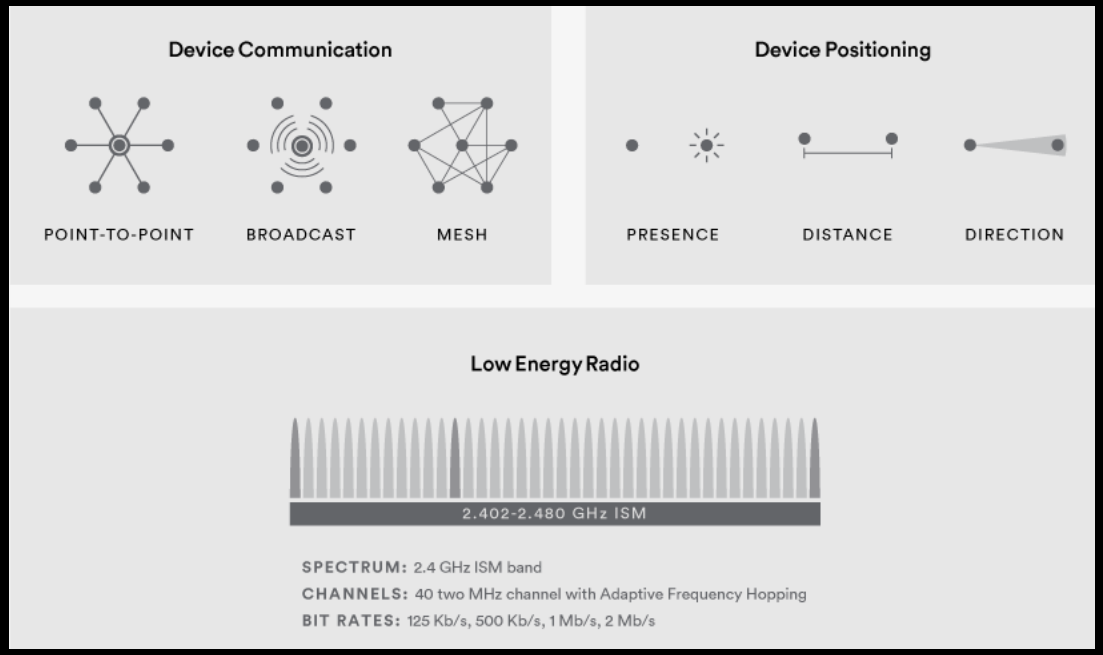
BLUETOOTH LOW ENERGY

Dino López

BLUETOOTH 4.0

- Bluetooth Low Energy (BLE) makes part of the
  - Bluetooth 4.0 standard
  - A low-power solution to control and monitor applications
- BLE is a single-hop solution
  - No routing services
  - Different to ZigBee
- Bluetooth Low Energy is not compatible with Bluetooth 3.0, nor compatible with 2.0

BLUETOOTH 4 AND LATER

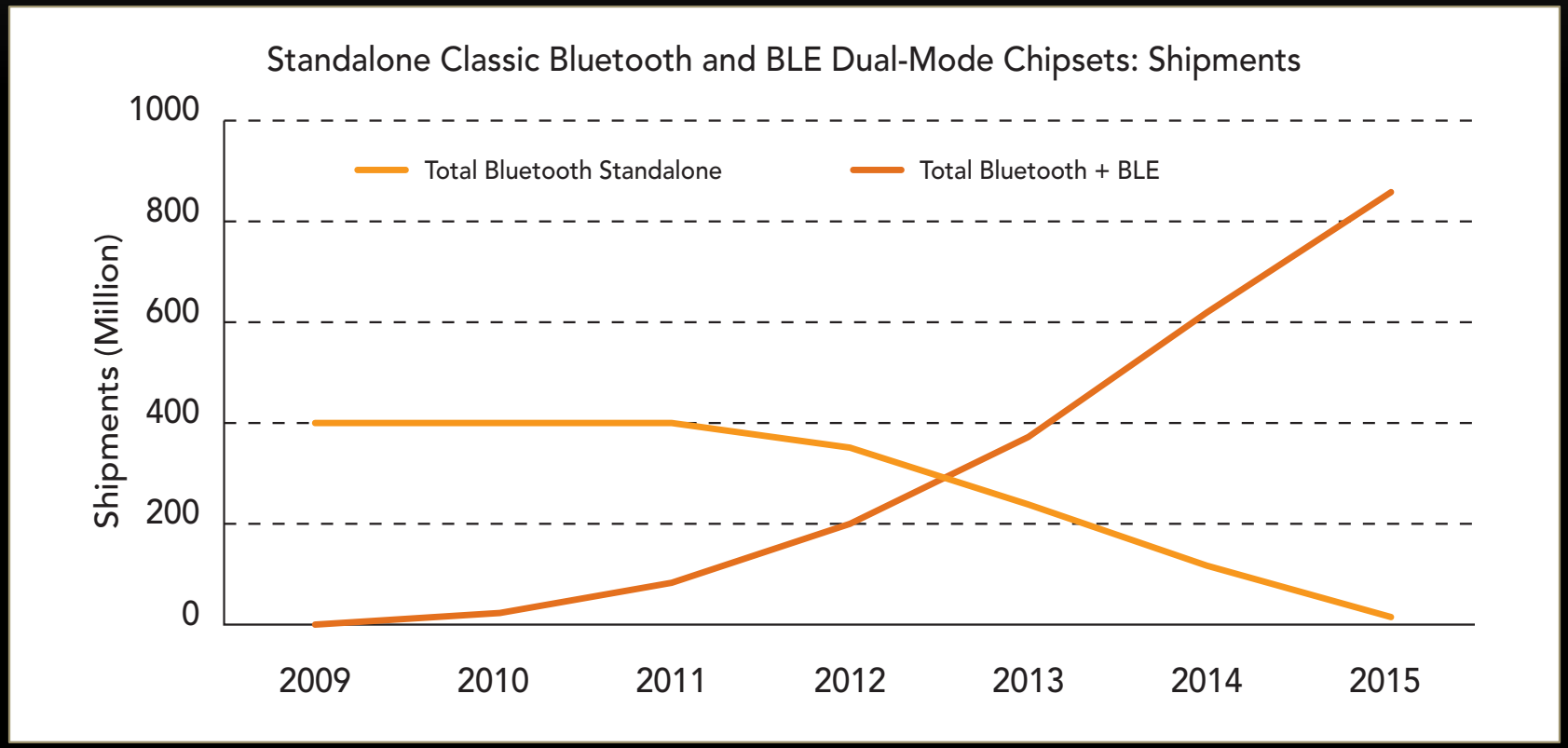


From https://www.bluetooth.com/wp-content/uploads/2021/01/Bluetooth_Technology_Overview_Graphic.png?time=1645218476

BLUETOOTH LOW ENERGY VS BLUETOOTH CLASSIC

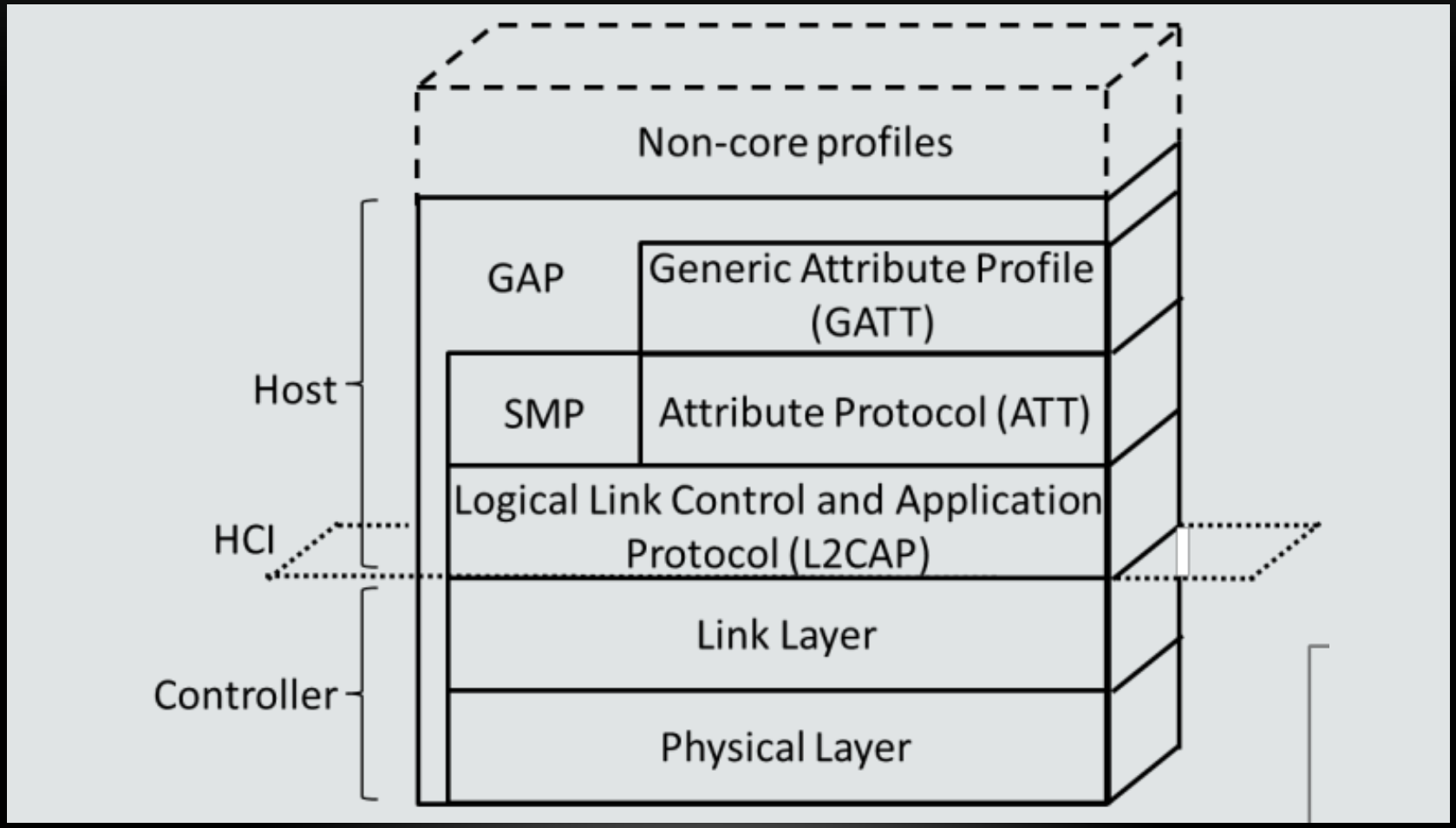
Technical specification	Classic Bluetooth	Bluetooth Low Energy
Frequency	2400 to 2483.5 MHz	2400 to 2483.5 MHz
Number of Channels	79	40
Channel Bandwidth	1 Mhz	2 MHz
Nominal Data Rate	1-3 Mbps	1 Mbps
Application Throughput	0.7-2.1 Mbps	< 0.3Mbps
Nodes / Active Slaves	7	Unlimited
Voice	Capable	Not Capable
Time to send data	100ms	3-6 ms
Peak Current Consumption	<30mA	<15mA (Av. 12µA – 1s connInterval)
Latency (connection)	~100ms	6 ms

THE BLE PROMISING MARKET



From http://litepoint.com/whitepaper/Bluetooth%20Low%20Energy_WhitePaper.pdf

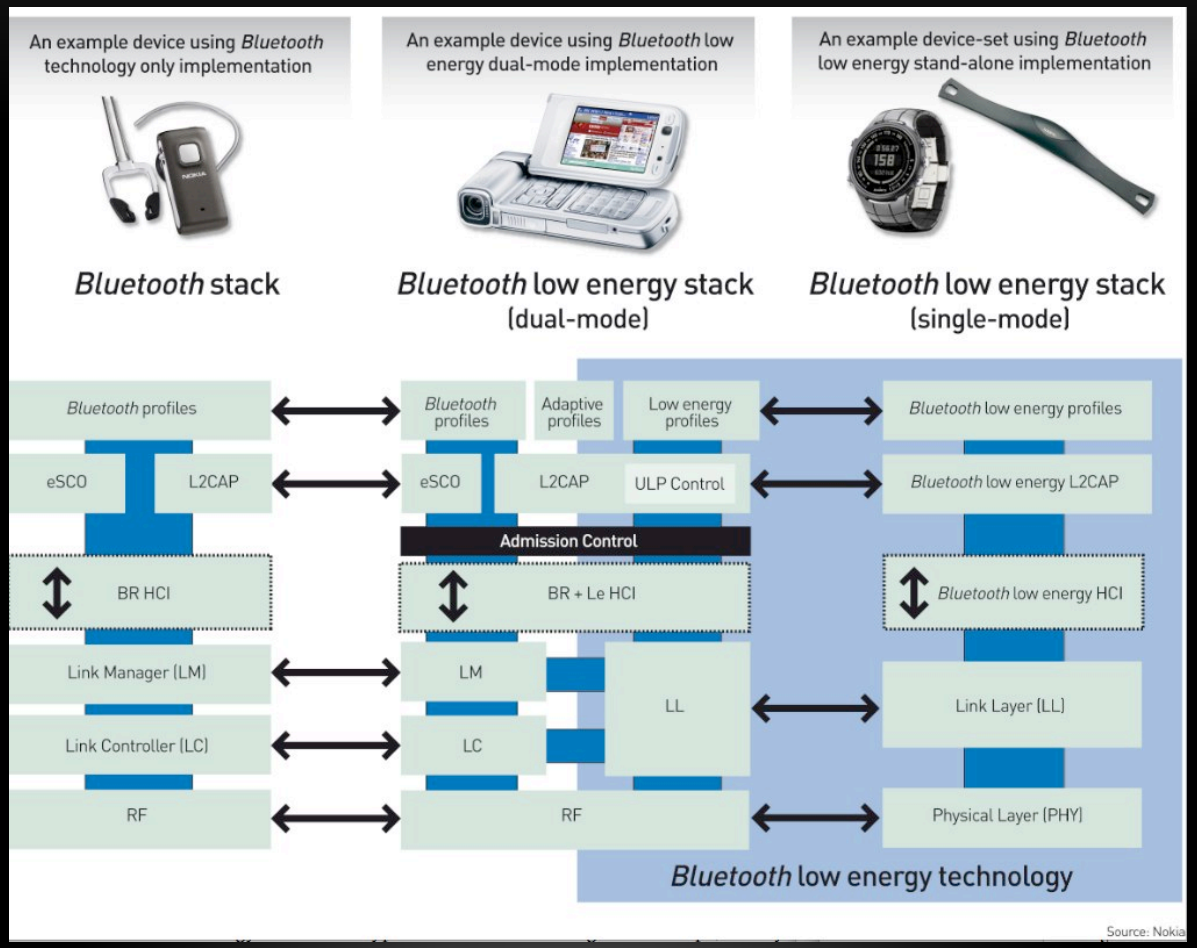
BLE PROTOCOL ARCHITECTURE



BLE PROTOCOL ARCHITECTURE

- BLE is composed of two main parts, the Host and the Controller, interconnected through the HCI
- The Controller features the Physical Layer and the Link Layer
- The Host part features the Generic Access Profile, Generic Attribute Profile, the Security Manager Protocol, the Attribute Protocol and the Logical Link Control and Application Protocol
- Dual-mode devices: devices provdigin both the BLE and the Classic Bluetooth protocol architecture
- Single-mode devices: devices providing only BLE

SINGLE-MODE AND DUAL-MODE DEVICES



PHYSICAL LAYER

- BLE operates at the 2.4 GHz band and defines 40 channels, spaced by 2MHz each one.
- There are 2 types of RF channels
  - 3 Advertising channels: device discovery, connection stablishment, broadcast transmission
  - 37 Data channels: bidirectionnal communications between connected devices
- The advertising channels are strategically placed to minimize the overlapping with the 1, 6 and 11, 802.11 channels
- A relaxed Frequency Hopping scheme is used to avoid interference with other technologies working in the 2.4GHz band
- Bandwidth equal to 1MHz

802.11 AND BLE COHABITATION

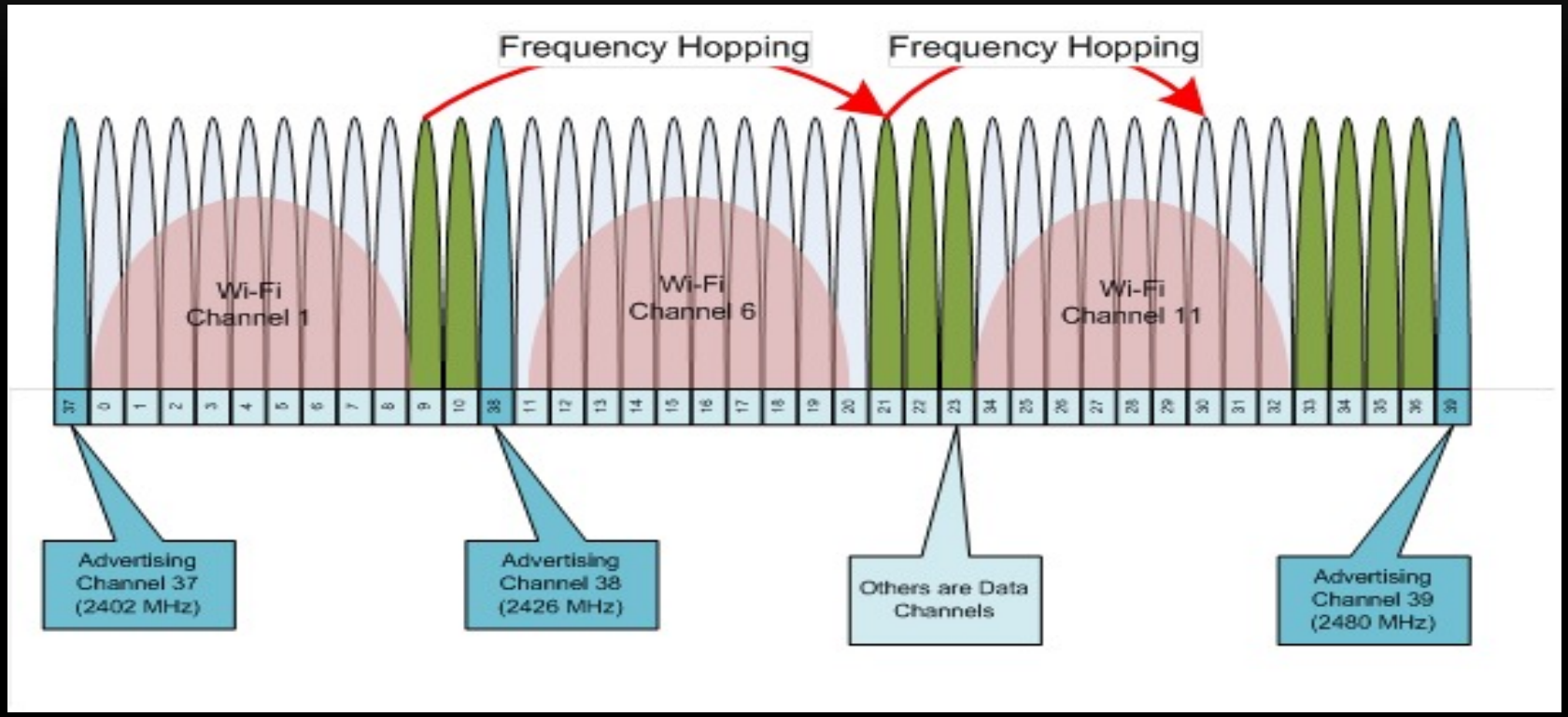


Image source: http://www.element14.com/community/groups/wireless/blog/2013/08/23/bluetooth-low-energy

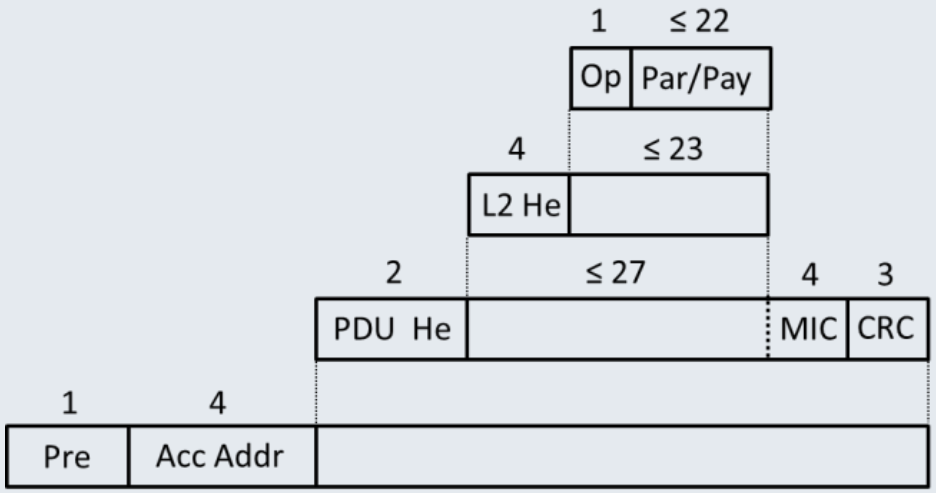
LINK LAYER

- Devices broadcast data through advertising channels, in advertising packets.
- Any device sending adversiting packets are called « advertisers »
- The tranmission of advertising packets are donne by intervals of times, called advertising events.
- Devices that receive data through advertising channels only are called « scanners »
- Bidirectionnal communication between devices requires a connection

CONNECTTION STABLISHMENT

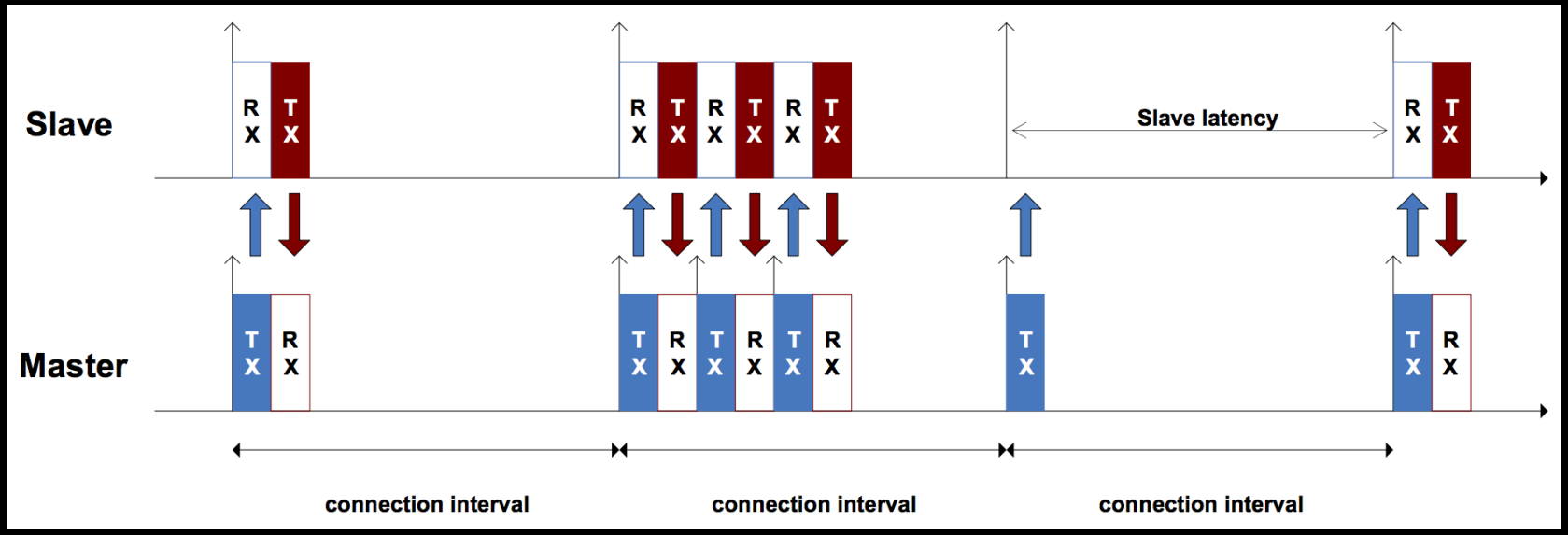
- One device must be in Advertising mode, and another one in Initiator mode
- An advertiser (that will become the slave) announces that it is a connectable device through the advertising channels
- An initiator (that will be the master), which listen for that advertising packets, can transmit a Connection Request to the advertiser
- A point-to-point connection is then created.
- Any packet from that connection will be identified by a 32-bits access code randomly generated
- One slave can connect to only one master. The master can connect to multiple slaves
- A piconet is thus composed by one master with its several slaves
- One device can belong to a single piconet

PACKET FORMAT



Par/Pay:	Parameters and Payload
Op:	ATT Opcode
PDU He:	PDU Header
L2 He:	L2CAP Header
Acc Addr:	Access Address
Pre:	Preamble
MIC:	Message Integrity Check
CRC:	Cyclic Redundancy Check

MASTER – SLAVE COMMUNICATION



MASTER – SLAVE COMMUNICATION

- Once a connection is created, the physical channel is divided into non-overlapping connection events intervals
- The Master is always the first one to send a packet in a connection event
- The slave is required to reply to each master packets. However, the master can or not reply to the slaves'packets.
- An IFS of at least 150 µs between to consecutives transmission is required
- A device having more data to send, must enable the More Data bit in the data packets
- When none of the devices has more data to send, the connection event is closed. The slave can turn its antena off and turn it on at the start of the next connection event
- A new connection event starts in a new physical channel
- The slave is allowed to sleep after connSlaveLatency consecutive connections (so don't reply back to the master)

L2CAP

- Multiplexe between the Attribut Protocol, Security Manager and Link Layer
- Simplified version of the L2CAP Classic Bluetooth stack.
  - No flow control
  - No retransmission
  - No segmentation/reassembly
- Maximum PDU of 23 bytes

ATT & GATT

- ATT (Attribute Protocol)
  - Defines the client-server architecture
  - Expose data as attributes
- GATT (Generic Attribute Profile)
  - Framework using ATT to provide services
  - Attribute to describe the general service
  - Attribute to retrieve the sensed data

ABOUT GATT

- In BLE, GATT servers maintain a DB with the attributes that describe the provided services
- A GATT DB implements one or more profiles. Each profile is made up of one or more services, and each service is composed of one or more characteristics
- Every BLE device acting as a GATT server must implement the Generic Access service (UUID = 0x1800), and two mandatory characteristics: Device Name (UUID = 0x2A00) and Appearance (UUID = 0x2A01).
- One characteristic may possess several attributes
  - Frequently, characteristics are also called attributes
- Characteristics are associated to a handle, so services are described by a group of handlers
- Standardized services use 16-bits UUIDs. No standardized services use 128-bits UUIDs

ConHnd	Handle	Uuid	Uuid Description	Value	Value Description	Properties
0x0000	0x0001	0x2800	GATT Primary Service Declaration	00:18		
0x0000	0x0002	0x2803	GATT Characteristic Declaration	02:03:00:00:2A		Rd 0x02
0x0000	0x0003	0x2A00	Device Name	Simple BLE Peripheral		Rd 0x02
0x0000	0x0004	0x2803	GATT Characteristic Declaration	02:05:00:01:2A		Rd 0x02
0x0000	0x0005	0x2A01	Appearance	00:00		Rd 0x02
0x0000	0x0006	0x2803	GATT Characteristic Declaration	02:07:00:04:2A		Rd 0x02
0x0000	0x0007	0x2A04	Peripheral Preferred Connection Parame...	50:00:A0:00:00:00:E8:03		Rd 0x02
0x0000	0x0008	0x2800	GATT Primary Service Declaration	01:18		
0x0000	0x0009	0x2800	GATT Primary Service Declaration	0A:18		
0x0000	0x000A	0x2803	GATT Characteristic Declaration	02:0B:00:23:2A		Rd 0x02
0x0000	0x000B	0x2A23	System ID	88:A9:08:00:00:0B:C9:68		Rd 0x02
0x0000	0x000C	0x2803	GATT Characteristic Declaration	02:0D:00:24:2A		Rd 0x02
0x0000	0x000D	0x2A24	Model Number String	Model Number		Rd 0x02
0x0000	0x000E	0x2803	GATT Characteristic Declaration	02:0F:00:25:2A		Rd 0x02
0x0000	0x000F	0x2A25	Serial Number String	Serial Number		Rd 0x02
0x0000	0x0010	0x2803	GATT Characteristic Declaration	02:11:00:26:2A		Rd 0x02
0x0000	0x0011	0x2A26	Firmware Revision String	Firmware Revision		Rd 0x02
0x0000	0x0012	0x2803	GATT Characteristic Declaration	02:13:00:27:2A		Rd 0x02
0x0000	0x0013	0x2A27	Hardware Revision String	Hardware Revision		Rd 0x02
0x0000	0x0014	0x2803	GATT Characteristic Declaration	02:15:00:28:2A		Rd 0x02
0x0000	0x0015	0x2A28	Software Revision String	Software Revision		Rd 0x02
0x0000	0x0016	0x2803	GATT Characteristic Declaration	02:17:00:29:2A		Rd 0x02
0x0000	0x0017	0x2A29	Manufacturer Name String	Manufacturer Name		Rd 0x02
0x0000	0x0018	0x2803	GATT Characteristic Declaration	02:19:00:2A:2A		Rd 0x02
0x0000	0x0019	0x2A2A	IEEE 11073-20601 Regulatory Certificati...	FE:00:65:78:70:65:72:69:6...		Rd 0x02
0x0000	0x001A	0x2803	GATT Characteristic Declaration	02:1B:00:50:2A		Rd 0x02
0x0000	0x001B	0x2A50	PnP ID	01:0D:00:00:00:10:01		Rd 0x02
0x0000	0x001C	0x2800	GATT Primary Service Declaration	5D:FE		
0x0000	0x001D	0x2803	GATT Characteristic Declaration	0A:1E:00:F1:FF		Rd Wr 0x0A
0x0000	0x001E	0xFFF1	Simple Profile Char 1	01		Rd Wr 0x0A
0x0000	0x001F	0x2901	Characteristic User Description	Characteristic 1		
0x0000	0x0020	0x2803	GATT Characteristic Declaration	02:21:00:F2:FF		Rd 0x02
0x0000	0x0021	0xFFF2	Simple Profile Char 2	02		Rd 0x02
0x0000	0x0022	0x2901	Characteristic User Description	Characteristic 2		
0x0000	0x0023	0x2803	GATT Characteristic Declaration	08:24:00:F3:FF		Wr 0x08
0x0000	0x0024	0xFFF3	Simple Profile Char 3			Wr 0x08
0x0000	0x0025	0x2901	Characteristic User Description	Characteristic 3		
0x0000	0x0026	0x2803	GATT Characteristic Declaration	10:27:00:F4:FF		Nfy 0x10

http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_40_00_45/docs/blestack/ble_user_guide/html/ble-stack-3.x/gatt.html#

BLUETOOTH SECURITY

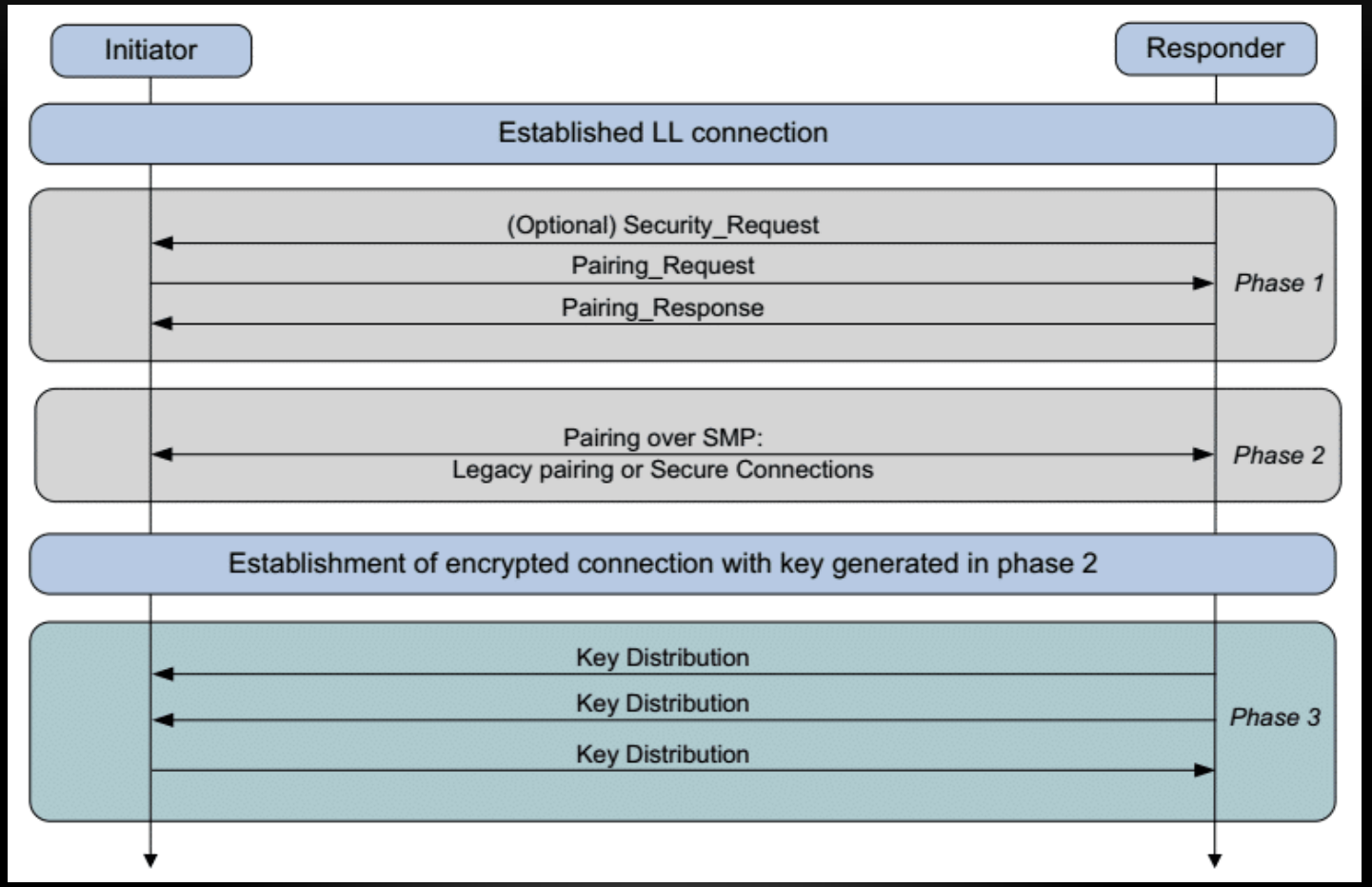
SECURITY FEATURES

- Encrypted links
  - Passive eavesdropping
- Authentication
  - Man-In-The-Middle
- Different security levels
  - Non-sensitive data -> Unprotected exchanges (Level 1 security)
    - SDP protocol
    - L2CAP Eco-request/replies
  - Sensitive data -> (Levels 2, 3 and 4)
    - Needs pairing

PAIRING METHODS

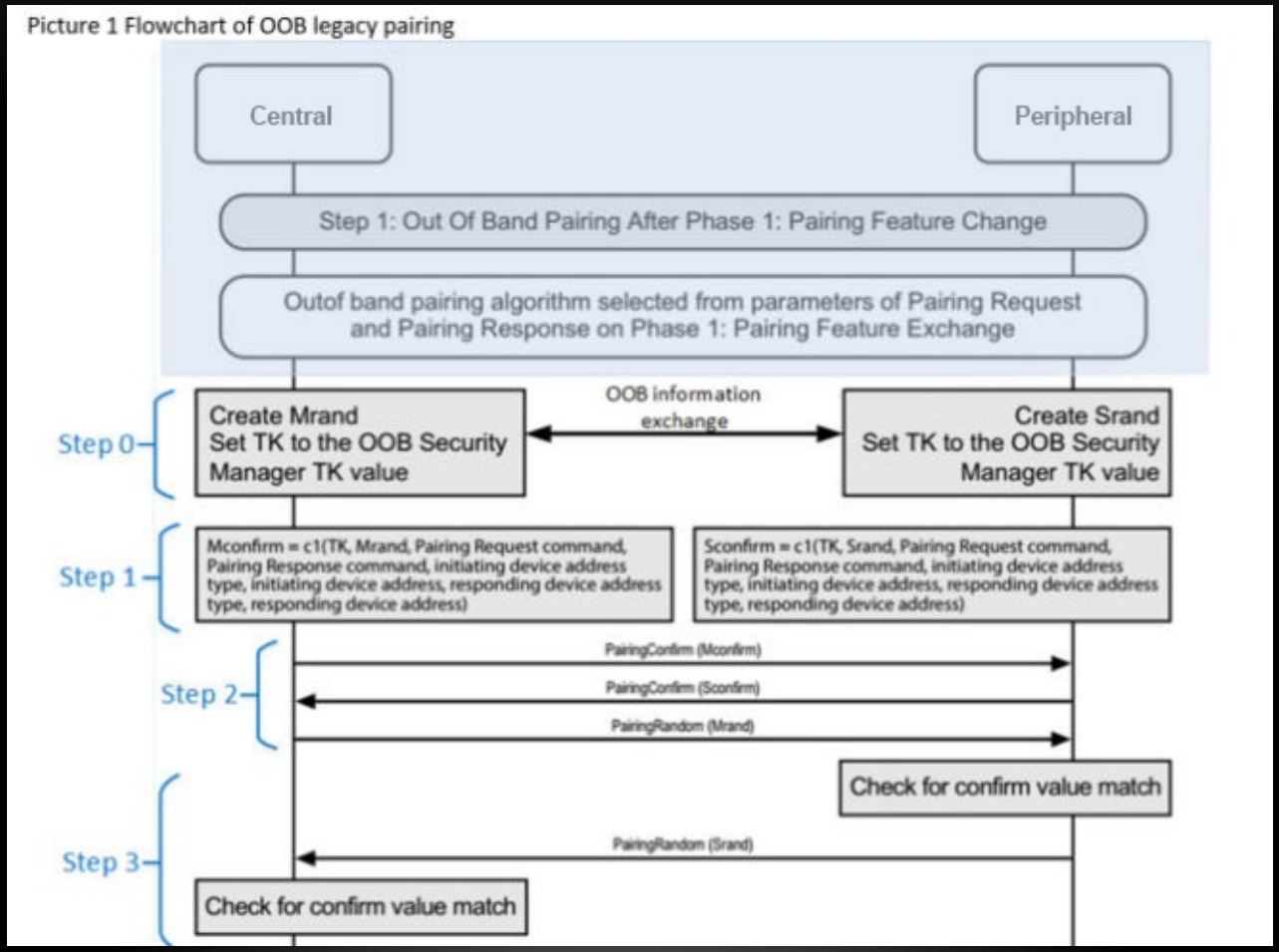
- Just Works
  - Short Temporary Key (STK) on data exchanged on plain text
  - No authentication
- Passkey entry
  - A six-digit passkey is required in one or both sides
- Out-Of-Band (OOB)
  - Rely on other technologies to exchange data for a key generation (e.g. NFC)
- Numeric Comparison (only for LE secure connections)
  - Elliptic curve Diffie–Hellman (ECDH) for key generation
  - Six-digit passkey verification (yes/no)

PAIRING PROCESS



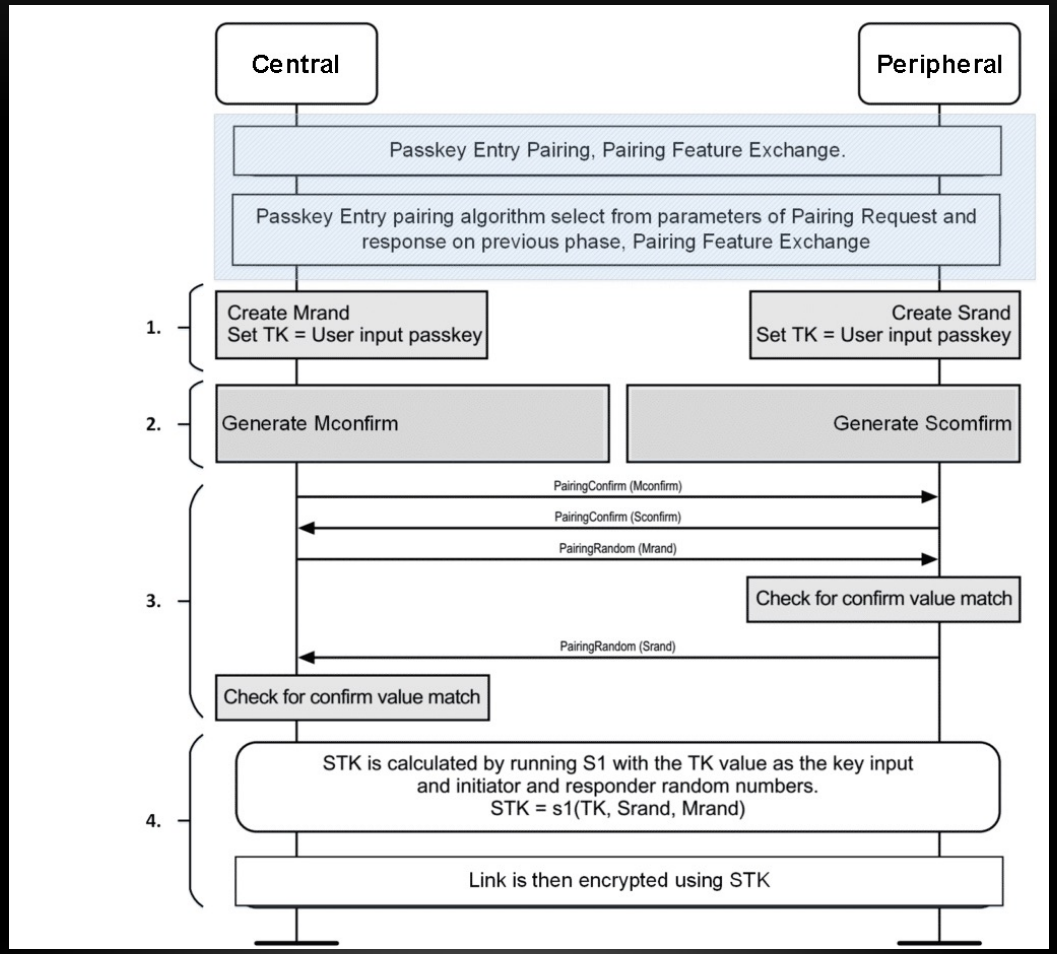
From https://www.bluetooth.com/blog/bluetooth-pairing-part-1-pairing-feature-exchange/

LEGACY PAIRING & OOB



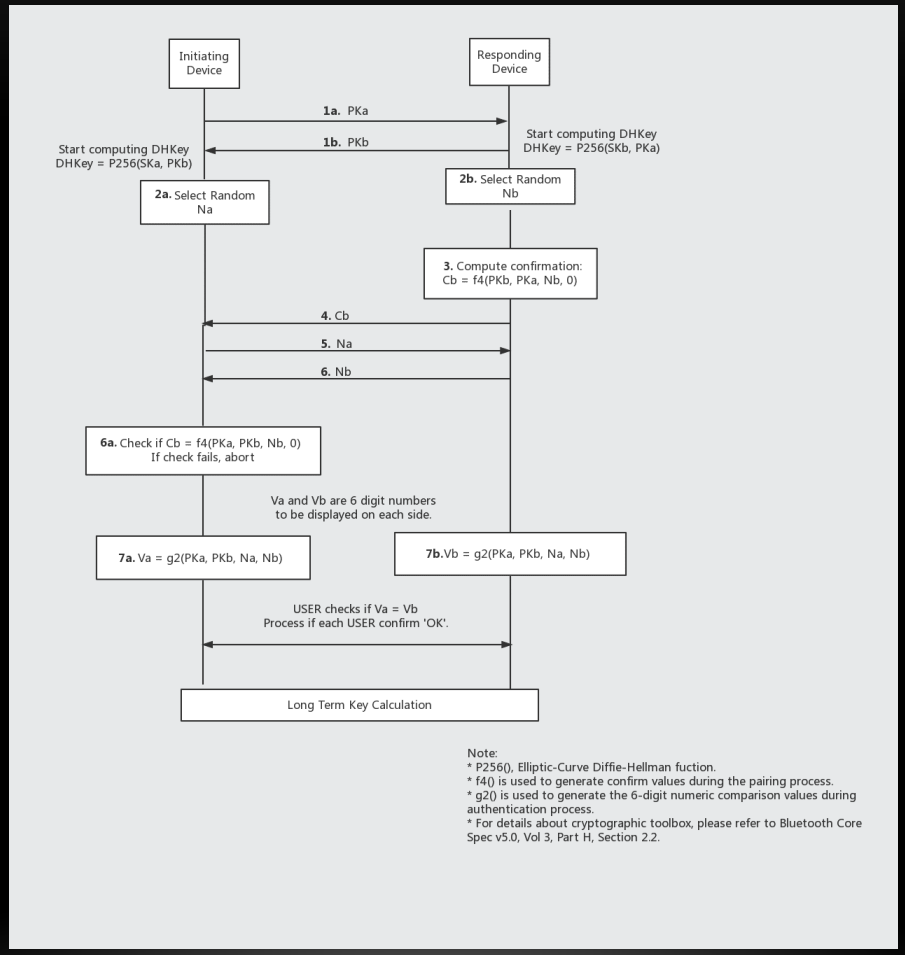
From https://www.bluetooth.com/blog/bluetooth-pairing-part-5-legacy-pairing-out-of-band/

PASSKEY ENTRY



From https://www.bluetooth.com/blog/bluetooth-pairing-passkey-entry/

NUMERIC COMPARISON



From https://www.bluetooth.com/blog/bluetooth-pairing-part-4/

SOME ATTACKS

- Old famous Bluejacking
- PIN – Link cracking
  - Sniffing -> needs costly hardware
  - BlueBorne -> connection without authentication - 2017
  - KNOB -> brut force key cracking (in real-time) - 2019
- CDV (2018) -> DoS
  - Connection Dumping Vulnerability
  - Queen's University
  - No need of special hardware, nor special software
  - Vulnerability found in various devices from different vendors

BLE IN LINUX (TESTED IN UBUNTU 15.10)

- BLE devices can be listed with the hcitool command
- To scan for BLE devices, you will issue the « hcitool lescan » command
- To access the GATT server, you can use gatttool
  - Interactive session "gatttool -i hci0 -b 00:01:02:03:04:05 -I”
  - To stablish a connection, issue the ”connect” command
  - To obtain the list of services, execute ”primary” from the gatttool session
  - To list the characteristics of a service, query the attribute group of your interest with ”char-desc 0x0000 0x000a”, assuming 0x0000 to be the starting attribute, and 0x000a the end group attribute
  - To read the value provided by a characteristic, use « char-read-hnd 0x0021 », assuming 0x0021 is the attribute handler of interest
  - To write to an attribute, use “char-write-req” indicating the handler attribute, followed by the value in hexadecimal format, with the least significant octet first (Little endian)

TOOLS (UBUNTU 15.10)

- If you need a VM, install VirtualBox, the Official and latest stable version
  - https://www.virtualbox.org/wiki/Linux_Downloads
- With APT
  - build-essential, python-bluez, bluez-*, pkg-config, libboost-python-dev, libboost-thread-dev, libbluetooth-dev, libglib2.0-dev, python-pip, python-gobject
- m-a prepare
- If VM, then install Vbox Guest Additions
  - shared folders
  - usb controller
- pip install gattlib