Department of Computer Science

FLORIDA TECH
FLORIDA'S STEM UNIVERSITY™

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

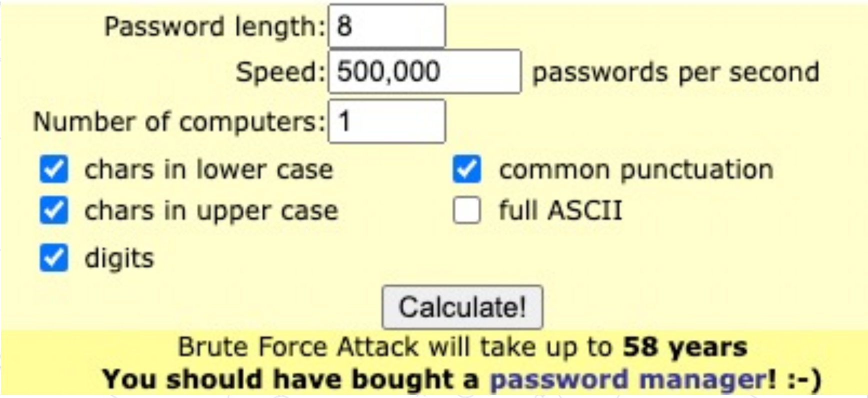
- However, authentication to an AP is conducted through management frames; meaning, if an attacker can capture the four-way handshake, they will have access to all factors which generated the PTK
  - Isolating the wireless password as the missing variable
- To crack the password, loop this pseudo-random 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

FLORIDA TECH

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

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

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
Device 2 (slave)	1	8	5	4	7	6	10	2	9	12	3	11

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

- 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

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

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
Device 2 (slave)	1	8	5	4	7	6	10	2	9	12	3	11

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

- 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

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

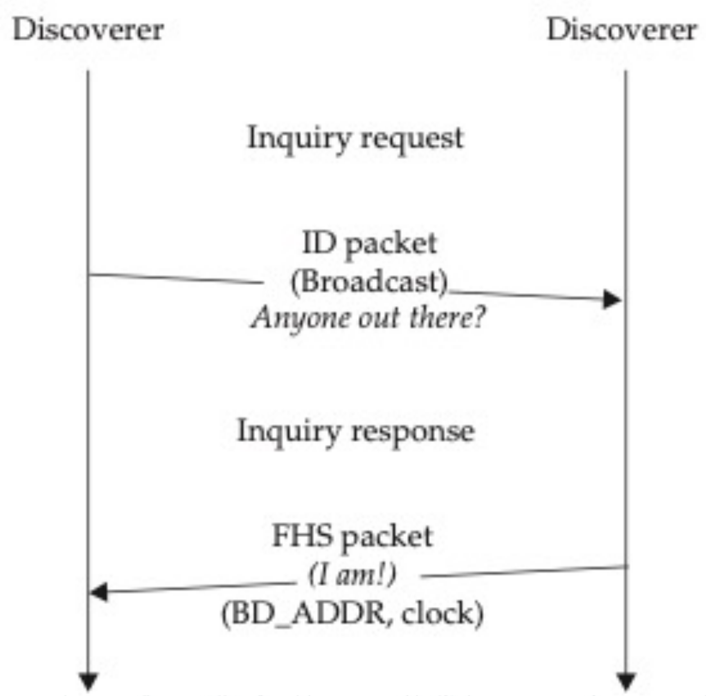
- Technically, discoverable devices are devices that enter the <u>inquiry scan substate</u>
  - These devices <u>respond to inquiry requests</u>
- On the other end of this frequency-hopping dance is the device doing the discovery
  - This <u>device has no knowledge</u> of its potential peer channels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

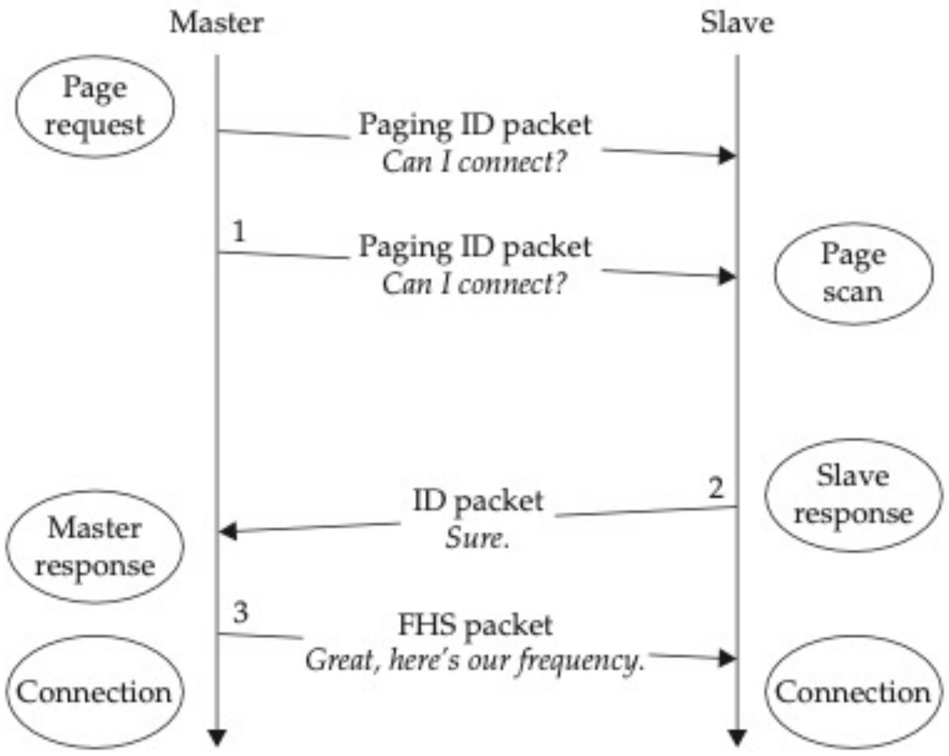


Bluetooth: Connection Establishment

- 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 <u>must know the target's BD_ADDR</u>, and that device must be interested in accepting connections

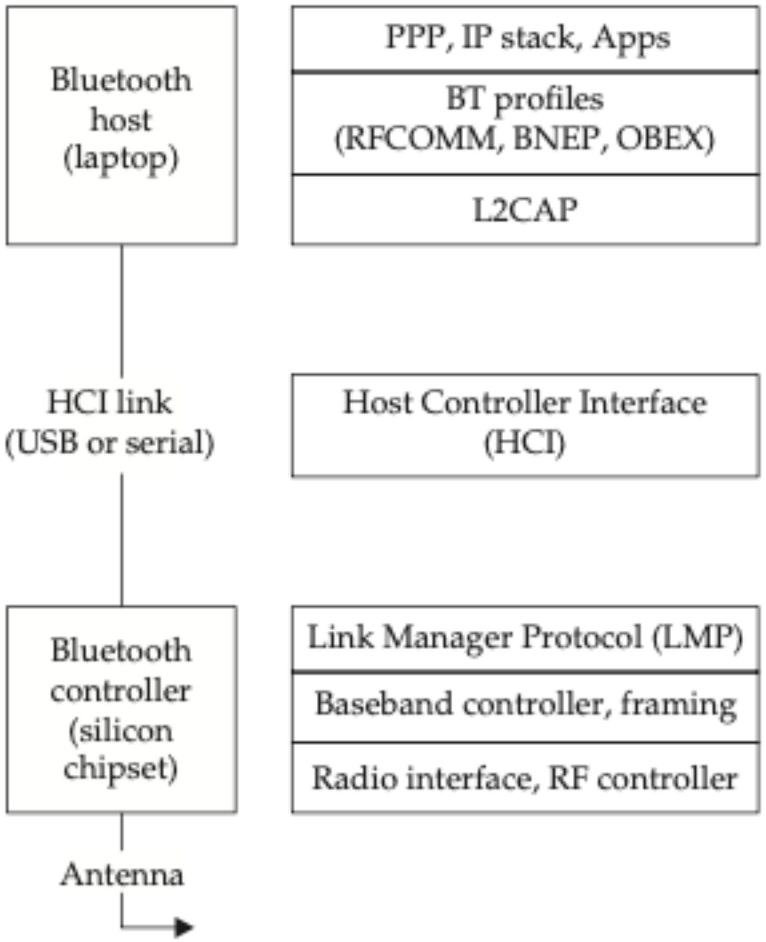


Bluetooth Protocol Overview

- 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 <u>similar to that of UDP</u> (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 <u>line between IP and UDP</u>
  - Usually L2CAP is used to carry <u>higher-level data packets</u>; however, on some platforms, an application programmer can make use of it directly

Host Controller Interface (HCI)

- HCI is a protocol that has no allegory 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 <u>Bluetooth connections need no modification at all</u>, <u>even</u> when a completely <u>different Bluetooth chipset</u> (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 <u>outside the capability of users</u>

Baseband

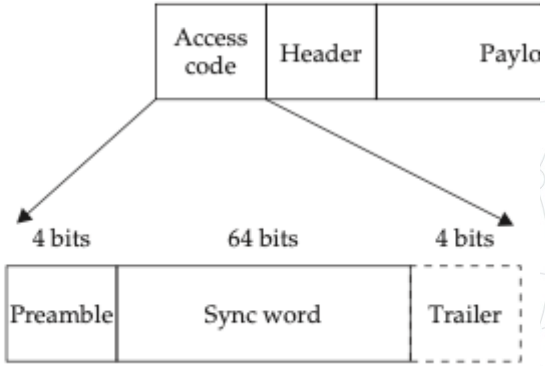
- The Bluetooth baseband specifies the over-the-air characteristics (such as the transmission rate) and the final layer of framing for a packet
- Unlike 802.11, where receiving all the packets on a channel is trivial, actually getting a packet with in-tact baseband headers out of the controller and into 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

Access code	Header	Payload

- 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 <u>examine the access code</u> 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

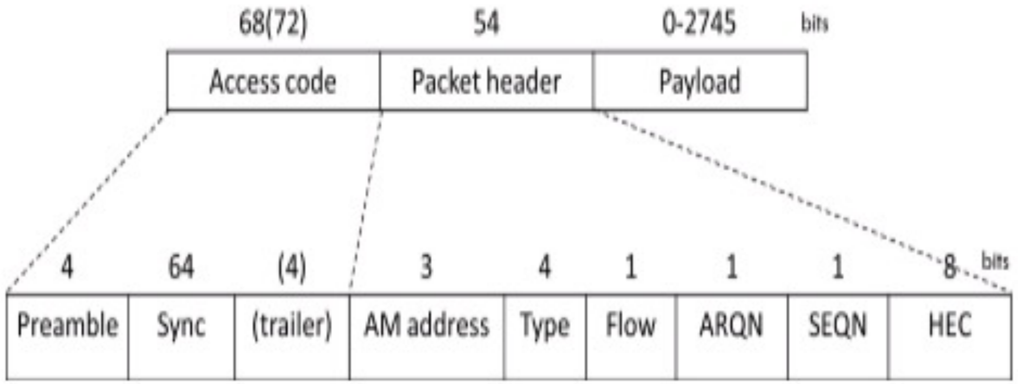
- At any given point in time, a Bluetooth controller will be interested in only a handful of sync words
  - Any packets received by the controller with sync words that aren't interesting won't be passed through the HCI link
    - These packets are assumed to be for another piconet
  - At any given time,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



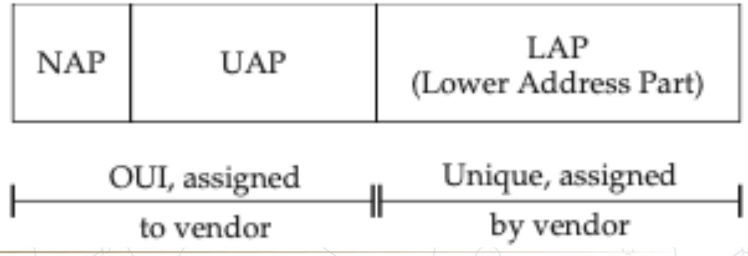
- Most of these fields aren’t of concern to us

unless we are implementing our own Bluetooth controller in software:

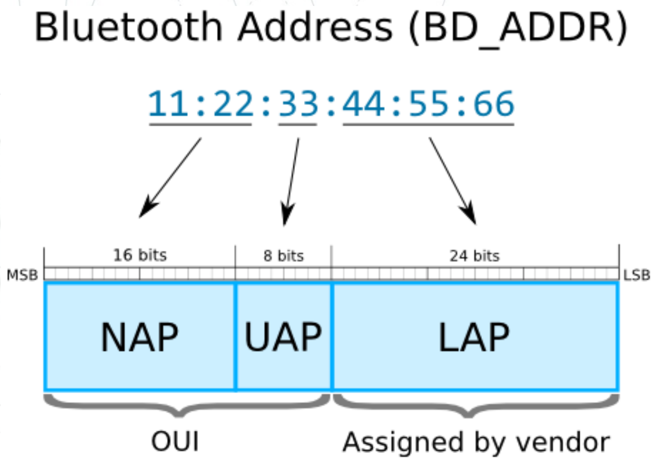
- AM Address (or LT_ADDR); Logical Transport Address
- Type; the type of packet being used, indicating the data type (ACL or SCO)
- Flow; a simple flow-control feature
    - When set to 1 (known as GO), the receiver has sufficient buffering space; 0 implies the opposite
- ARQN or sequence bit (SEQN); 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



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

FLORIDA TECH

FLORIDA'S STEM UNIVERSITY™