How Smart Is Bluetooth Smart?

Mike Ryan
iSEC Partners

ShmooCon 9

Feb 16, 2013

Slides and More Info

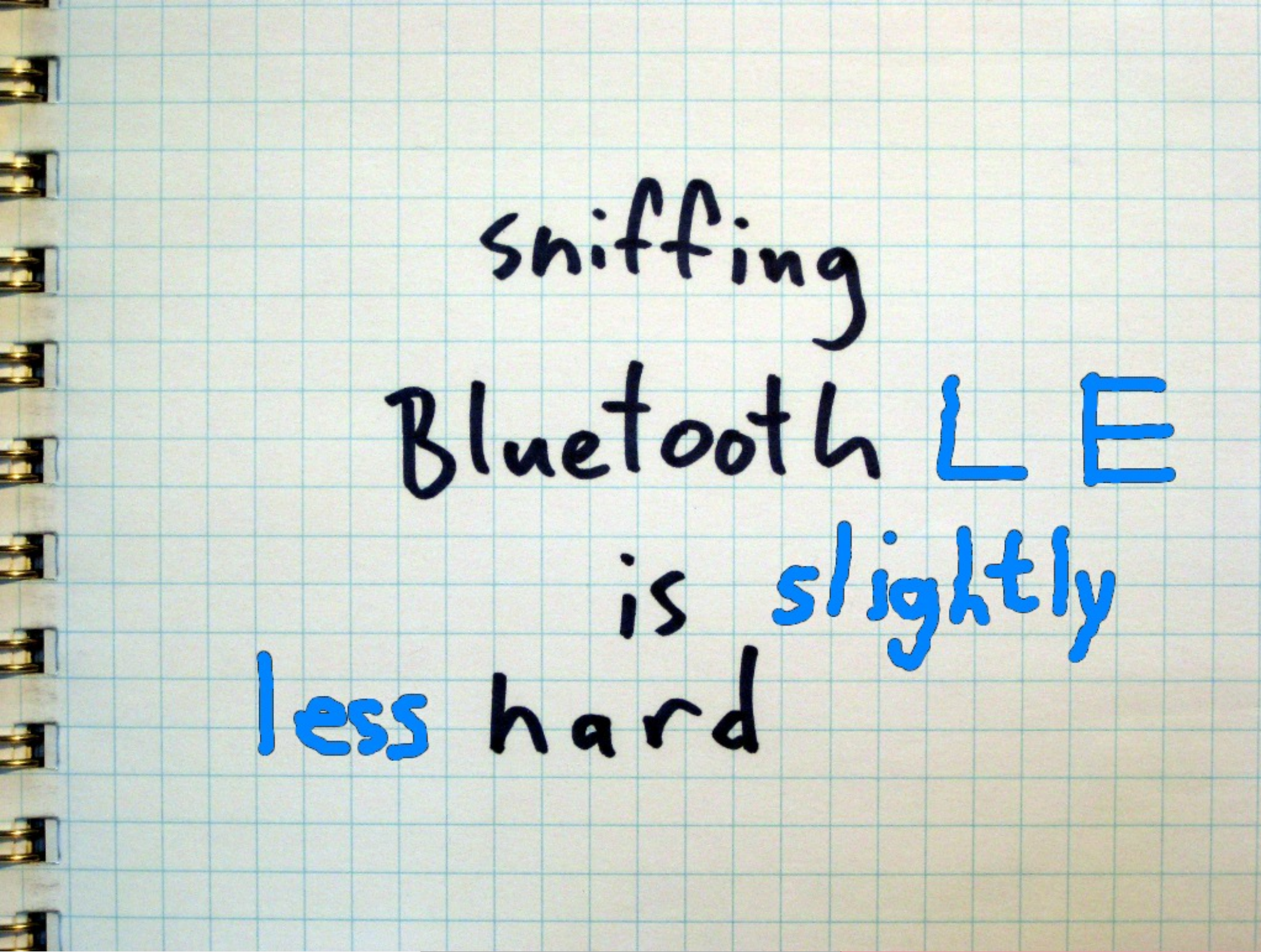
hmm,
seems legit

http://lacklustre.net/bluetooth/

Outline

- What is Bluetooth Smart / Low Energy / BTLE
- Cool Stuff
- More Cool Stuff
- Conclusion


sniffing
Bluetooth
is
hard


sniffing
Bluetooth LE
is slightly
less hard

What is Bluetooth Smart?

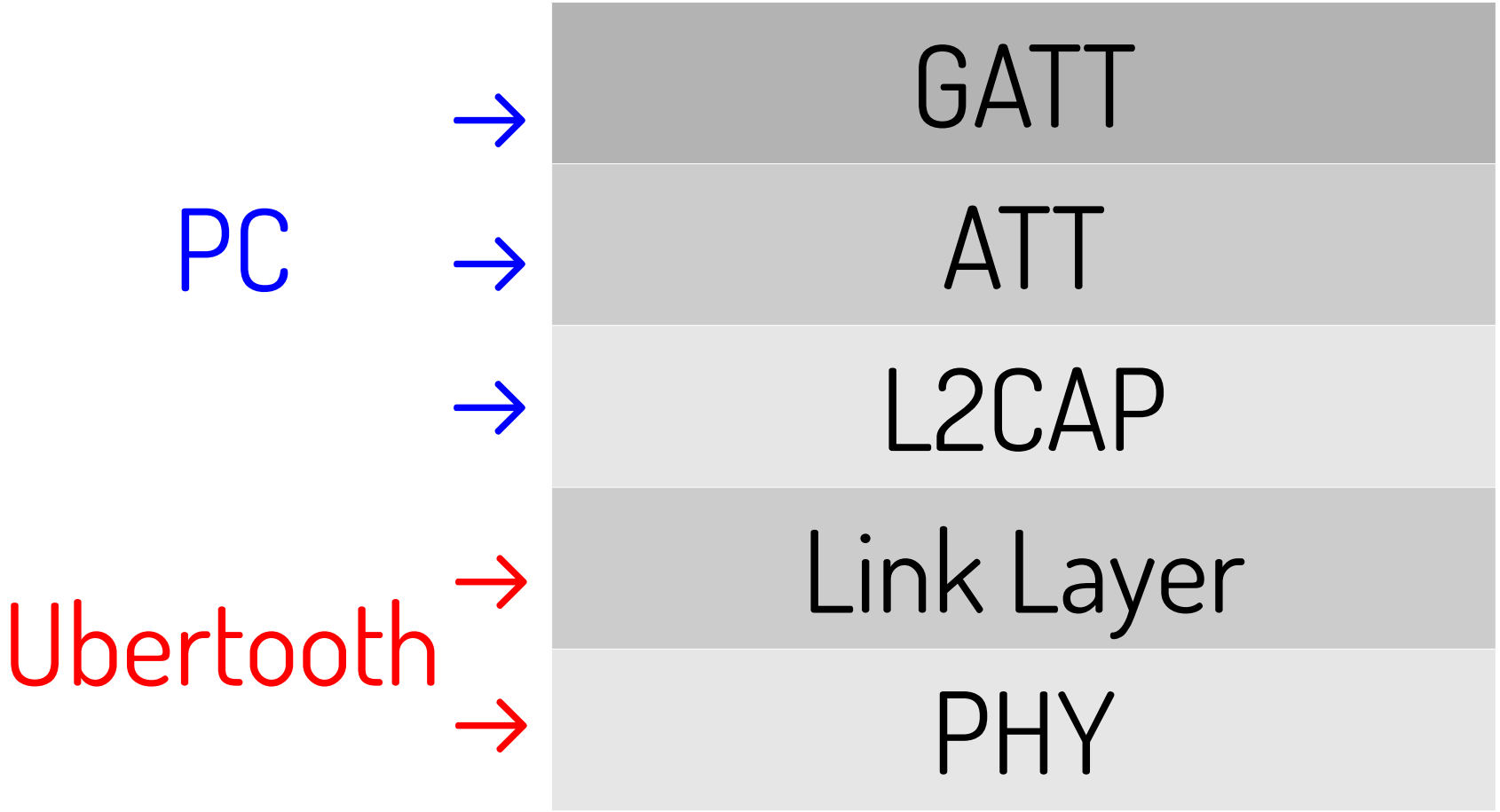
- New modulation and link layer for low-power devices
- Introduced in Bluetooth 4.0 (2010)
- AKA Bluetooth Low Energy / BTLE
- vs classic Bluetooth
  - Incompatible with classic Bluetooth devices
  - PHY and link layer almost completely different
  - High-level protocols the same (L2CAP, ATT)

Where is BTLE?

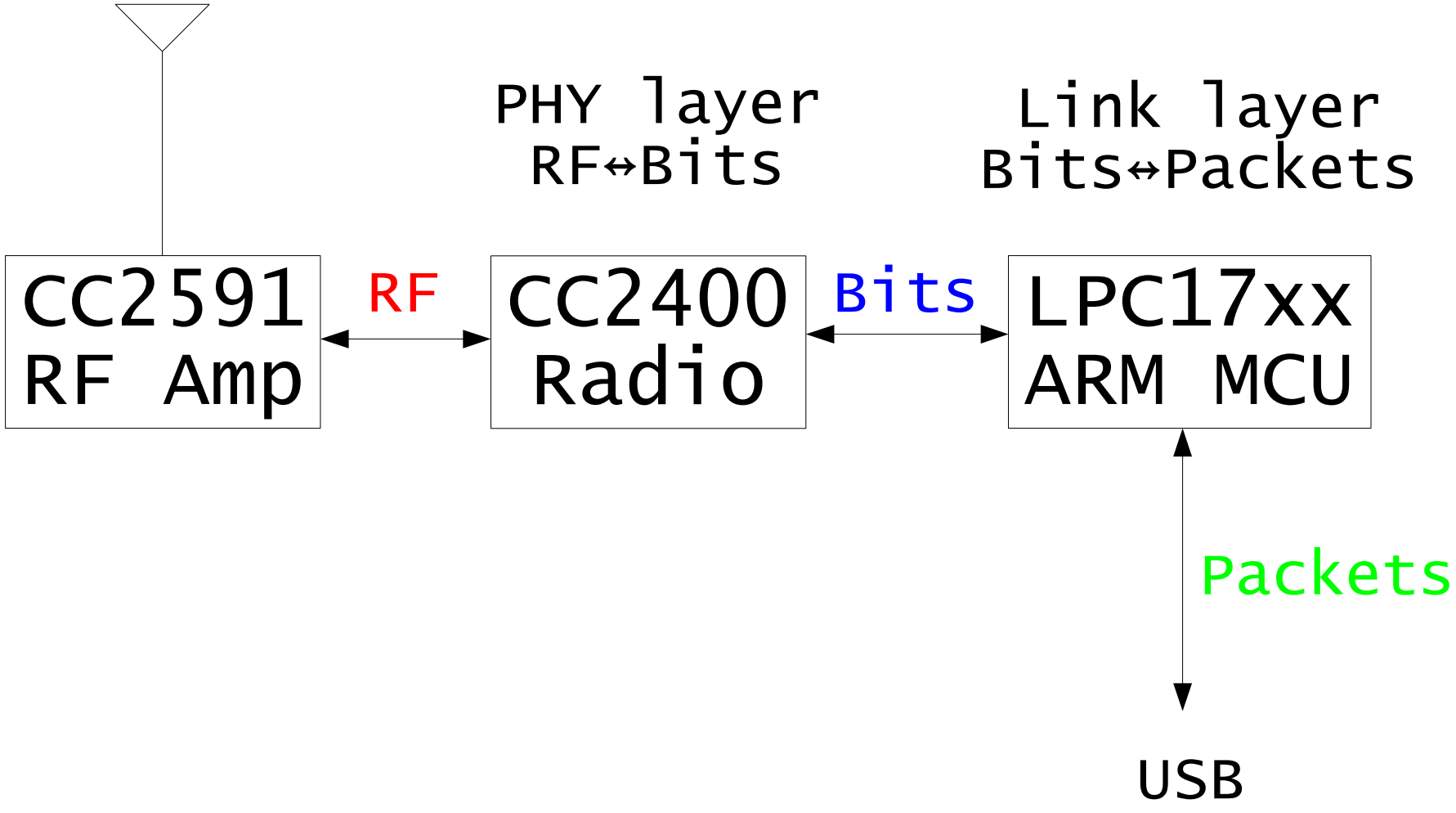
- Sports devices
- High-end smart phones
- Places you wouldn't expect it

How do we sniff it?

Start at the bottom and work our way up:



Ubertooth Block Diagram



PHY Layer

- GFSK, 1 Mbit/sec, +/- 250 kHz
- 40 channels in 2.4 GHz (37 data)
- Hopping

Hopping

- Hop along 37 data channels
- One data packet per timeslot
- Next channel = (channel + hop increment) mod 37

3 → 10 → 17 → 24 → 31 → 1 → 8 → 15 → …

hop increment = 7

Capturing Packets

- Configure CC2400
  - Set modulation parameters to match BTLE
  - Tune to proper channel
- Follow connections according to hop pattern
  - Hop increment and hop interval, sniffed from connect packet or recovered in promiscuous mode
- Hand off bits to ARM MCU

Link Layer

LSB MSB

Preamble (1 octet)	Access Address (4 octets)	PDU (2 to 39 octets)	CRC (3 octets)

Figure 2.1: Link Layer packet format

What we have: Sea of bits

What we want: Start of PDU

What we know: AA

10001110111101010101
10011100001000110011
11100100110100011101

PHY Layer.. Link Layer..

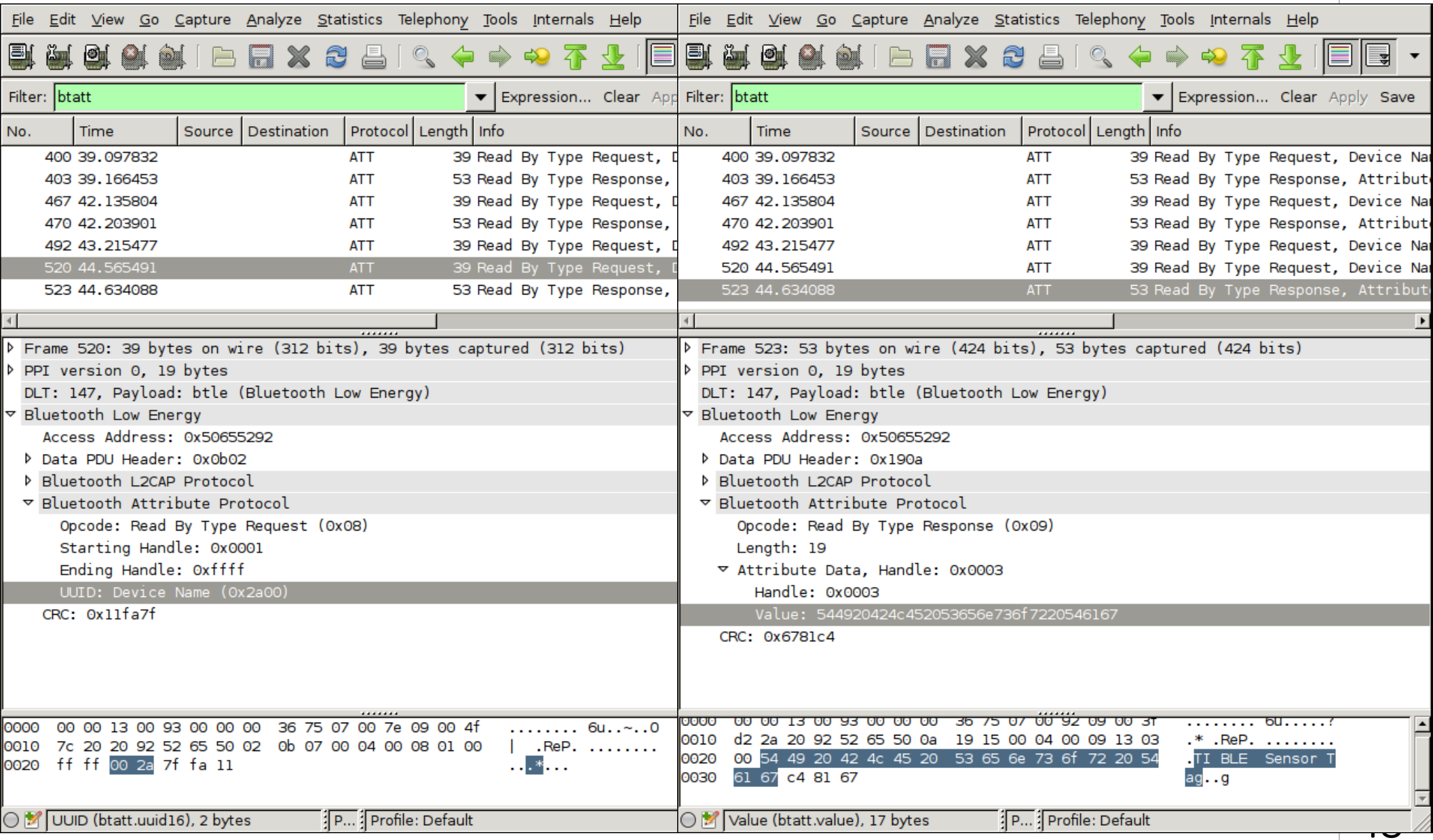
We converted RF to packets
Now what?

Capturing Packets... To PCAP!

- ubertooth-btle speaks packets
- libpcap → dump raw packet data
- PPI header (similar airodump-ng and kismet)

- Still waiting on a DLT for BTLE
  - Unique identifier for the protocol
  - Really shouldn't make a public release without this

Wireshark Awesomeness



Injection

- Pretty much the same as receiving, opposite direction
- Follow the spec!
  - Link layer header
  - Payload data
- Hand that off to Ubertooth
  - Calculate CRC
  - Whiten
- Devil is in the CC2400 details

Faux Slave: Status

- Demo
  - Demo
    - Demo
      - Demo
        - Demo
- Ubertooth shows up under device scan
- Does not yet respond to scan requests
  - Master → slave: “What is your name?”


GOOD IDEA
BAD IDEA

Good Idea: Using AES-CCM

Bad Idea: Custom Key Exchange Protocol

Custom Key Exchange Protocol

- 3 pairing methods
  - Just Works™
  - 6-digit PIN
  - OOB

- "None of the pairing methods provide protection against a passive eavesdropper" -Bluetooth Core Spec

Cracking the TK

```
confirm
=
AES(TK, AES(TK, rand XOR p1) XOR p2)
```

GREEN = we have it
RED = we want it

TK: integer between 0 and 999,999

Just Works™: always 0!

Cracking the TK – With *crackle*

Total time to crack:

< 1 second

And That's It

- You're done
- With the TK, you can derive every other key
- You can capture the LTK exchange

SECURITY = DEAD

Decrypting – With *crackle*

- Yes, crackle does that too!
- crackle will decrypt
  - a PCAP file with a pairing setup
  - a PCAP file with an encrypted session, given an LTK

BTLE Encryption: DEAD

- crackle can...
  - crack the pairing TK
  - decrypt all future communications

- 100% passively

BTLE Encryption: Caveats

- Every session uses a different session key
- Every session uses several nonces

  Solution: jam the connection to restart a session

- LTK exchanged once, used many times

  Solution: inject LTK_REJECT_IND message

Encryption: Postscript

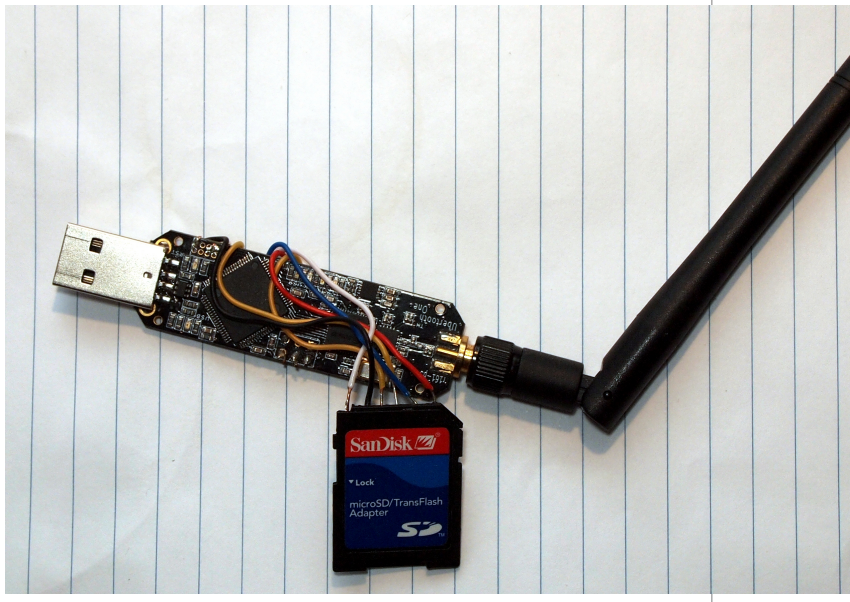
- Don't rely on Just Works or 6-digit PIN
- OOB is not compromised
- Idea: Implement DH in-band to exchange OOB key

Summary

- BTLE sniffing: following and promiscuous
- BTLE injection: PoC slave on Ubertooth
- Capturing to PCAP
- Wireshark plugins for BTLE and BTSM
- Cracking BTLE pairing
- Decrypting passively intercepted communications

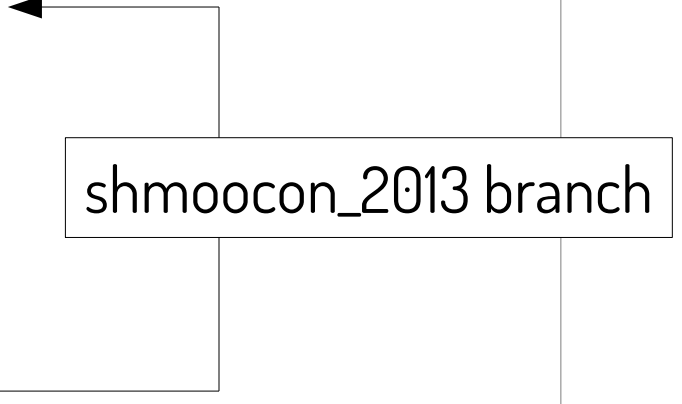
Future Work

- Wireshark capture source → with dragorn
- Flesh out slave on dongle
- Master on dongle
- BTLE stack fuzzer
- SD + battery



It's MY Software and I want it NOW

- crackle
  - http://lacklustre.net/projects/crackle/
  - git://lacklustre.net/crackle
- Ubertooth
  - http://ubertooth.sourceforge.net/
  - git://git.code.sf.net/p/ubertooth/code
- libbtbb (Wireshark plugins)
  - http://libbtbb.sourceforge.net/
  - git://git.code.sf.net/p/libbtbb/code



Thanks

Mike Ossmann
Dominic Spill

Mike Kershaw (dragorn)
#ubertooth on freenode
bluez
Bluetooth SIG

ShmooCon!
iSEC Partners

Thank You

Mike Ryan

iSEC Partners

@mpeg4codec

mikeryan@lacklustre.net

http://lacklustre.net/

http://ubertooth.sf.net/