All BLE guides are wrong
including this one
David Burnett, UC Berkeley, 2018 Mar 06
db@berkeley.edu

Preamble (1 byte)
Access address (4 bytes)
Header (2 bytes)
Advertising address (6 bytes)
Payload (0 to 31 bytes)
CRC (3 bytes)
All BLE guides are wrong
including this one

Preamble (1 byte, depends on access address)
Access address (4 bytes, but write out all 4 bytes and reverse them)
Header (2 bytes, first 4 bits are reversed from spec table, length is LSB first)
Advertising address (6 bytes, but write out all 6 bytes and reverse them)
Payload (0 to 31 bytes, individual bytes are LSB first)
CRC (3 bytes, MSB first but packet sniffers all calculate hex codes as if it was LSB first)
(and the most popular graphic about CRC has the wrong LFSR init code)
All BLE guides are wrong
including this one

Preamble (1 byte)
Access address (4 bytes)
Whiten:
Header (2 bytes)
-- 4 bits denoting type of packet (LSB first, so backward wrt spec)
-- 1 bits reserved for future use
-- 1 bit that does something new in BLE 5, don’t remember what, set it to 0
-- 1 bit about Tx address
-- 1 bit about Rx address
-- 8 bits length (as of BLE 5; LSB first)
Advertising address (6 bytes)
Payload (0 to 31 bytes), can contain multiple sub-payloads each with:
-- 1 byte length
-- 1 byte type, listed in “Supplement to the Bluetooth Core Specification”
-- N bytes data
CRC (3 bytes)
-- LFSR initialized to 0x555555
Preamble

• If first bit of access address 0
  • Preamble is 01010101

• If first bit of access address 1
  • Preamble is 10101010
Access address

>> dec2bin(hex2dec('8E89BED6'),32)
10001110100010011011111011010110
Access address

```bash
>> dec2bin(hex2dec('8E89BED6'),32)
10001110100010011011111011010110

preamble: 10101010

total: [10101010 10001110100010011011111011010110]
```
Access address

>> dec2bin(hex2dec('8E89BED6'),32)
10001110100010011011111011010110

preamble: 10101010

total: [10101010 10001110100010011011111011010110]

nope -- access address is transmitted LSB first
see BT core spec 5.0 Vol 6, Part B, section 1.2 (page 2,555)

>> fliplr(dec2bin(hex2dec('8E89BED6'),32))
01101011011111011001000101110001

preamble: 01010101

total: [01010101 01101011011111011001000101110001] ^first out of ant       last out of ant^
Some reserved for future use (RFU) sections are in use for BLE 5.
• We used ADV_NONCONN_IND for our example packet
  • Transmitted LSB-first: 0100 in order of radio transmission
Header: RFU, TxAdd, RxAdd

- Set both RFU bits to 0
- One of them has a new use in BT 5.0

- TxAdd, RxAdd:
  - 0 means public address, 1 means randomized address
  - Set both to 0 for this example
Header: length

- Size of payload in bytes/octets
- Length word is 6 bits long in BLE 4, 8 bits in BLE 5
- Length will be least 6 for advertiser address
  - I used an address from a prior packet capture
- At most 37 (typically)
- We used 16 in our example
- Transmitted LSB first, so length 16 is 00001000 in order of transmission
BLE 4 header & payload

2 bits RFU in BT 4
Used for more length bits in BT 5

http://j2abro.blogspot.com/2014/06/understanding-bluetooth-advertising.html
PDU Type: 0100 (ADV_NONCONN_IND code expressed LSB-first)
RFU: 00
TxAdd: 0
RxAdd: 0
Length: 00001000 (0x10 expressed LSB-first)

Total header: 0100 00 0 0 00001000
^first out of antenna
Payload

- Advertiser address (6 bytes)
- 0 or more payload sections consisting of:
  - 1 byte length (not including self)
  - 1 byte GAP code describing this payload section
  - 0 to N bytes data
- For our example: 2 sections
  - First: 0x02 (length), 0x01 (“flags”), 0x05 (flag data)
  - See “Supplement to the Bluetooth Core Specification” part A section 1.3 at bluetooth.com/specifications/bluetooth-core-specification
  - Second: 0x06 (length), type 0x08 (“short name”), data 0x53 0x43 0x55 0x4D 0x33 (ASCII code for “SCUM3”)

Payload

• Advertiser address (6 bytes)
  • 0x90d7ebb19299, from TI packet capture*
  • Transmitted LSB first

  \[
  \gg \text{fliplr(dec2bin(hex2dec('90d7ebb19299'),48))}
  \]
  100110010100100110001101110101111110101100001001

• Data: transmitted LSB-first \textbf{per-byte}, e.g.,
  • 0x02 0x01 0x05 becomes
  • 01000000 10000000 10100000
  • ^ first bit out of antenna

**CRC**

- 24-bit linear feedback shift register (LFSR) in spec
- Initialized to 0x555555 as shown above
- Feed data in LSB-first at far right
- Input data is header, adv. address, payload(s)
- Result is transmitted **MSB-first** (position 23 above)
  - Wireshark reports CRC values as if they were LSB-first!
- Could be calculated with very few (1?) clock cycles

Note “Example CRC encoding...” figure is initialized to the wrong value; corrected figure shown here
<table>
<thead>
<tr>
<th>P.nbr.</th>
<th>Time (us)</th>
<th>Channel</th>
<th>Access Address</th>
<th>Adv PDU Type</th>
<th>Adv PDU Header</th>
<th>AdvA</th>
<th>AdvData</th>
<th>CRC</th>
<th>RSS (dBm)</th>
<th>FCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>246</td>
<td>+99995</td>
<td>0x25</td>
<td>0x8E8BED6</td>
<td>ADV_IND</td>
<td>0x907EBB19299</td>
<td>0x01830EA965F</td>
<td>0x90D7EBB19299</td>
<td>0xEF5DA8</td>
<td>-59</td>
<td>OK</td>
</tr>
<tr>
<td>247</td>
<td>367</td>
<td>0x25</td>
<td>0x8E8BED6</td>
<td>ADV_CONNECT_REQ</td>
<td>0x907EBB19299</td>
<td>0x90D7EBB19299</td>
<td>0xEF5DA8</td>
<td>-59</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>248</td>
<td>16575</td>
<td>0x06</td>
<td>0x6A9E2E6D</td>
<td>L2CAP-C</td>
<td>0x3A369</td>
<td>-30</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>249</td>
<td>231</td>
<td>0x06</td>
<td>0x6A9E2E6D</td>
<td>L2CAP-C</td>
<td>0x3A369</td>
<td>-30</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Capturing device: Advertising Channel: 37 (2402 MHz)
Connect to Initiator Address: 0x001830EA965F

Whitening

• Remove long strings of 0 or 1
  • Easier for the receiver to track center frequency and determine if bit is +df or -df

• 7-bit linear feedback shift register (LFSR) in spec

• Initialize to channel (37 in our example, not above) plus 1 added to LSB-side

• Feed in everything you fed into CRC, plus CRC, in transmit order

Note “Image illustrating logic inside the data whitening..” figure is not initialized to the claimed 19 value
Putting it all together

• See ble_adv_scm3.mat
Recording commercial packets

BLE USB dongle producing ad packet
OpenMote antenna attached to LNA
Not pictured: mixer w/ 2.400GHz from Rohde

IF on scope showing ad packets on each BLE ad channel: 2MHz, 26MHz, 80MHz pulses
Recording commercial packets

• hcitool:
  • linux app, only works with some BLE dongles
  • Can specify custom payload in hex
• Pre-preamble 000111
• Length is 37 even if payload isn’t
  • Remaining space is filled with extra sub-payload w/ 0x00 for length, 0x00 for type and garbage (?) data
Keysight waveform generator producing I/Q of packet Rohde combining & upconverting to BLE ch 37 (2.402GHz)

iphone w/ BLE scanner app
Test:
12:30pm