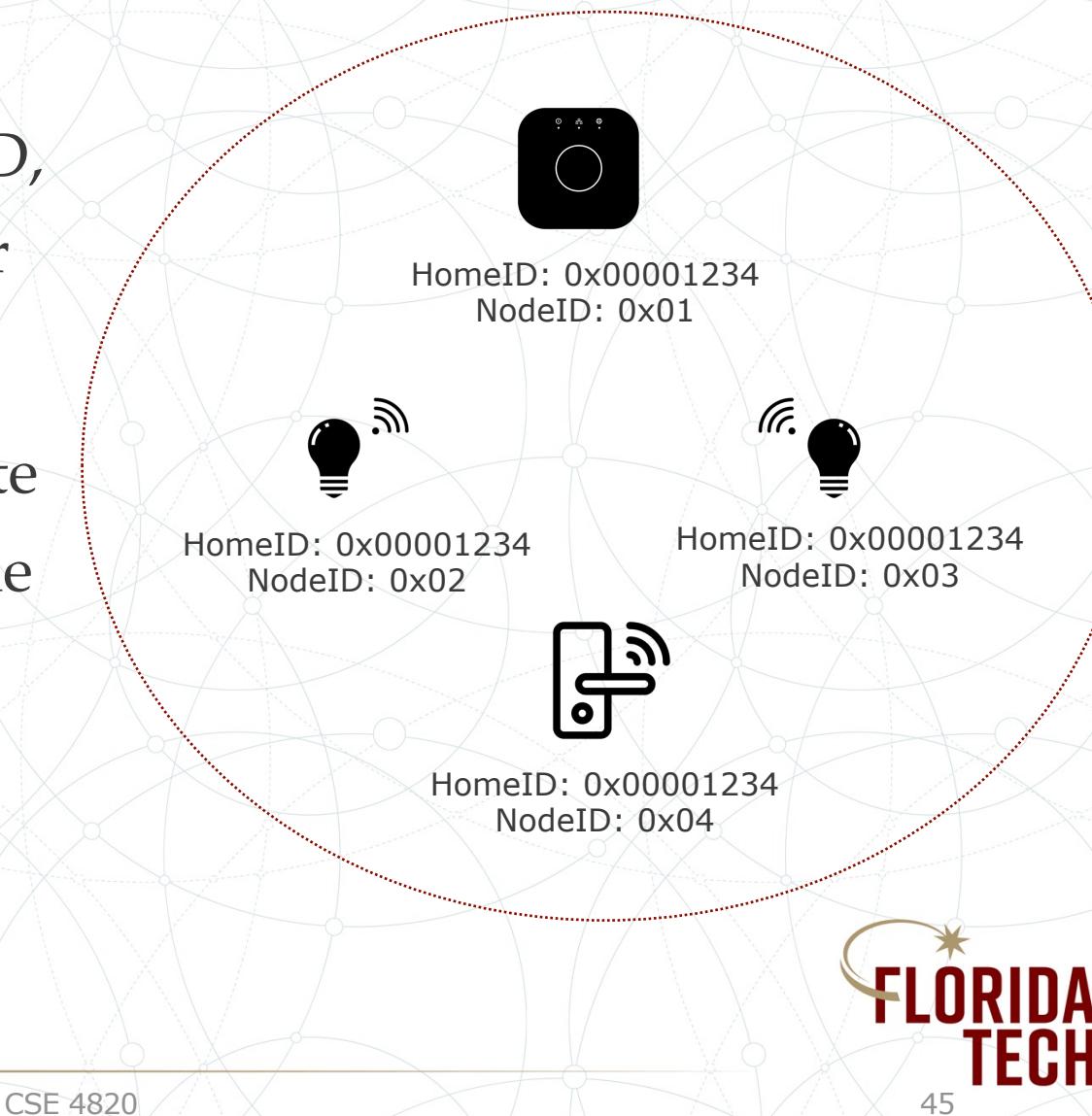
- Similar to inclusion but functionally opposite
- A node joined the Z-Wave network via inclusion cannot leave the network to join a different Z-Wave controller without completing the exclusion process
- Involves pressing a physical button on the controller and the device node, causing device to return to unallocated nodeID 0x00

Z-Wave Inclusion & Exclusion

- Strong component of the overall Z-Wave security
 - Requires physical access to the controller
- Is it secure enough?
 - What about spoofing?

Z-Wave Network Topology

- Nodes in a Z-Wave have a 1-byte NodeID, which must be different than every other node in the network
- Nodes in a Z-Wave network have a 4-byte HomeID, assigned by the controller at the time of inclusion
- Nearby networks must have different HomeIDs



Z-Wave Application Layer

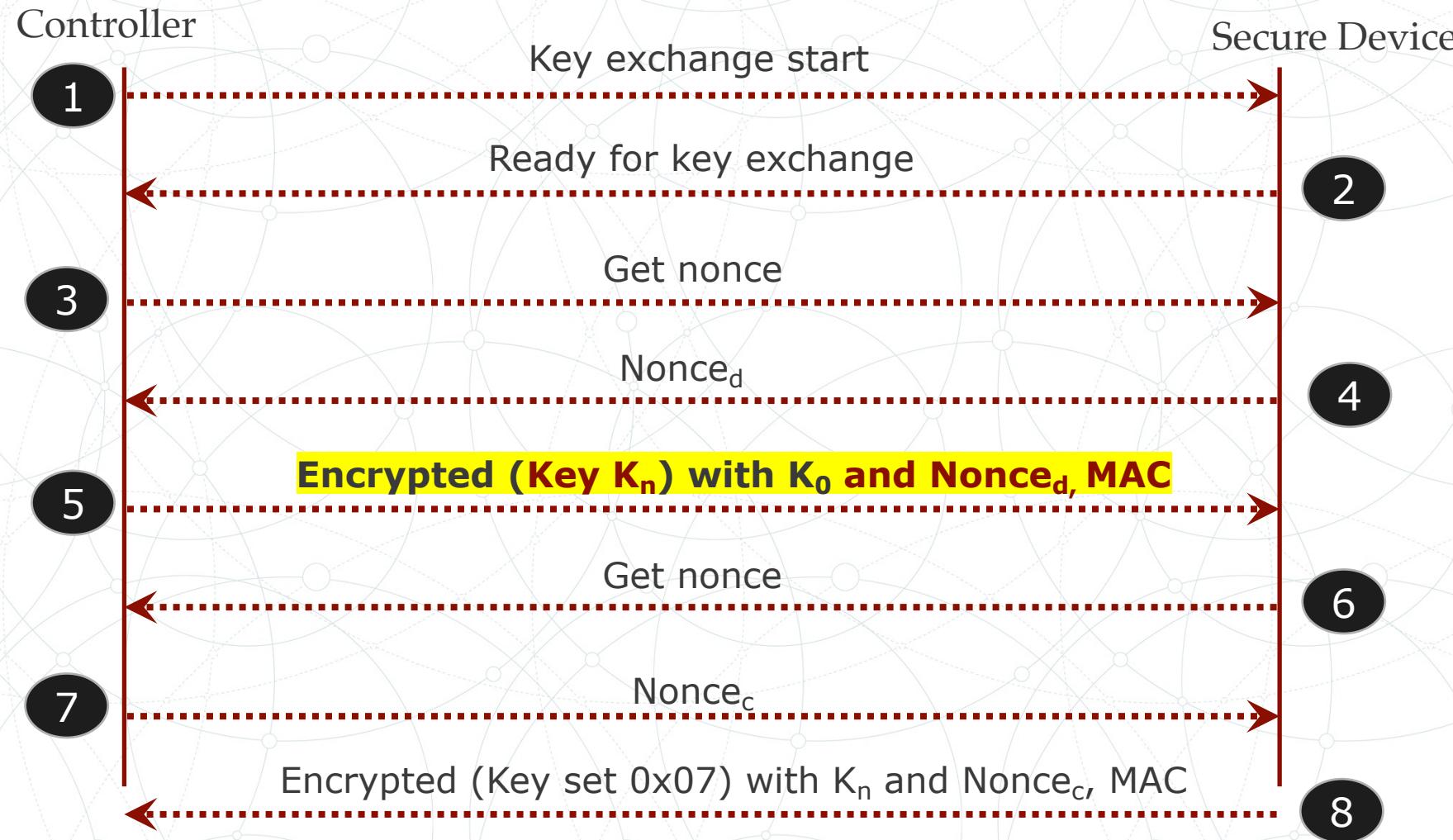
- Responsible for parsing and processing the data requests and responses in the packet payload
- Handles both application-specific and Z-Wave application control data
- Basic application payload format:



Z-Wave Security

- Uses AES-OFB (Output Feedback Mode) to provide data confidentiality on the network
 - Conserve the amount of payload content transmitted in Z-Wave frames while being NIST (National Institute of Standards and Technology) approved
- AES CBC-MAC (cipher block chaining message authentication code) for data integrity protection
- CLASS_SECURITY command; key exchange process to derive keys

Z-Wave Key Exchange Process



Temporal Key
 $K_0 = 16$ bytes of 0x00

Network Key
 $K_n = \text{Chosen by controller}$

Z-Wave Key Exchange Vulnerabilities

- MitM:
 - The secure device does not validate the identity of controller other than validating the MAC of encrypted Kn message using the temporary key K_0
 - Attacker can use any Z-Wave controller that support CLASS_SECURITY command class to intercept the inclusion process with a target device
 - Causing victim to associate to a malicious network

Z-Wave Key Exchange Vulnerabilities

- Key recovery attack:
 - There is no confidentiality protection in the delivery of the K_n key over the network since the K_0 is well known
 - Attacker passively observing the inclusion process can recover the network key K_n
 - Use it later to decrypt or forge arbitrary packets on the network

Z-Wave Key Exchange's Solution

- Low power inclusion mode
 - Controller and secure device transmit using minimal power capabilities
 - Require no more than 3 feet apart to complete the process
 - Also infrequent practice of adding new devices
 - Results in less opportunity for the attacker

Z-Wave Key Derivation

- After the K_n (network key) is established, the device generates two additional keys K_c (packet encryption) and K_m (message auth key)
 - K_c (packet encryption) = AES-ECB_{kn}(Password_c)
 - K_m (message auth key) = AES-ECB_{kn}(Password_m)
- Where Password_c and Password_m are static values across all Z-Wave devices
- However, if we observe Kn and we know Password_c and Password_m, we can compute the packet encryption and message auth keys

Z-Wave Key Establishment Attack

- We can force the key establishment process to establish a new K_n if we missed the original key establishment
 - Similar to an IEEE 802.11 Deauth

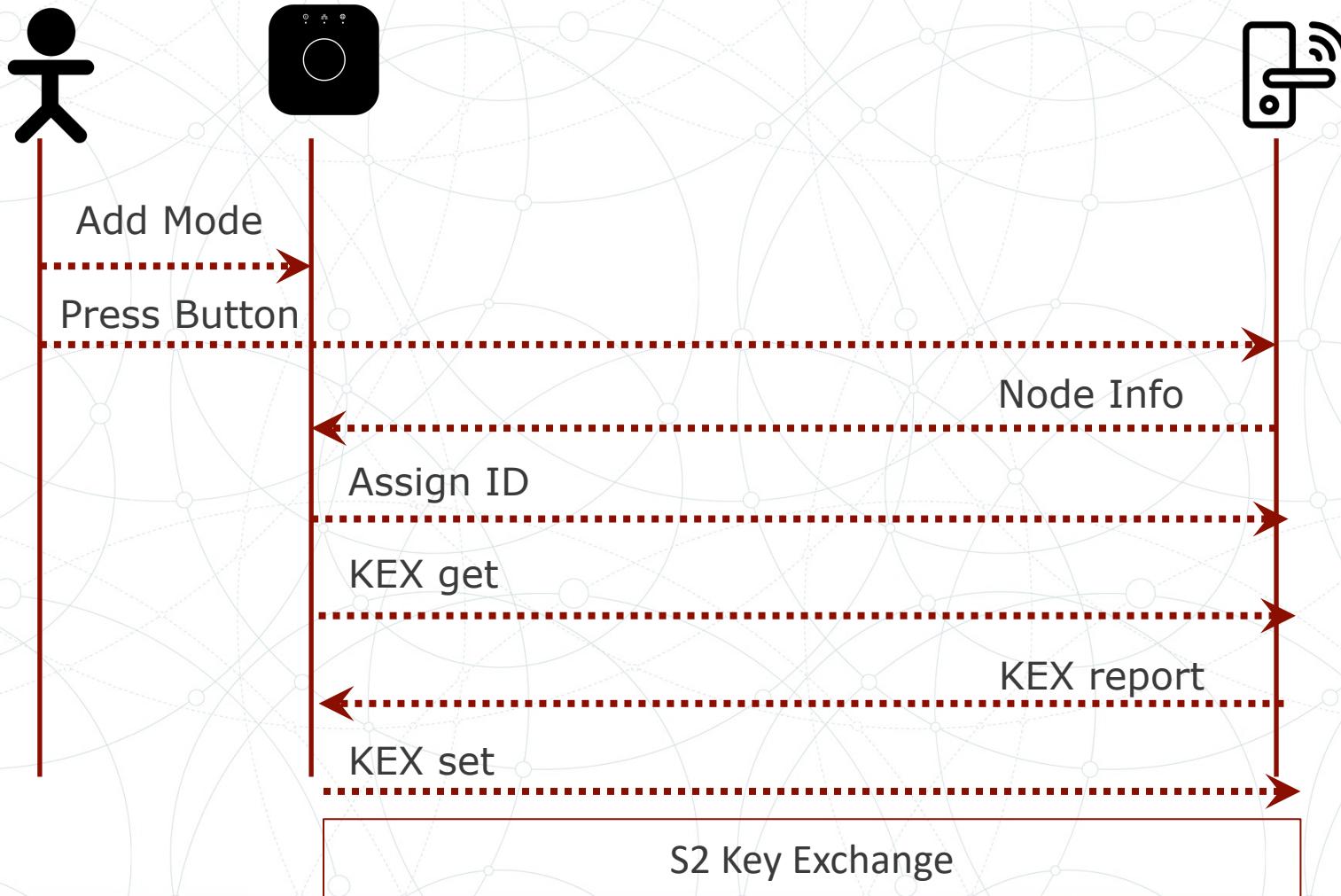


Z-Wave Key Establishment: Solution

- In response to the flawed S_0 Key Establishment and Pairing Process, SI Labs (i.e., Z-Wave Alliance) responded with a S_2 Key Establishment
- Removed the null temporal key
- Each devices has a DSK [device specific key] – 16 byte key
- Key exchanged using Diffie Hellman key exchange protocol
- Removed issues with man-in-the-middle
 - Are we safe now?

https://community.silabs.com/s/topic/0TO1M000000qHcQWAU/zwave?language=en_US&tabset-178da=2

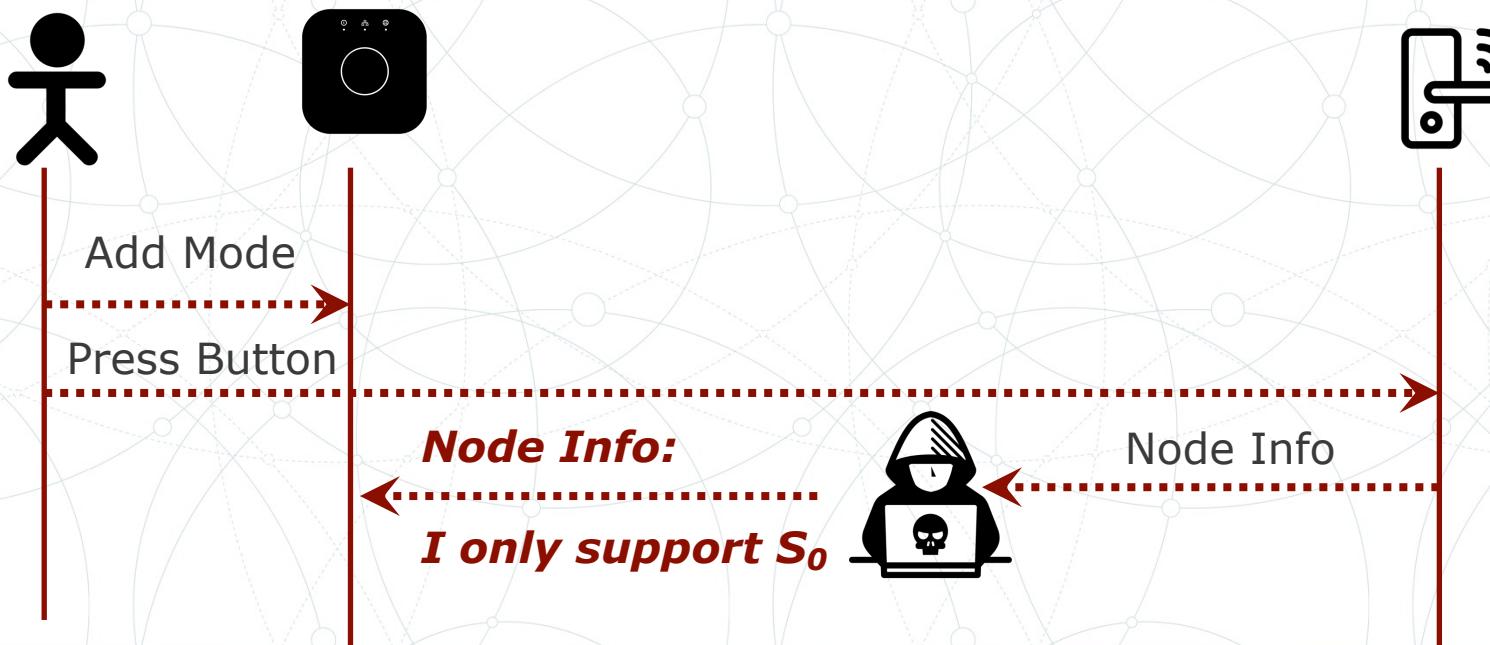
Z-Wave Key Exchange



S2 Z-Wave Rollback Attack

- An attacker jams the node info or responds prior to the node
- Provides node info that device only supports flawed S_0 pairing

mode



Mitigating Eavesdropping in Z-Wave

- As in any wireless technology, attacker can always capture the traffic (aka eavesdropping)
 - Useless if confidentiality and integrity of the data is ensured
- Unfortunately, the use of encryption in Z-Wave is optional
 - Switch to a vendor that offers network confidentiality and integrity control

Injection Attacks in Z-Wave

- If no encryption, attacker can capture and replay the packets
- Injection is also done if the device is in the Z-Wave network
 - To be in the network, Z-Wave inclusion needed
 - Does it solve this problem?
 - Attacker can spoof the address and inject any packet

<https://github.com/joswr1ght/killerzee>

Scapy-Radio Framework

- Modified version of Scapy to support
 - Zwave
 - Zigbee
 - 802.15.4
- Works with software defined radios (SDR)

| | | | | |
|-----------------------|---|-----|------------------------|--------------|
| Balint Seeber | Removed deprecated function from Wireshark dissector. | ... | f1240ab on Apr 1, 2016 | ⌚ 12 commits |
| gnuradio | Removed deprecated function from Wireshark dissector. | | | 7 years ago |
| scapy | disables xbee for the moment | | | 8 years ago |
| utils/Zwave | Add copyright stuff | | | 8 years ago |
| wireshark/scapy-radio | Removed deprecated function from Wireshark dissector. | | | 7 years ago |
| .hgignore | initial import of scapy-radio | | | 8 years ago |
| README.md | Add note about GNU Radio 3.7.5 in README | | | 8 years ago |
| install.sh | Bugfix on installation script | | | 8 years ago |

☰ README.md

Introduction

This tool is a modified version of scapy that aims at providing an quick and efficient pentest tool with RF capabilities.

It includes:

- A modified version of scapy that can leverage GNU Radio to handle a SDR card
- GNU Radio flow graphs (GRC files) we have build that allows full duplex communication
- GNU Radio blocks we have written to handle several protocols

LoRaWAN Outline

LoRaWAN

As a Wireless Standard

History

Characteristics

Network Architecture

Message Types

Adaptive Data Rates

Security

LoRaWAN

- Long Range (LoRa) Wide Area Network (WAN)
 - “A Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated ‘things’ to the internet in regional, national or global networks,
 - And support targets key Internet of Things (IoT) requirements such as bi-directional communication, end-to-end security, mobility and localization services”

LoRaWAN Protocol

- The LoRaWAN specification is open so anyone can set up and operate a LoRa network
 - Wireless audio frequency technology that operates in a license-free radio frequency spectrum
- It uses a narrow band waveform with a central frequency to send data
 - Makes it robust to interference

LoRaWAN Protocol

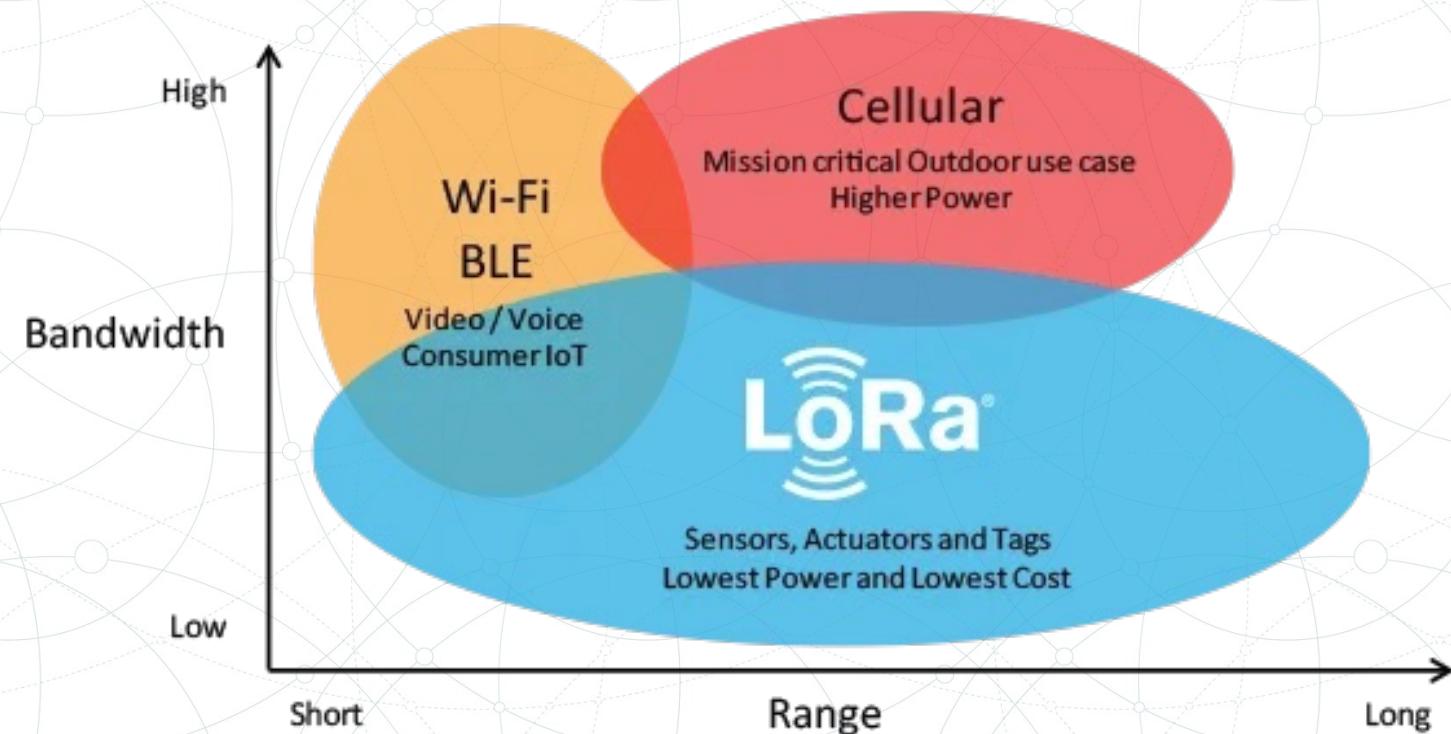
- LoRa is a physical layer protocol that uses spread spectrum modulation and supports long-range communication at the cost of a narrow bandwidth
 - Modulation technique derived from Chirp Spread Spectrum (CSS) technology
 - It encodes information on radio waves using chirp pulses;
 - Similar to the way dolphins and bats communicate!

LoRaWAN as Wireless Standard

- LoRa is ideal for applications that transmit small chunks of data with low bit rates
- Data can be transmitted at a longer range compared to technologies like WiFi, Bluetooth or ZigBee
- These features make LoRa well suited for sensors and actuators that operate in low power mode

LoRaWAN as Wireless Standard

- Suitable for transmitting small size payloads (like sensor data) over long distances
 - Greater communication range with low bandwidths than other competing wireless data transmission technologies



LoRaWAN Use Cases

- Vaccine cold chain monitoring;
 - LoRaWAN sensors are used to ensure vaccines are kept at appropriate temperatures in transit
- Animal conservation;
 - Tracking sensors manage endangered species such as Black Rhinos and Amur Leopards
- Dementia patients;
 - Wristband sensors provide fall detection and medication tracking

Characteristics of LoRaWAN

- LoRaWAN is a Media Access Control (MAC) layer protocol built on top of LoRa modulation
 - A software layer which defines how devices use the LoRa hardware, for example when they transmit, and the format of messages
- Long range communication up to 10 miles in line of sight
- Long battery duration of up to 10 years
 - For enhanced battery life, you can operate your devices in class A or class B mode, which requires increased downlink latency

Characteristics of LoRaWAN

- Low cost for devices and maintenance
- License-free radio spectrum but region-specific regulations apply
- Has a limited payload size of 51 bytes to 241 bytes depending on the data rate
 - The data rate can be 0,3 Kbit/s – 27 Kbit/s

LoRaWAN Network Overview

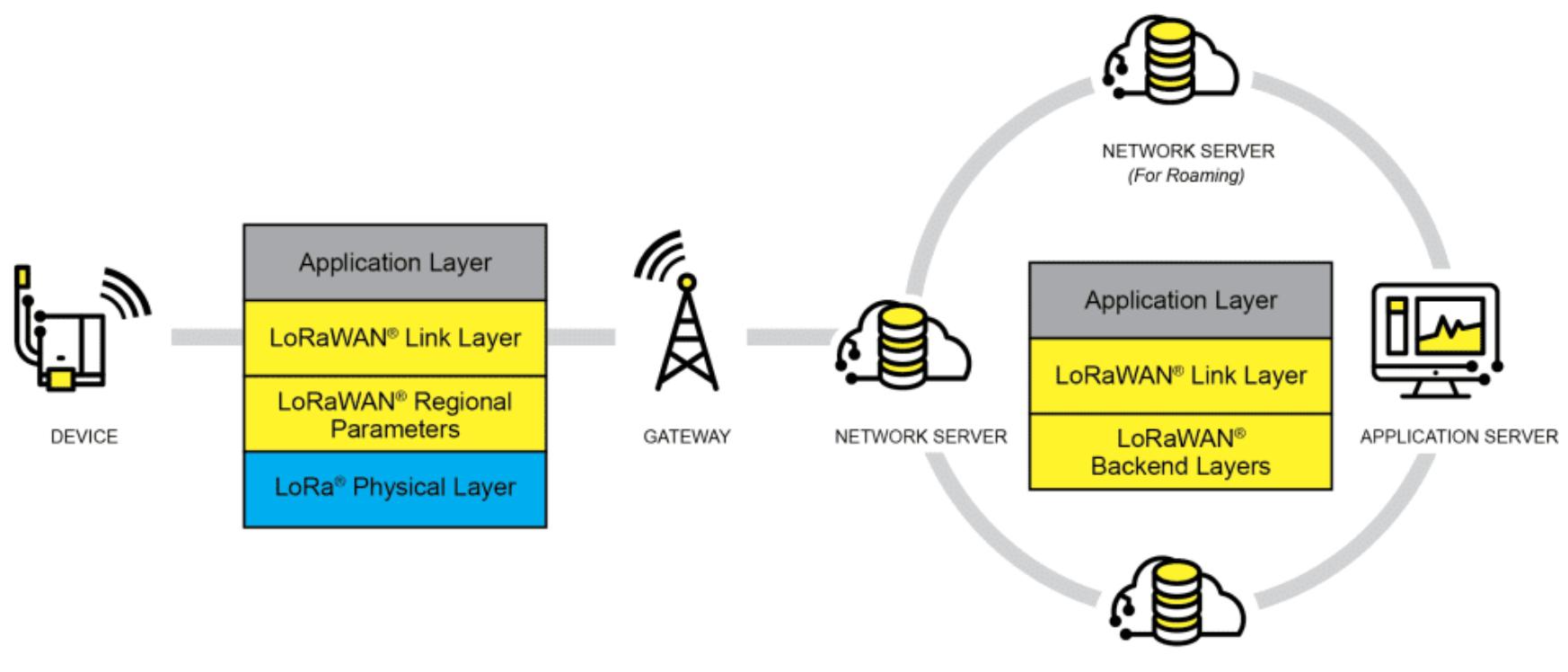
- LoRaWAN® network architecture is deployed in a star-of-stars topology in which gateways relay messages between end-devices and a central network server
- The gateways are connected to the network server via standard IP connections and act as a transparent bridge, simply converting RF packets to IP packets and vice versa

LoRaWAN Network Overview

- The wireless communication takes advantage of the Long Range characteristics of the LoRa physical layer, allowing a single-hop link between the end-device and one or many gateways
- All modes are capable of bi-directional communication,
 - And there is support for multicast addressing groups to make efficient use of spectrum during tasks such as Firmware Over-The-Air (FOTA) upgrades or other mass distribution messages

LoRaWAN Network Overview

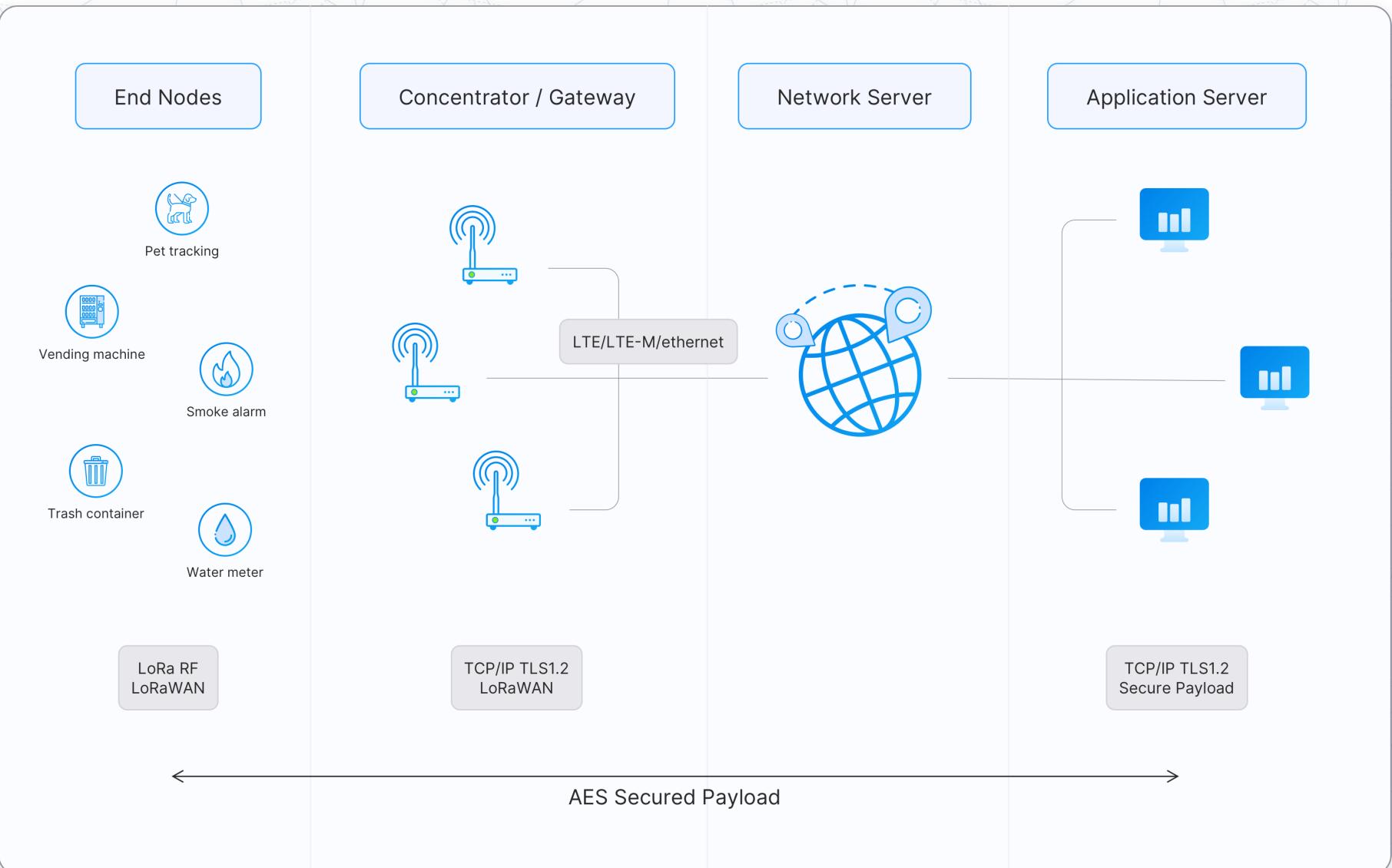
- The specification defines the device-to-infrastructure (LoRa®) physical layer parameters & (LoRaWAN®) protocol and so provides seamless interoperability between manufacturers



LoRaWAN Network Devices

- LoRaWAN Device Types:
 - End Devices
 - Gateways
 - Network Server
 - Application servers
 - Join Server

LoRaWAN Architecture



LoRaWAN: End Device

- A LoRaWAN end device can be a sensor, an actuator, or both
- They are often battery operated
- These end devices are wirelessly connected to the LoRaWAN network through gateways using LoRa RF modulation
- *LoRaWAN end device - The Things Industries Generic Node Sensor Edition:*
 - An end device that consists of sensors like temperature, humidity, and fall detection



LoRaWAN Device Classes

- LoRaWAN has three different classes of end-point devices to address the different needs reflected in the wide range of applications:
 - Class A – Lowest power, bi-directional end-devices
 - Class B – Bi-directional end-devices with deterministic downlink latency
 - Class C – Lowest latency, bi-directional end-devices

LoRaWAN: Gateways

- Each gateway is registered (using configuration settings) to a LoRaWAN network server
- A gateway receives LoRa messages from end devices and simply forwards them to the LoRaWAN network server
- Gateways are connected to the Network Server using a backhaul like Cellular (3G / 4G / 5G), WiFi, Ethernet, fiber-optic or 2.4 GHz radio links

LoRaWAN: Network Server

- The Network Server manages gateways, end-devices, applications, and users in the entire LoRaWAN network
- A typical LoRaWAN Network Server has the following features:
 - Establishing secure 128-bit AES connections for the transport of messages between end-devices and the Application Server (end-to-end security)
 - Validating the authenticity of end devices and integrity of messages
 - Deduplicating uplink messages
 - Selecting the best gateway for routing downlink messages

LoRaWAN: Network Server

- Sending ADR commands to optimize the data rate of devices
- Device address checking
- Providing acknowledgements of confirmed uplink data messages
- Forwarding uplink application payloads to the appropriate application servers
- Routing uplink application payloads to the appropriate Application Server
- Forwarding Join-request and Join-accept messages between the devices and the join server
- Responding to all MAC layer commands

LoRaWAN: Application Server

- Processes application-specific data messages received from end devices
- It also generates all the application-layer downlink payloads and sends them to the connected end devices through the Network Server
- A LoRaWAN network can have more than one Application Server
- The collected data can be interpreted by applying techniques like machine learning and artificial intelligence to solve business problems

LoRaWAN: Join Server

- Assists in secure device activation, root key storage, and session key generation
- The join procedure is initiated by the end device by sending the Join-request message to the Join Server through the Network Server
- The Join-server processes the Join-request message, generates session keys, and transfers NwkSKey and AppSKey to the Network server and the Application server respectively
- The Join Server was first introduced with LoRaWAN v1.1

LoRaWAN MAC Message Types

| LoRaWAN 1.0.x | LoRaWAN 1.1 | Description |
|-----------------------|-----------------------|--|
| Join-request | Join-request | An uplink message, used by the over-the-air activation (OTAA) procedure |
| Join-accept | Join-accept | A downlink message, used by the over-the-air activation (OTAA) procedure |
| Unconfirmed Data Up | Unconfirmed Data Up | An uplink data frame, confirmation is not required |
| Unconfirmed Data Down | Unconfirmed Data Down | A downlink data frame, confirmation is not required |
| Confirmed Data Up | Confirmed Data Up | An uplink data frame, confirmation is requested |
| Confirmed Data Down | Confirmed Data Down | A downlink data frame, confirmation is requested |
| RFU | Rejoin-request | 1.0.x - Reserved for Future Usage 1.1 - Uplink over-the-air activation (OTAA) Rejoin-request |
| Proprietary | Proprietary | Used to implement non-standard message formats |

LoRaWAN: Join-Request

- The Join-request message is always initiated by an end device and sent to the Network Server
- In LoRaWAN versions earlier than 1.0.4 the Join-request message is forwarded by the Network Server to the Application Server
 - In LoRaWAN 1.1 and 1.0.4+, the Network Server forwards the Join-request message to the device's Join Server
- The Join-request message is not encrypted

LoRaWAN: Join-Accept

- In LoRaWAN versions earlier than 1.0.4 the Join-accept message is generated by the Application Server
 - In LoRaWAN 1.1 and 1.0.4+ the Join-accept message is generated by the Join Server
 - In both cases the message passes through the Network Server
- Then the Network Server routes the Join-accept message to the correct end-device

LoRaWAN: Join-Accept

- The Join-accept message is encrypted as follows
 - In LoRaWAN 1.0, the Join-accept message is encrypted with the AppKey
 - In LoRaWAN 1.1, the Join-accept message is encrypted with different keys:

| If triggered by | Encryption Key |
|---------------------------------|----------------|
| Join-request | NwkKey |
| Rejoin-request type 0, 1, and 2 | JSEncKey |

Message Integrity Code (MIC)

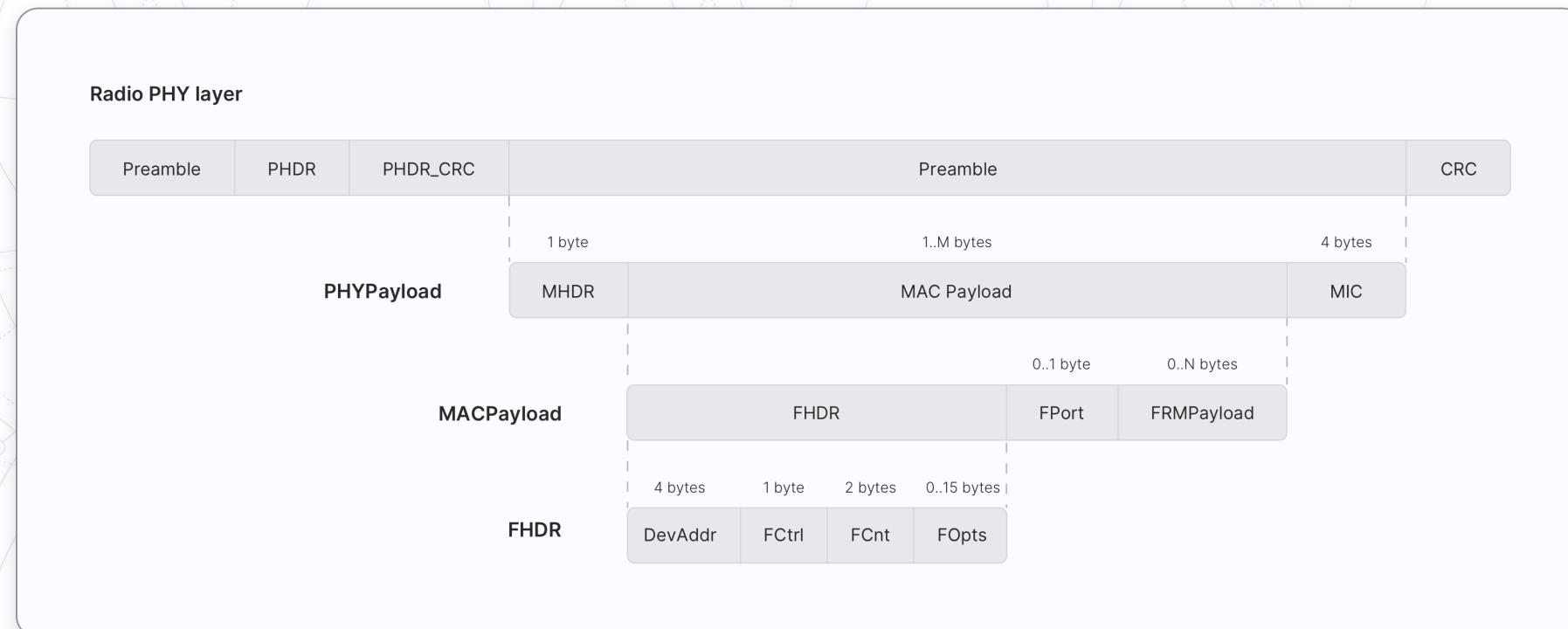
- The Message Integrity Code (MIC) ensures the integrity and authenticity of a message
- The message integrity code is calculated over all the fields in the message and then added to the message itself

| Message Type | Fields |
|--------------------------------|--|
| Join-request | MHDR JoinEUI DevEUI DevNonce |
| Join-accept | MHDR JoinNonce NetID DevAddr DLSettings RxDelay CFList |
| Rejoin-request Type 0 and 2 | MHDR Rejoin Type NetID DevEUI RJcount0 |
| Rejoin-request Type 1 | MHDR Rejoin Type JoinEUI DevEUI RJcount1 |
| Data messages (up and down) | MHDR FHDR FPort FRMPayload |

LoRaWAN Data Messages

- Are used to transport both MAC commands and application data which can be combined together in a single message

- Data messages can be confirmed or unconfirmed



LoRaWAN End Device Activation

- Every end device must be registered with a network before sending and receiving messages
 - This procedure is known as activation
- There are two activation methods available:
 - Over-The-Air-Activation (OTAA)
 - Activation By Personalization (ABP)

OTAA

- The most secure and recommended activation method for end devices
- Devices perform a join procedure with the network, during which a dynamic device address is assigned and security keys are negotiated with the device
- The join procedure for LoRaWAN 1.0.x and 1.1 is slightly different

OTAA in LoRaWAN 1.1

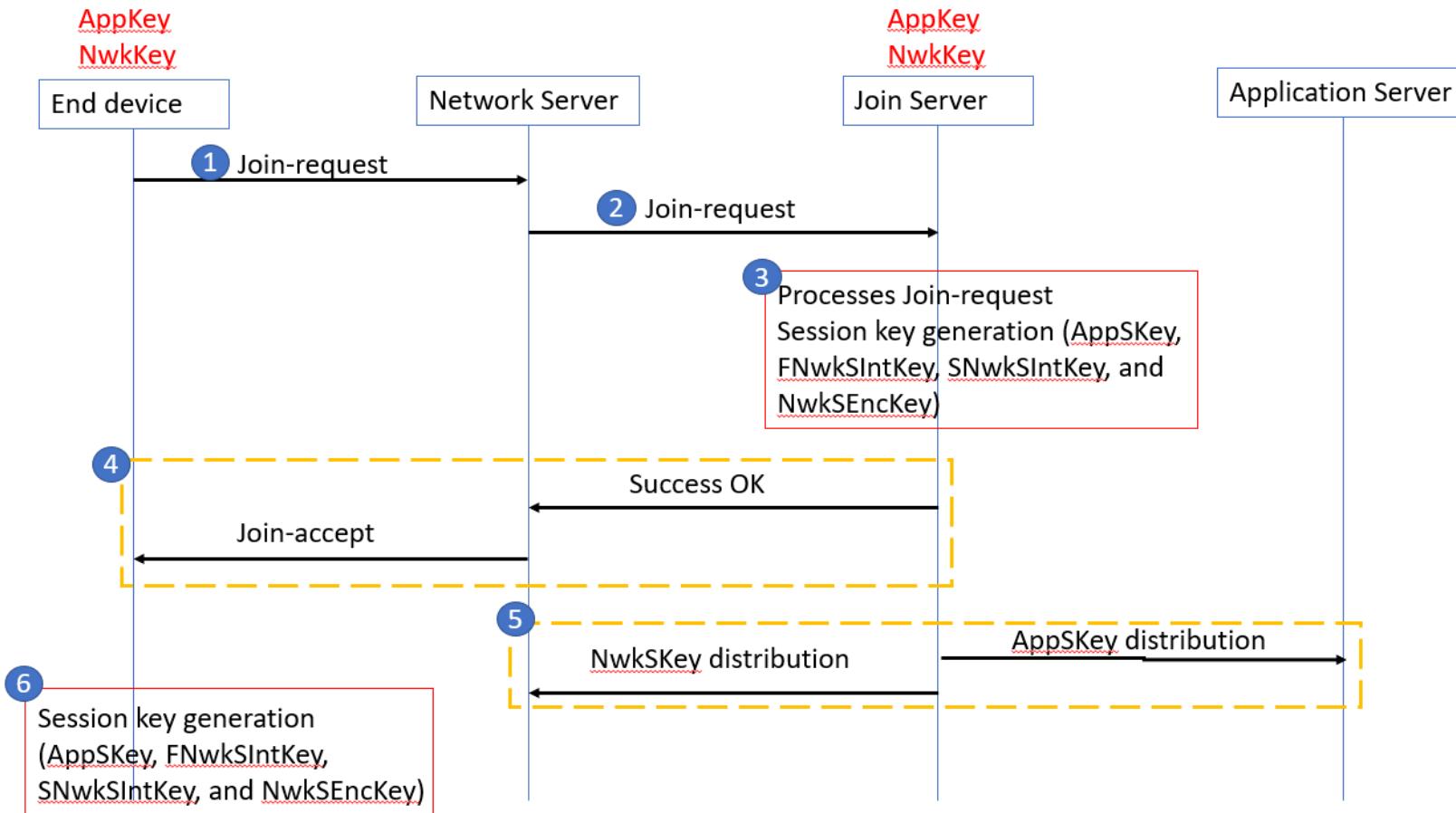
- The join procedure requires two MAC messages to be exchanged between the end device and the Join Server:
 - Join-request - from end device to the Join Server
 - Join-accept - from Join Server to the end device
- Before activation, the JoinEUI, DevEUI, AppKey, and NwkKey should be stored in the end device

OTAA in LoRaWAN 1.1

- The AppKey and NwkKey are AES-128 bit secret keys known as root keys
- The matching AppKey, NwkKey, and DevEUI should be provisioned onto the Join Server that will assist in the processing of the join procedure and session key derivation
- The JoinEUI and DevEUI are not secret and visible to everyone
- **Note:** The AppKey and NwkKey are never sent over the network

OTAA

- The join procedure is always initiated by the end device
- The end device sends the Join-request message to the network that is going to be joined



Activation By Personalization (ABP)

- ABP directly ties an end-device to a pre-selected network, bypassing the over-the-air-activation procedure
 - Requires hardcoding the device address as well as the security keys in the device
- ABP is the less secure activation method, and also has the downside that devices can not switch network providers without manually changing keys in the device
 - A Join Server is not involved in the ABP process
- An end device activated using the ABP method can only work with a single network and keeps the same security session for its entire lifetime

ABP in LoRaWAN 1.1

- The DevAddr and the four-session keys FNwkSIntKey, SNwkSIntKey, NwkSEncKey, and AppSKey are directly stored into the end device instead of the DevEUI, JoinEUI, AppKey, and NwkKey
- The same DevAddr, FNwkSIntKey, SNwkSIntKey, and NwkSEncKey should be stored in the Network Server and the AppSKey should be stored in the Application Server



LoRaWAN Data Rates

- In addition to frequency hopping, all communication packets between end-devices and gateways also include a variable 'Data rate' (DR) setting
- The selection of the DR allows a dynamic trade-off between communication range and message duration
- Also, due to the spread spectrum technology, communications with different DRs do not interfere with each other and create a set of virtual 'code' channels increasing the capacity of the gateway

LoRaWAN Data Rates

- To maximize both battery life of the end-devices and overall network capacity, the LoRaWAN network server manages the DR setting and RF output power for each end-device individually by means of an Adaptive Data Rate (ADR) scheme
- LoRaWAN baud rates range from 0.3 kbps to 50 kbps

LoRaWAN Security

- LoRaWAN 1.0 specifies a number of security keys: NwkSKey, AppSKey and AppKey
- All keys have a length of 128 bits
- The algorithm used for this is AES-128, similar to the algorithm used in the 802.15.4 standard

LoRaWAN Session Keys

- When a device joins the network (this is called a join or activation), an application session key AppSKey and a network session key NwkSKey are generated
 - The NwkSKey is shared with the network, while the AppSKey is kept private
 - These session keys will be used for the duration of the session

LoRaWAN Session Keys

- The Network Session Key (NwkSKey) is used for interaction between the Node and the Network Server
- This key is used to validate the integrity of each message by its Message Integrity Code (MIC check)
- This MIC is similar to a checksum, except that it prevents intentional tampering with a message

LoRaWAN Session Keys

- For this MIC, LoRaWAN uses AES-CMAC
 - This validation is also used to map a non-unique device address (DevAddr) to a unique DevEUI and AppEUI

The MIC of the Join-accept message is computed using the NwkKey:

```
cmac = aes128_cmac(NwkKey, MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay  
| CFList)  
  
MIC = cmac[0..3]
```

LoRaWAN Session Keys

- The Application Session Key (AppSKey) is used for encryption and decryption of the payload
- The payload is fully encrypted between the Node and the Handler / Application Server component of The Things Network, which you can run on your own server
 - This means that nobody except you is able to read the contents of messages you send or receive

```
AppSKey = aes128_encrypt(AppKey, 0x02 | JoinNonce | JoinEUI | DevNonce | pad16)
```

LoRaWAN Session Keys

- These two session keys (NwkSKey and AppSKey) are unique per device, per session
- If you dynamically activate your device (OTAA), these keys are re-generated on every activation
- If you statically activate your device (ABP), these keys stay the same until you change them

```
FNwkSIntKey = aes128_encrypt(NwkKey, 0x01 | JoinNonce | JoinEUI | DevNonce | pad16)
```

```
SNwkSIntKey = aes128_encrypt(NwkKey, 0x03 | JoinNonce | JoinEUI | DevNonce | pad16)
```

```
NwkSEncKey = aes128_encrypt(NwkKey, 0x04 | JoinNonce | JoinEUI | DevNonce | pad16)
```

LoRaWAN Application Keys

- The application key (AppKey) is only known by the device and by the application
- Dynamically activated devices (OTAA) use the Application Key (AppKey) to derive the two session keys during the activation procedure
- In The Things Network you can have a default AppKey which will be used to activate all devices or customize the AppKey per device

```
AppSKey = aes128_encrypt(AppKey, 0x02 | JoinNonce | JoinEUI | DevNonce | pad16)
```

LoRaWAN Security

- By providing these two keys, it becomes possible to implement 'multi-tenant' shared networks without the network operator having visibility of the users payload data
- The keys can be Activated By Personalisation (ABP) on the production line or during commissioning, or can be Over-The-Air Activated (OTAA) in the field
 - OTAA allows devices to be re-keyed if necessary

LoRaWAN: Frame Counters

- Anyone will be able to capture and store messages
 - It's not possible to read these messages without the AppSKey, because they're encrypted
 - Nor is it possible to tamper with them without the NwkSKey, because this will make the MIC check fail
 - It is however possible to re-transmit the messages
- These so-called replay attacks can be detected and blocked using frame counters

LoRaWAN: Frame Counters

- When a device is activated, these frame counters (FCntUp and FCntDown) are both set to 0
- Every time the device transmits an uplink message, the FCntUp is incremented and every time the network sends a downlink message, the FCntDown is incremented
- If either the device or the network receives a message with a frame counter that is lower than the last one, the message is ignored

LoRaWAN: Frame Counters

- These frame counters reset to 0 every time the device restarts (when you flash the firmware or when you unplug it)
- As a result, The Things Network will block all messages from the device until the FCntUp becomes higher than the previous FCntUp
- Therefore, you should re-register your device in the backend every time you reset it

Cellular Networks Outline

Cellular Networks

2G – Edge

Femtocell

4G / LTE

Cellular Network Radio Frequencies

- Modern cell phones use a number of different frequencies to transmit data
 - Depends on the country and mobile operator network
- Some countries provide government-owned cellular services
 - Some lease spectrum to mobile operators to provide cellular access

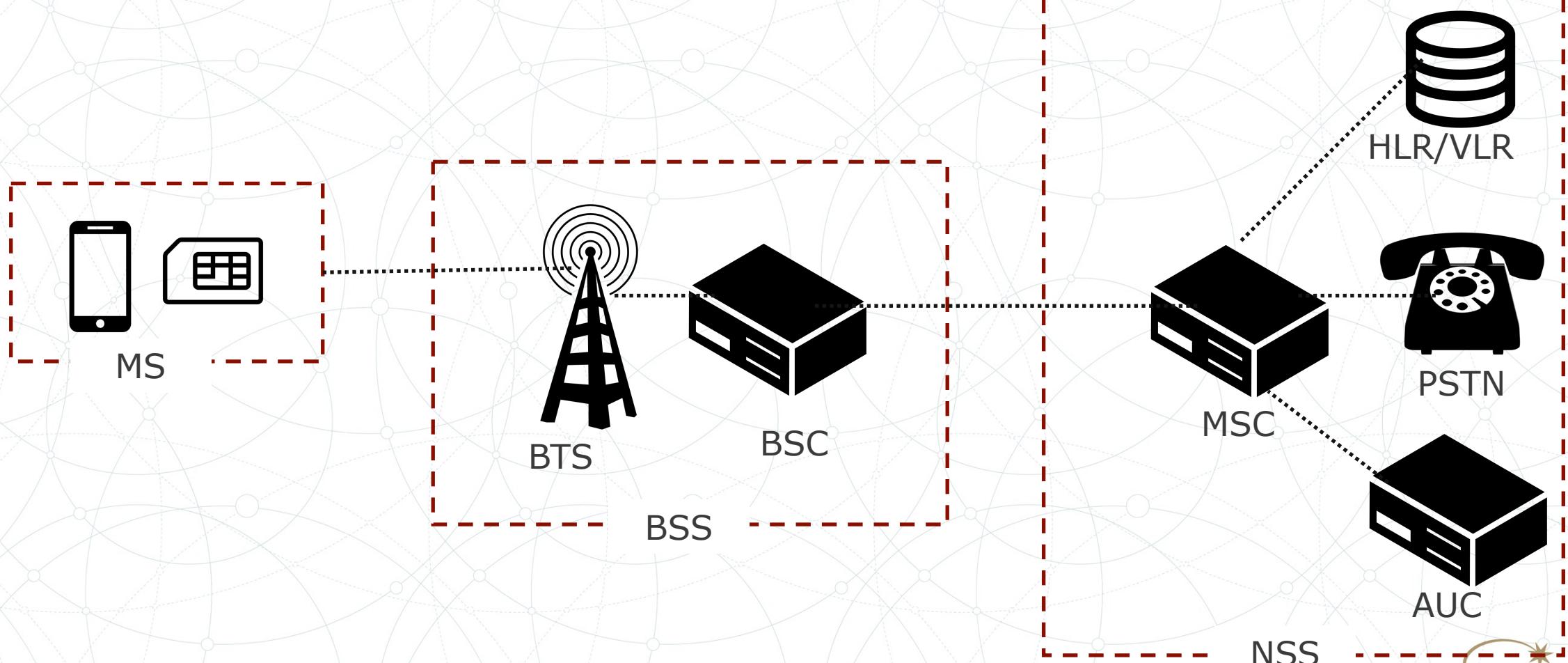
2G

- 2.5G Global System for Mobile (GSM), 2.75G General Packet Radio Service (GPRS), and Enhanced Data Rates for GSM Evolution (EDGE)
- Has been the most widespread cellular protocol used worldwide
 - Some devices still use it for backward-compatibility or back-up connection in the absence of other access opportunities

GSM Protocol

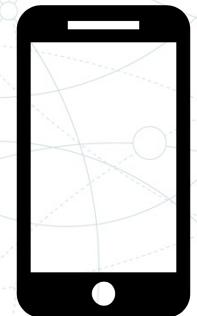
- Global System for Mobile Communications (GSM)
- First digital, circuit-switched network for carrying voice
- Expanded for carrying data (GPRS)
- Implemented cryptographic algorithms for security
- By mid-2010s, had over 90% of market share
- Obsoleted by LTE (4G); reached end-of-life for AT&T in 2017

GSM Network Model



GSM Mobile Station

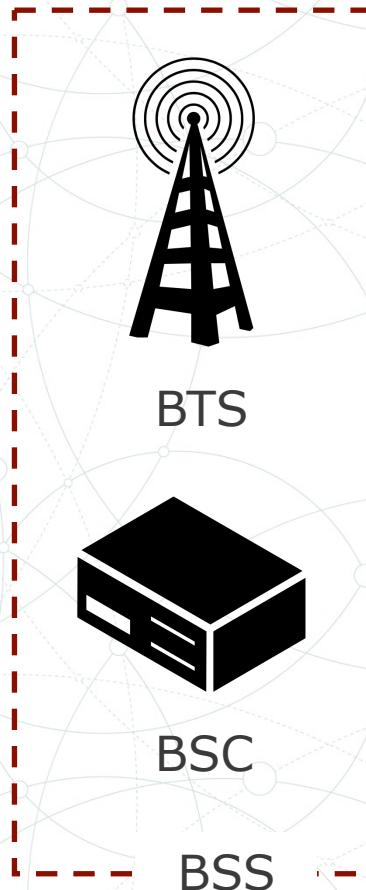
- Mobile Station (MS):
 - User handset/device that connects to the GSM network
- Subscriber Identity Mobile (SIM):
 - Removable media that identifies the unique International Mobile Subscriber Identifier (IMSI) for the mobile station and the 128-bit Authentication Key (K_i)
- International Mobile Subscriber Identifier (IMSI):
 - Unique identifier consisting of Mobile Country Code (3 digits), Mobile Network Code (2 | 3 digits), Mobile Subscriber Identification (10 | 15 digits)



MS

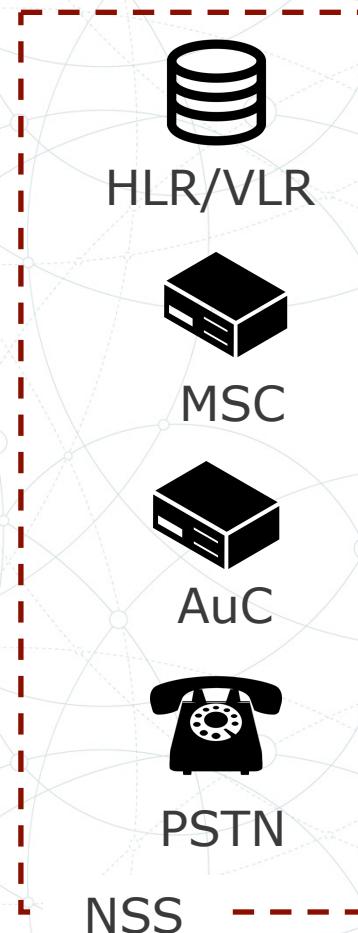
GSM Base Station Subsystem

- Base Transceiver Station (BTS):
 - Devices that facilitates radio connectivity for the mobile station (e.g., cell tower)
- Base Station Controller (BSC):
 - Facilitates the management of several BTSSs;
 - Handles radio allocation, BTS handovers, etc.
- Base Station Subsystem (BSS):
 - Consists of BTS/BSC;
 - GSM network component that connects mobile station and network switching system



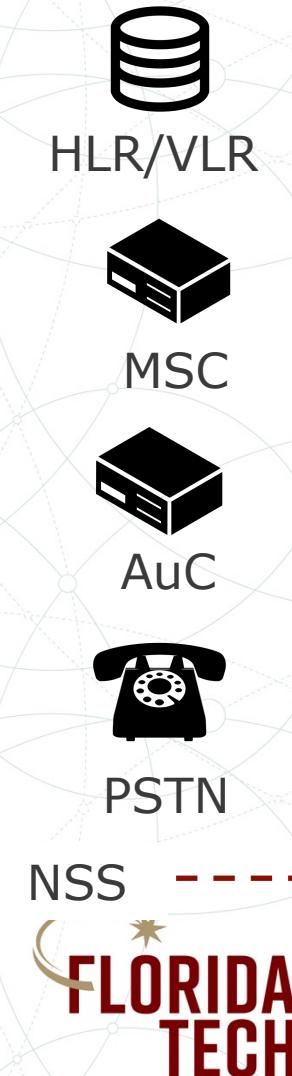
GSM Network Switching Subsystem

- Network Switching Subsystem (NSS):
 - Back-end network for GSM provider and services
- Home Location Register (HLR):
 - Database that contains the IMSI for each MS on the network
- Visitor Location Register (VLR):
 - Database that facilitates roaming operations for MS devices

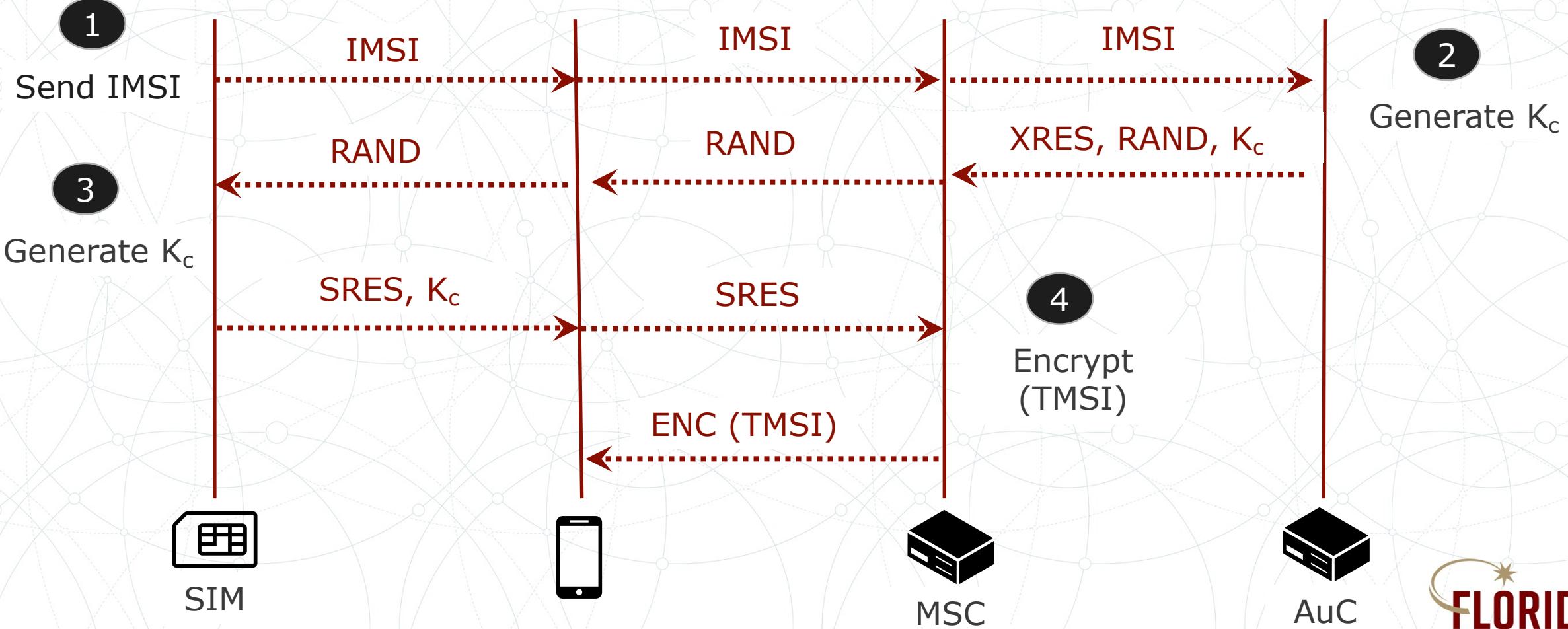


GSM Network Switching Subsystem

- Mobile Switching Center (MSC):
 - Responsible for routing
- Authentication Center (AuC):
 - Facilitates user identification and authentication for SIM cards
- Public Switched Telephone Network (PSTN):
 - Public interface to GSM network and other network providers



GSM Authentication



GSM Authentication

- Exchange validates the identity of the subscriber through the use of the IMSI and the associated subscriber key, K_i , stored on the SIM card
 - Involves the SIM, the MS, the MSC, and the AuC
- AuC and SIM have knowledge of IMSI and Ki as part of the device registration process
 - When the MS connects to network, SIM shares IMSI info which is forwarded to AuC

GSM Authentication

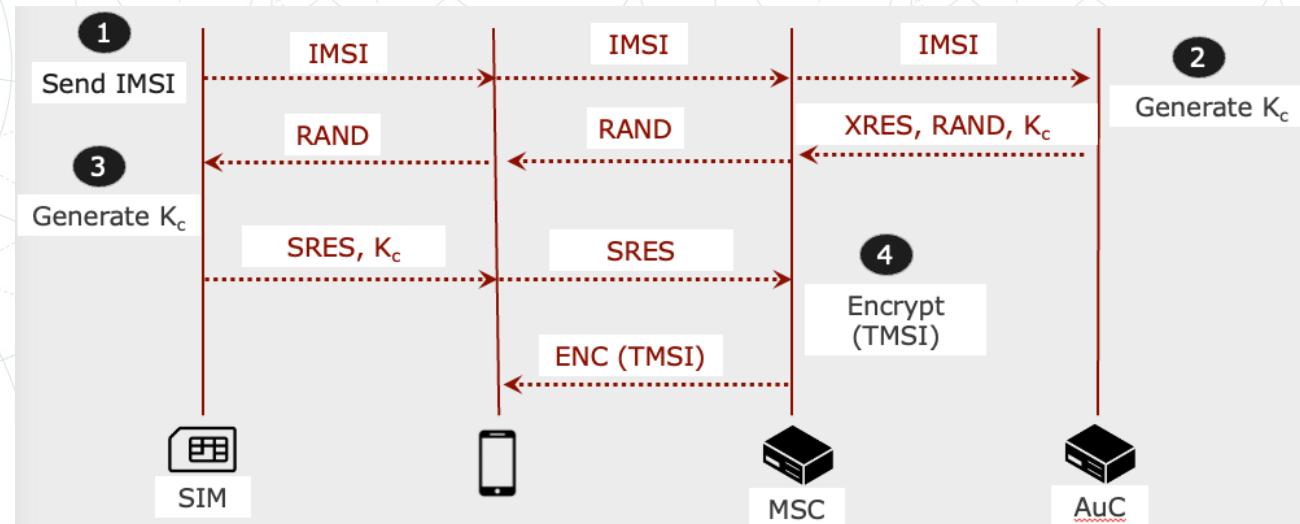
- The AuC retrieves K_i linked to IMSI and selects a random value RAND
- The RAND is used with two algorithms, A3 and A8, with the subscriber key K_i to generate the temporary cipher key K_c and the expected response (XRES)
- AuC shares K_c , RAND and XRES with the MSC, ending its role in the authentication process

GSM Authentication

- Next, MSC shares the RAND with the MS, which sends it to the SIM
 - Similar to AuC, the SIM uses the RAND to generate Kc and signed response SRES
 - SRES is delivered to MS, which forwards it over the air interface to the MSC for validation

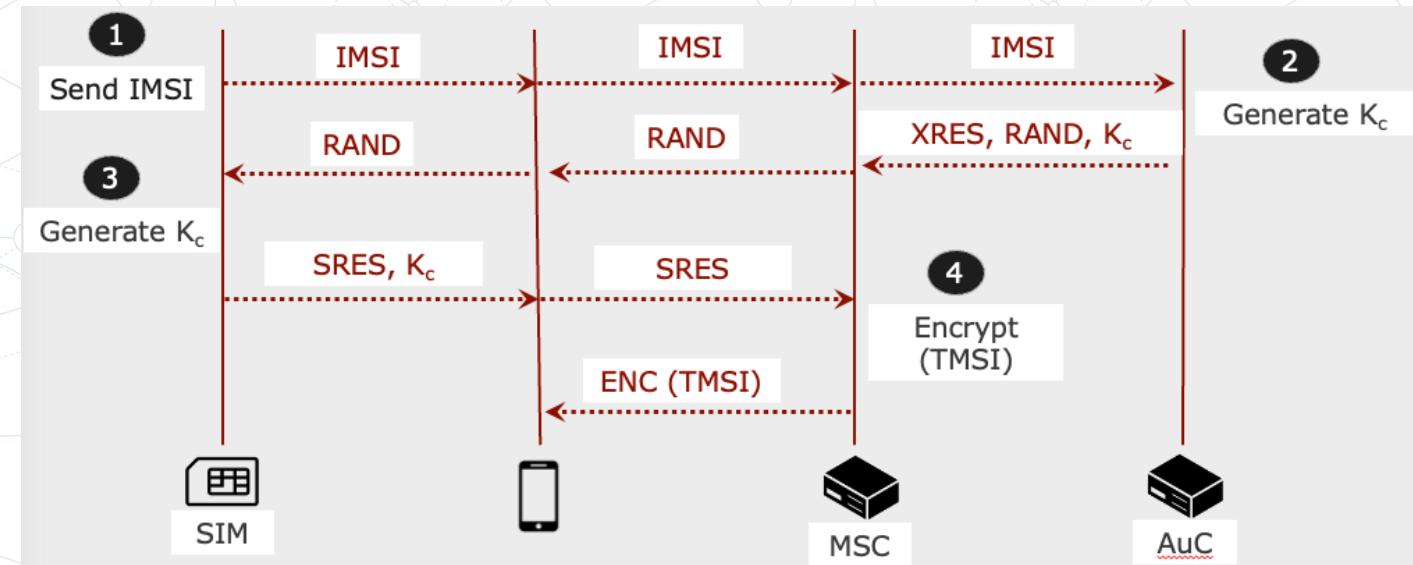
GSM Authentication

- The MSC compares SRES to XRES;
 - If match, validating the ME's identity
 - Then, MSC generates and encrypts TMSI for the ME to use and delivers it over the air interface



GSM Authentication: Vulnerability

- SIM card (and so the ME) is authenticated to the AuC
 - Prevent unauthorized devices accessing network services
- The authentication exchange does not validate the identity of the provider to the ME

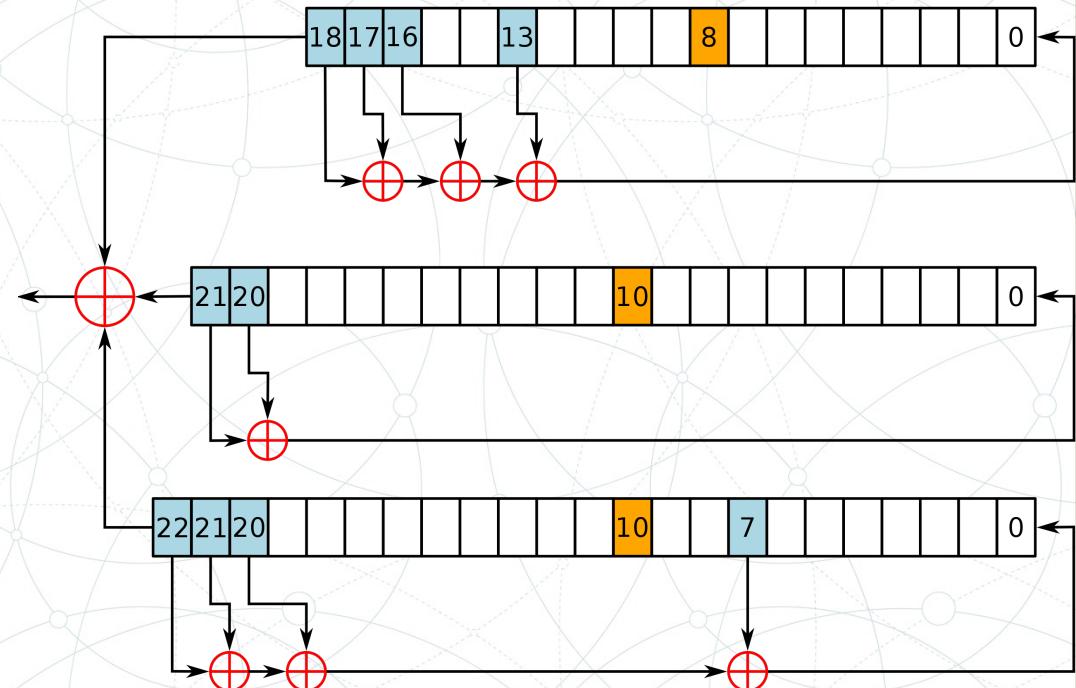


GSM Encryption

- Use the A5/1 cipher to provide confidentiality controls of traffic delivered over the GSM air interface between MS and BSC
- A5/1 is a stream cipher implemented using a Linear Feedback Shift Register (LFSR) mechanism
- Temporary cipher key K_c is used as the A5/1 input to generate keystream data
 - Ease of implementation in hardware and reduced implementation cost

A5 / 1

- Developed in 1987 (before GSM)
- 114 bits of GSM are XORed with 114 bits of Keystream material to produce encrypted data
- Uses three Linear Feedback Shift Registers (LFSR)



GSM Encryption

- The plaintext data is XOR'd with the keystream data to generate ciphertext
- Ciphertext is similarly XOR'd with matching keystream data at the recipient to decrypt

GSM Eavesdropping

- Commercial tools are expensive
 - But, you can build your own GSM sniffer using inexpensive hardware
 - For ex. RTL-SDR
- GSM packet captures disclose basic information about the network, but do not reveal the contents of phone call/SMS etc. due to being encrypted

A5/1 Key Recovery

- Precomputed reference attack for full key recovery
- In 2008, gsm-tvoid; keystream data to known keystream
state information in lookup tables
 - Using set of precomputed 288 quadrillion possible entries (apprx 2tb storage), adversary recovers K_i in approx. 30mins
 - It was taken offline without explanation
 - Possible government intervention

A5/1 Key Recovery

- In 2009, A5/1 cracking project reproduced the work previously published
 - As a plus, they distribute key recovery lookup tables through peer-to-peer networks to prevent them from being taken offline
- In 2011, ‘Kraken’, practical tool that integrates with AirProbe for effective capture and decryption of GSM traffic with the A5/1 tables
 - Later, ‘Pytacle’ tool improved features of the tool by adding play back capabilities for full and simple passive GSM decryption attack

<https://github.com/0xh4di/kraken>

Femtocell

- Extend the carrier network, leveraging the consumer's broadband connection for uplink connectivity
- Femtocell devices (e.g., Home NodeB or HNB) allow consumers to establish a relatively short-range extension of the carrier network that provides similar connectivity services (e.g., voice, data, SMS / MMS)
 - Also offers attackers new opportunities to attack the carrier infrastructure, as well as User Equipment devices

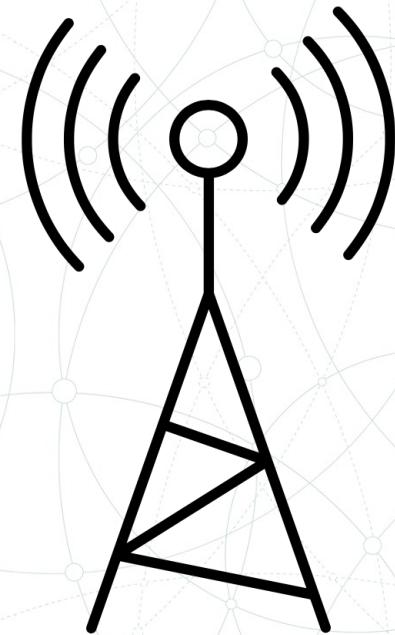
Femtocell Attack

- HNB is authorized device on the carrier network and has access to dynamic key information used to encrypt/decrypt the 3G connection
 - MiTM to manipulate and intercept phone calls
- They found a way to have root access to femtocell device and run their codes in them to sniff the traffic
 - Presented at Defcon 21

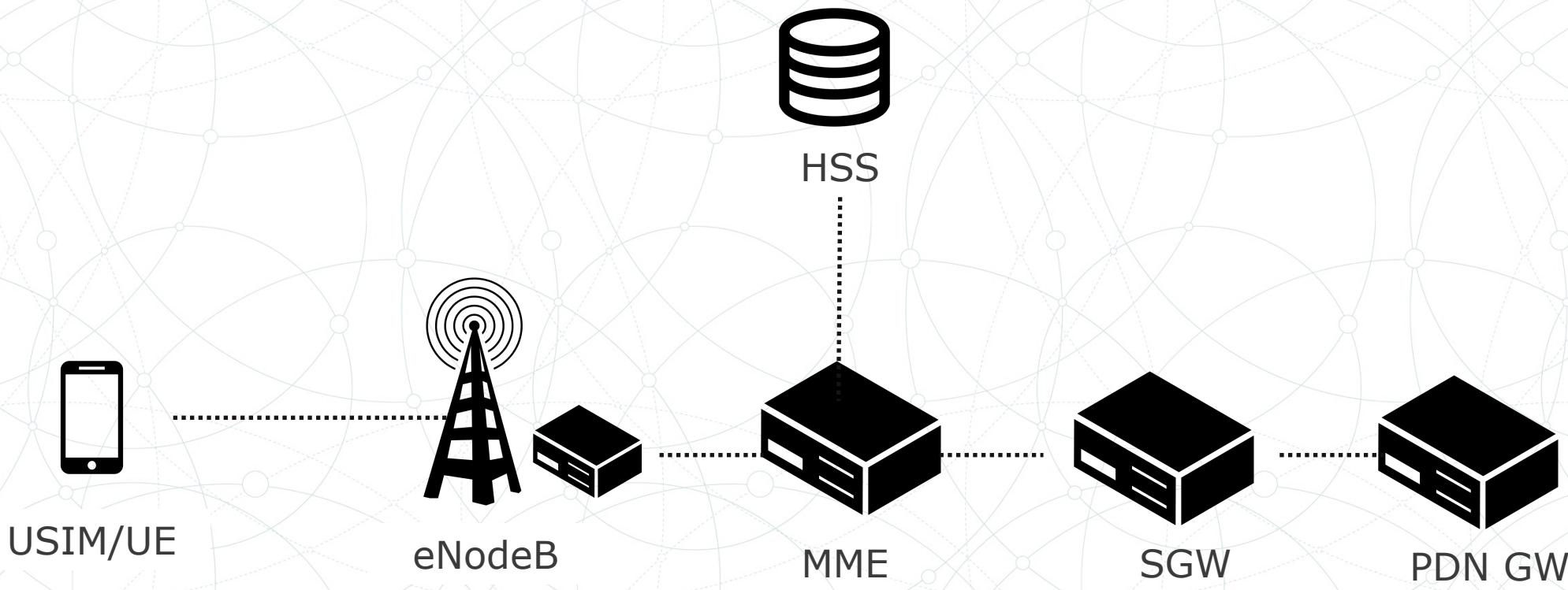
<https://www.youtube.com/watch?v=gfcq8clu1RI>

4G / LTE

- Long-Term Evaluation (LTE) Protocol
- Predecessor to GSM
- Marketed as 4G LTE or Advanced 4G
- In addition to higher speeds
 - Offers improvements for privacy
 - Introduces new encryption schemes

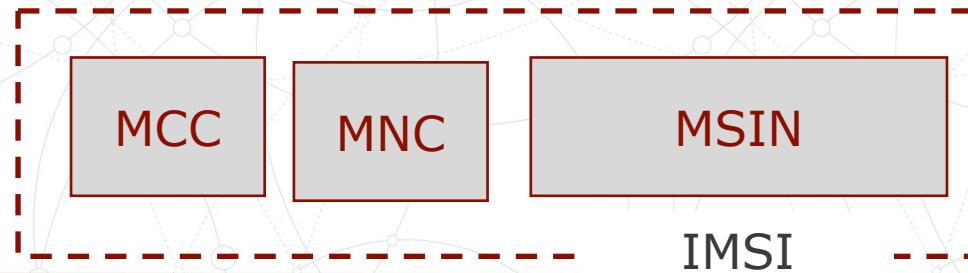


LTE Network Model



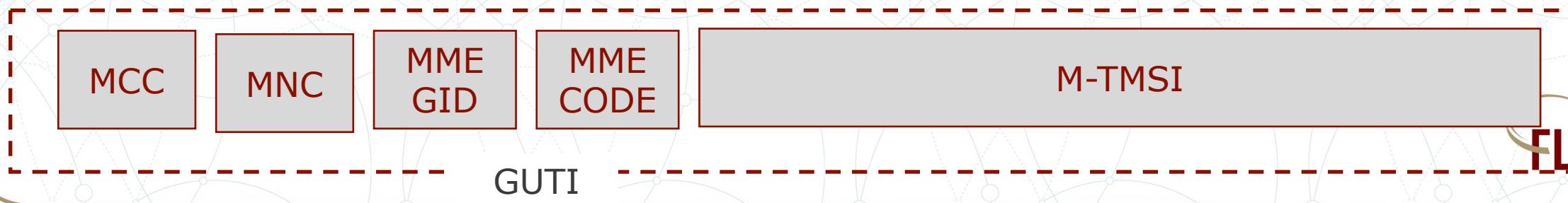
LTE Addressing Scheme for IMSI

- IMSI is made up of three components:
 - MCC: Mobile Country Code, identifying the country of the end-user
 - MNC: Mobile Network Code, identifying the home network
 - MSIN: Mobile Subscriber Identification Number, identifying the user within MCC and MNC context



LTE Addressing Scheme for IMSI

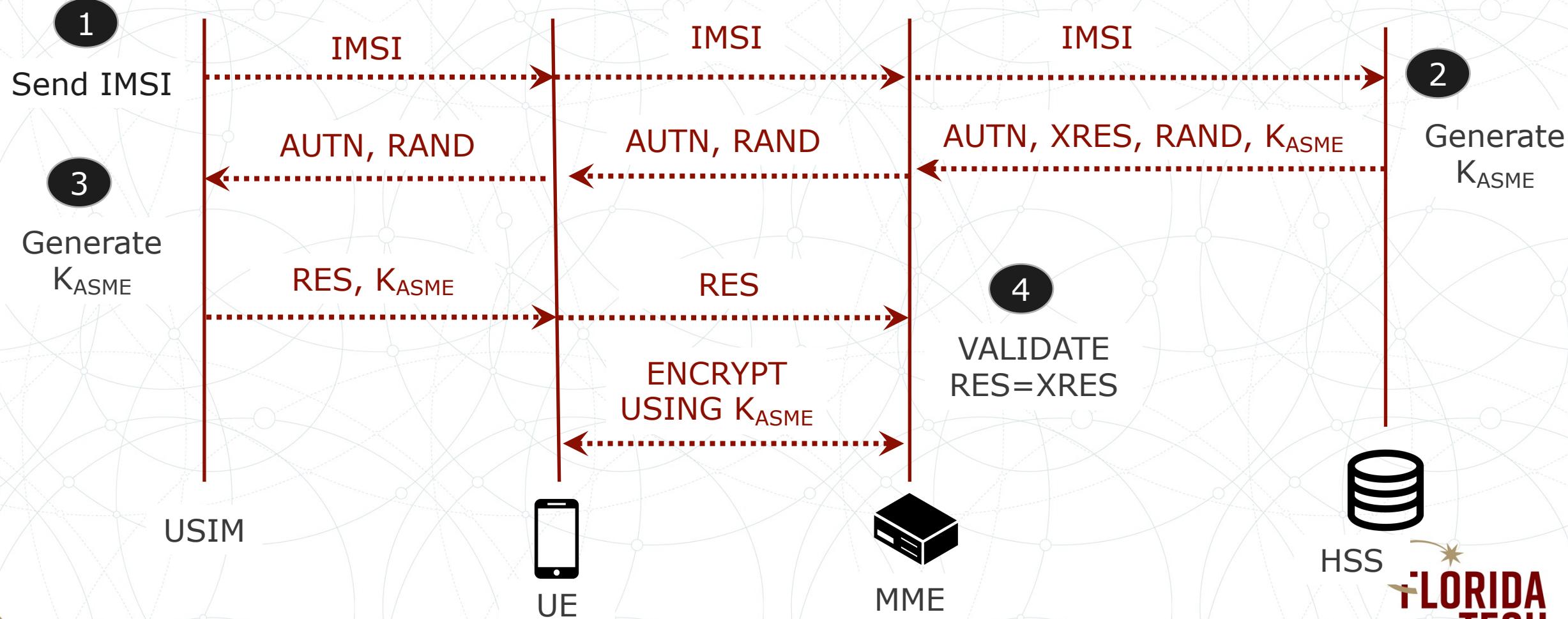
- IMSI value is stored on the USIM and a fixed value
 - Acts as a shared identifier for UE (e.g., LTE phone) and the HSS for the associated authentication key "K"
- To ensure privacy of the unique handset (such as identifying an IMSI to an individual), LTE introduces a Globally Unique Temporary ID (GUTI) which consists of the MCC, MNC MME info and the Temporary Mobile Subscriber ID (TMSI)



LTE Authentication

- Mutual auth of the handset and the network infrastructure through the Evolved Packet System Authentication and Key Agreement (EPS-AKA)
- Similar to GSM/3G, authentication in LTE relies on the identification function and shared key content provided by the IMSI (International Mobile Subscriber Identity)

LTE Authentication



LTE Authentication Vulnerability

- The IMSI is sent in plaintext
 - Rogue LTE network can get IMSI
 - Privacy threat to IMSI
- Yet, the secret K never is disclosed to the UE from USIM, preventing rogue applications from stealing the value and limiting attacker's ability to clone the value onto another USIM

LTE Encryption

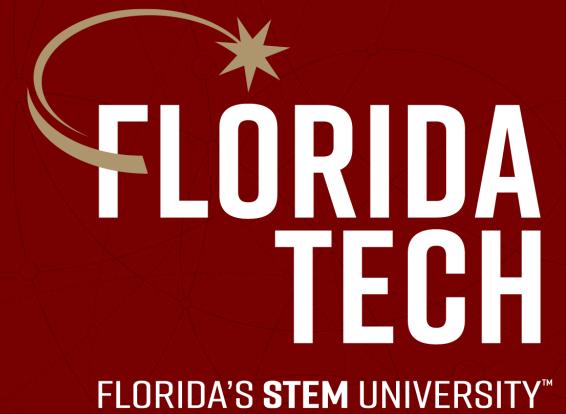
- LTE supports algorithm flexibility
- 3GPP systems were limited to a handful algorithms and these could not be replaced without changes to the network infrastructure
- Yet, LTE networks could adapt to new algorithm option to mitigate any flaw
 - Let's say there is a flaw found in AES

LTE Supported Encryption Algorithms

- NULL Algorithm:
 - Does not provide confidentiality of network traffic
 - In some cases, need to provide service outweighs the desire for security in LTE
 - Provides network access for devices lacking USIM card for situations such as emergency services (e.g., 911 in US)
 - May create opportunity for attacker to impersonate a legitimate carrier network without the need for cryptographic attacks

LTE Encryption Algorithm Tradeoffs

| Scheme | Advantages | Disadvantages |
|----------|---|---|
| Kasumi | Offers strong encryption via 128-bit keys Optimized for hardware implementation Offers resistance to block cipher attacks | Vulnerable to algebraic attacks |
| SNOW 3G | Fits 3G security requirements Offers protection against algebraic attacks | Computationally complicated |
| Milenage | Fits 3G security requirements Offers strong encryption via 128-bit keys Protects against side-channel attacks | Does not require standard algorithm Some interoperability issues |
| ZUC | Fits 3G security requirements Offers strong encryption via 128-bit keys Built on sound design principles | Still under scrutiny |



**Thank you.
Questions?**

Dr. Abdullah Aydeger