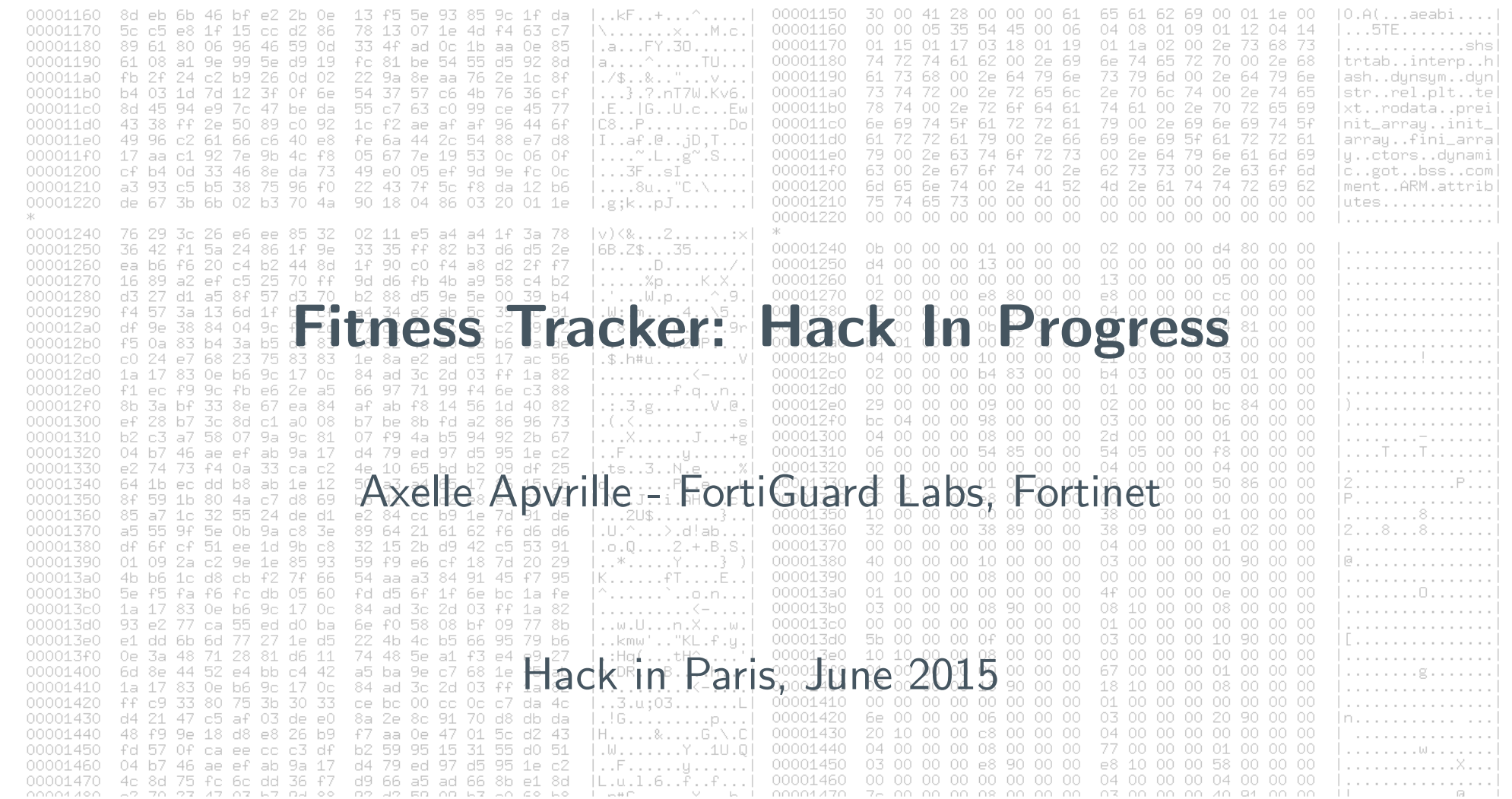


Fitness Tracker: Hack In Progress

Axelle Apvrille - FortiGuard Labs, Fortinet

Hack in Paris, June 2015

FORTINET

Outline

Wearables

Smart watch

Wearables

Smart watch

Wearable camera

Wearables

Smart watch

Wearable camera

wristbands

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

Drowsing detector

Don't drowse during my talk !!!

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

Music beany

Drowsing detector

Don't drowse during my talk !!!

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

Music beany

Drowsing detector
Don't drowse during my talk !!!

Connected T-shirt

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

Music beany

Drowsing detector
Don't drowse during my talk !!!

Connected T-shirt

Augmented reality glasses

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

Music beany

Drowsing detector
Don't drowse during my talk !!!

Connected T-shirt

Augmented reality glasses

Connected shoes

Wearables

Smart watch

Wearable camera

wristbands

Skin exposure detector

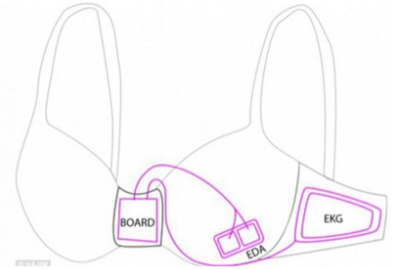
Music beany

And many others: smart helmets, smart pants smart socks

Drowsing detector
Don't drowse during my talk !!!

Connected T-shirt

Augmented reality glasses



Smart bra

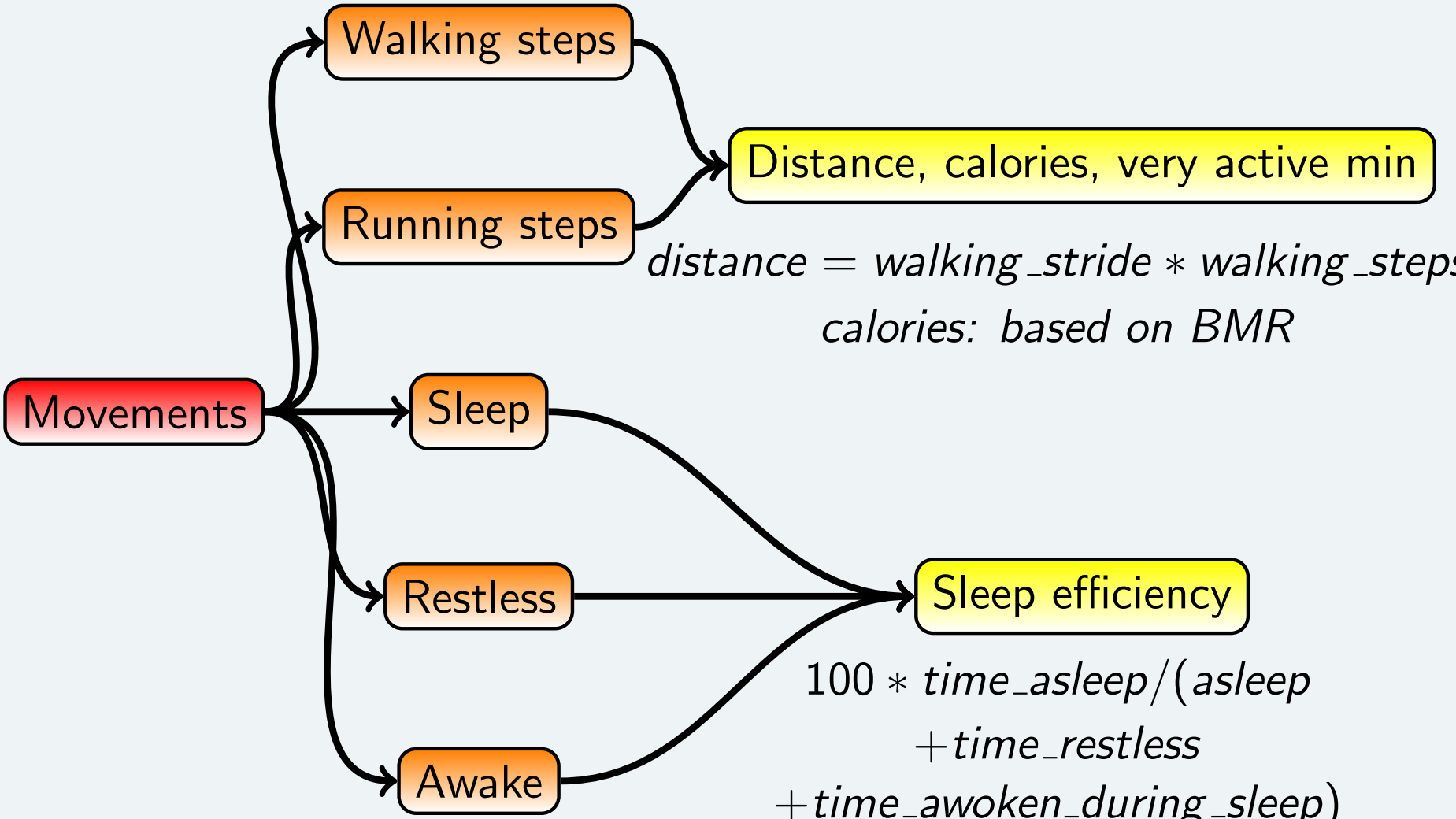
Connected shoes

Outline

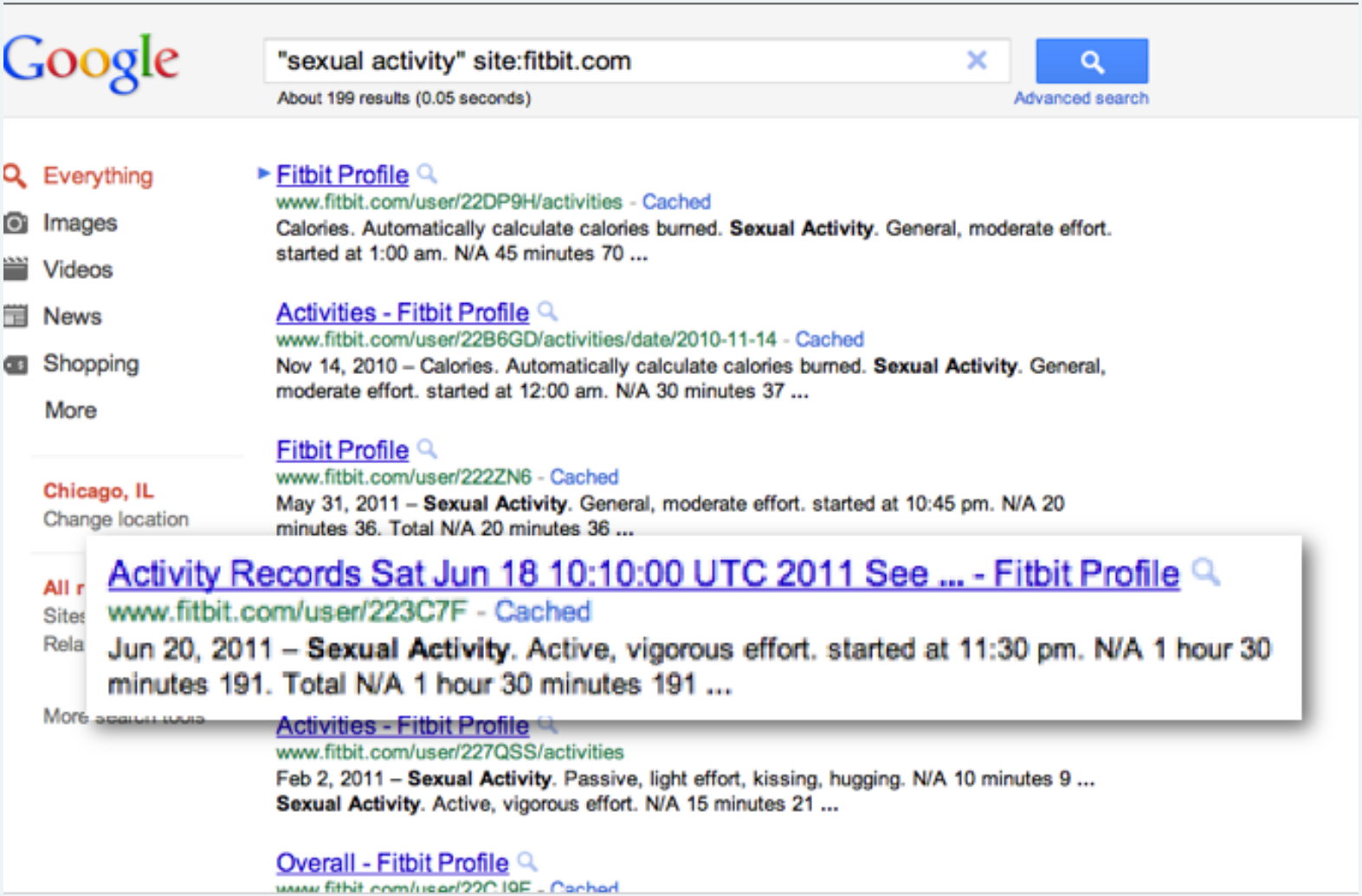
Meet Fitbit Flex

- ST Microelectronics 32L151C6
- Nordic Semiconductor nRF8001 for **Bluetooth Low Energy** v4.0
- ST Microelectronics LIS2DH tri-axial **accelerometer**
- TI BQ24040 **battery** Li-Pol charger
- **No altimeter, no GPS** on Flex. Only on Charge or Surge.

How does it work? (reverse engineered)



Prior issues: default privacy settings of user profiles



Wait, how can the Fitbit Flex track sexual activity?!

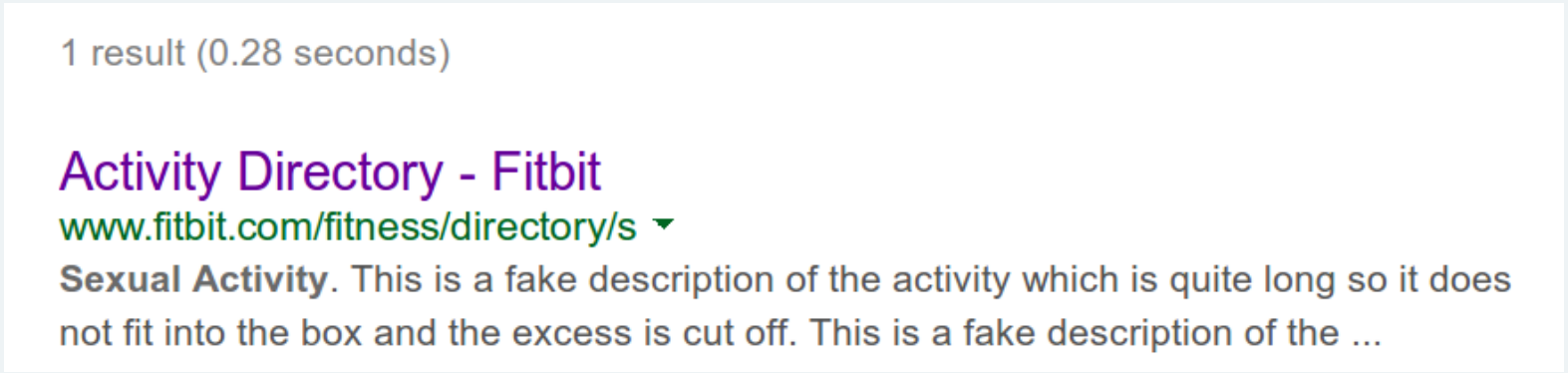
There's only an accelerometer!
How does it the tracker know what I'm doing?
It does not. You enter it manually on the website.

List of activities

Sexual activity - vigorous effort - 105 calories per hour
Cooking Indian bread on an outside stove - 211 calories per hour
Vacuuming - 246 calories per hour
Horse grooming - 422 calories

Those precise categories *no longer exist* (add manually)

The issue is fixed

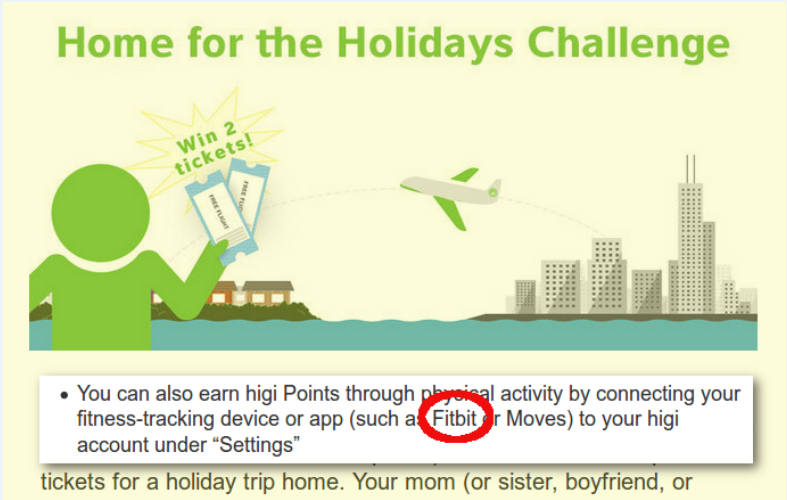


Quick fix the next day

Prevent access to user profiles by search engines
+ erased from Google, Yahoo and Bing

Other prior issues:

- **Clear text login information**. In HTTP POST data.
- **No consistency check, no authentication, no encryption in sync protocol**.
  Set dummy values on tracker or account - yours or someone's else. Earn undeserved awards and profit.

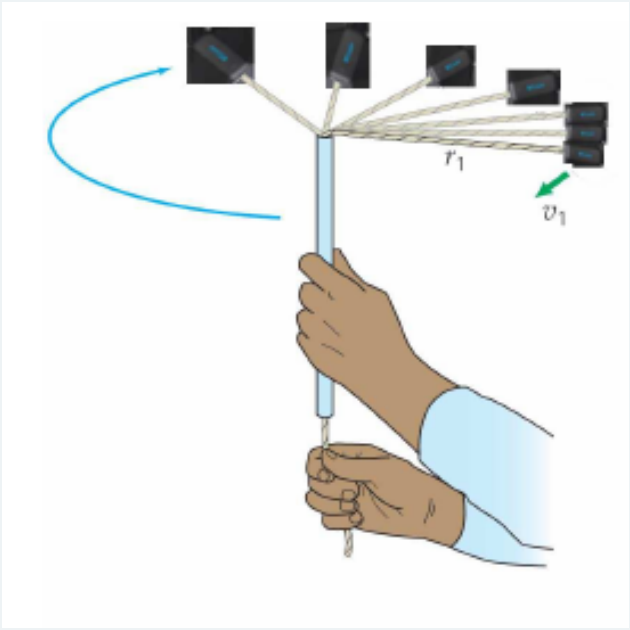


See *Rahman et al. Fit and Vulnerable: Attacks and Defenses for a Health Monitoring Device, 2013*.

Good news

No longer work - *I've checked ;)*

Known (but not so nasty) issues that still work



Images courtesy of Rahman et al. *Fit and Vulnerable* - 2013

- **Abusing physical sensors**. Shaking firmly towards the ground. Or attach to a rope. Or to a car wheel.

Still works

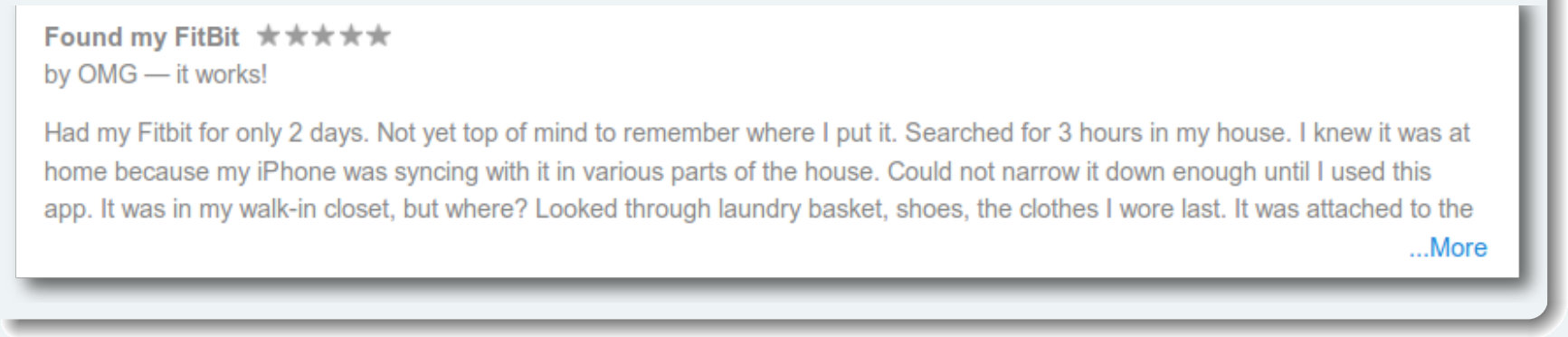
I tested: 50 spins is approx 40 steps.

Known (but not so nasty) issues that still work

Bluetooth tracking:

- LightBlue on iOS, Bluetooth Finder on Android.
- Blueberry sniffer (Raspberry Pi based).

Used to find lost devices



Hack your tracker: why?

New Fitbit trackers have been significantly **locked up**:

- If Fitbit servers are down... your trackers are *useless*
  - You can't read your own walking or running steps
  - You can't set an alarm
  - You can't sync...
- No support besides **Mac** and **Windows**
- What are the trackers transmitting *about us*?
- Should we become concerned by the possibility of trackers getting *infected*?

Outline

Available documentation

Proprietary!

No technical user/ developer/ contributor documentation

Everything has to be reverse engineered

What do we have to start with?

- Product Manual: high level user manual
- **Ben Allard**'s Python utility to sync Flex (Galileo)
- A few (incomplete) notes: Sam Marshall, RAThomas
- My choice: **software** analysis only - "black box analysis"

This is "hack in progress": still much work to do.
Feel free to join.

Actors

BTLE

HTTPS

Fitbit Servers

USB

EndPoint for Tracker

EndPoint for Dongle

Hacker's Host

Actors

BTLE

HTTPS

Fitbit Servers

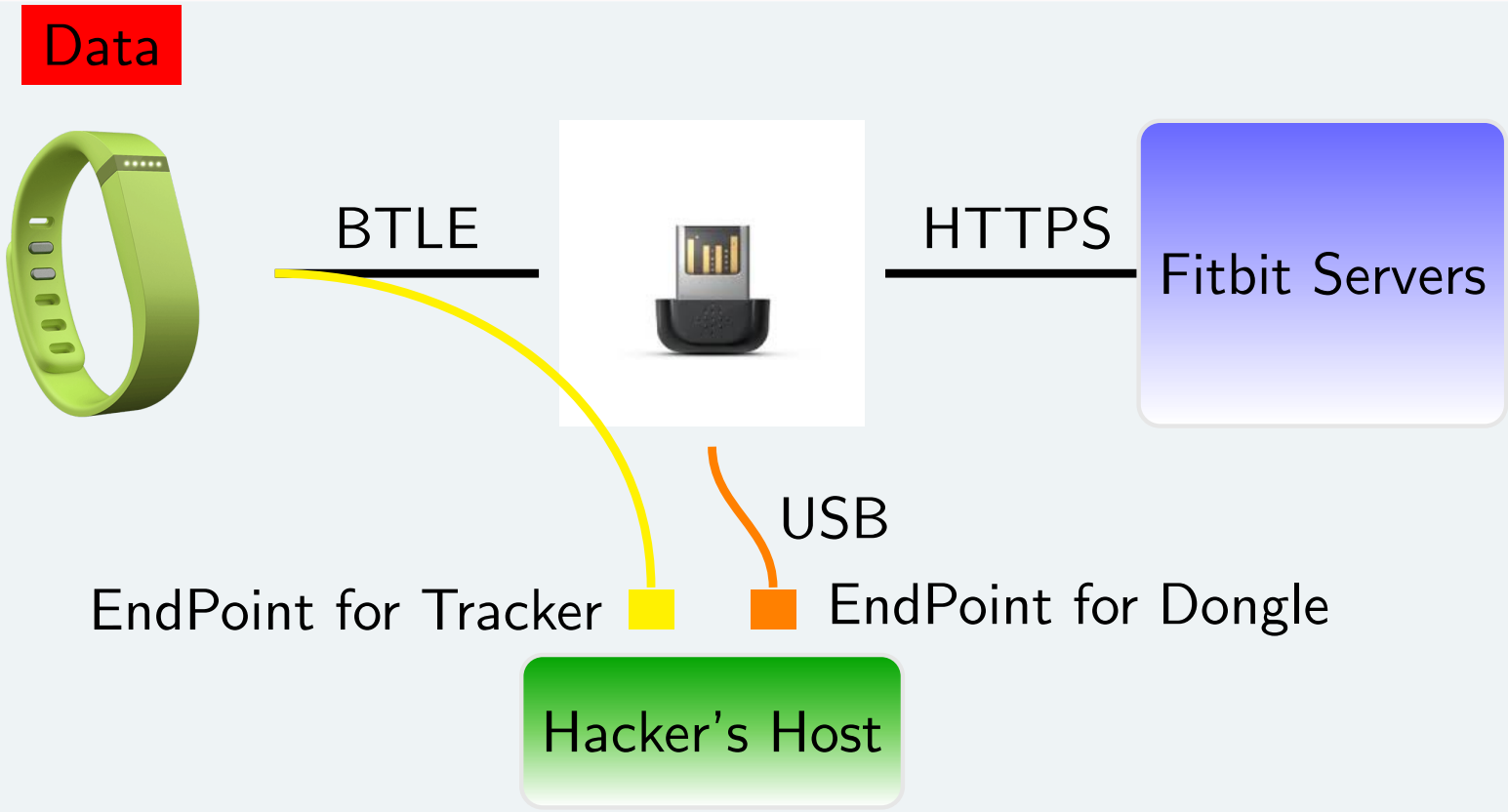
USB

EndPoint for Tracker

EndPoint for Dongle

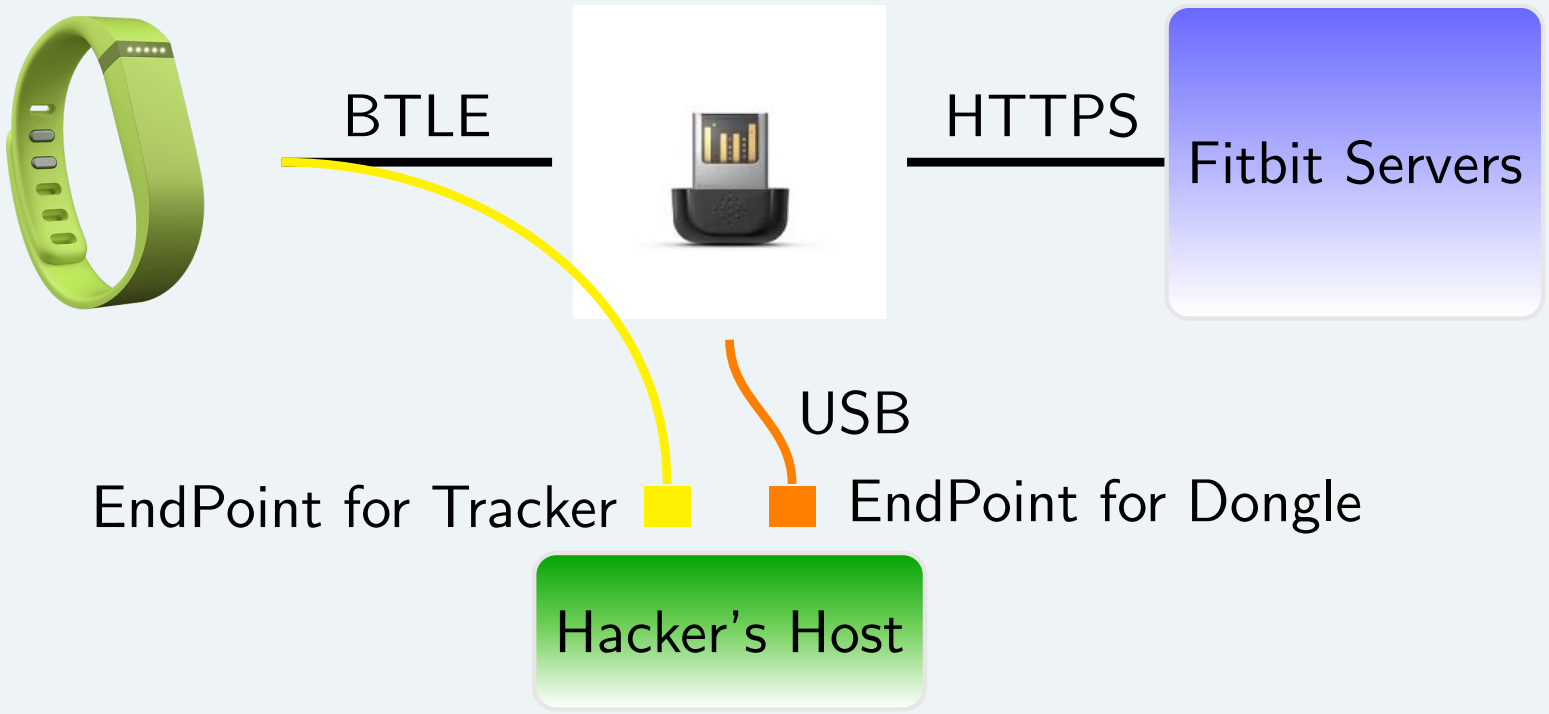
Hacker's Host

Actors


Data
BTLE
HTTPS
Fitbit Servers
USB
EndPoint for Tracker
EndPoint for Dongle
Hacker's Host

Actors

Encrypted Data

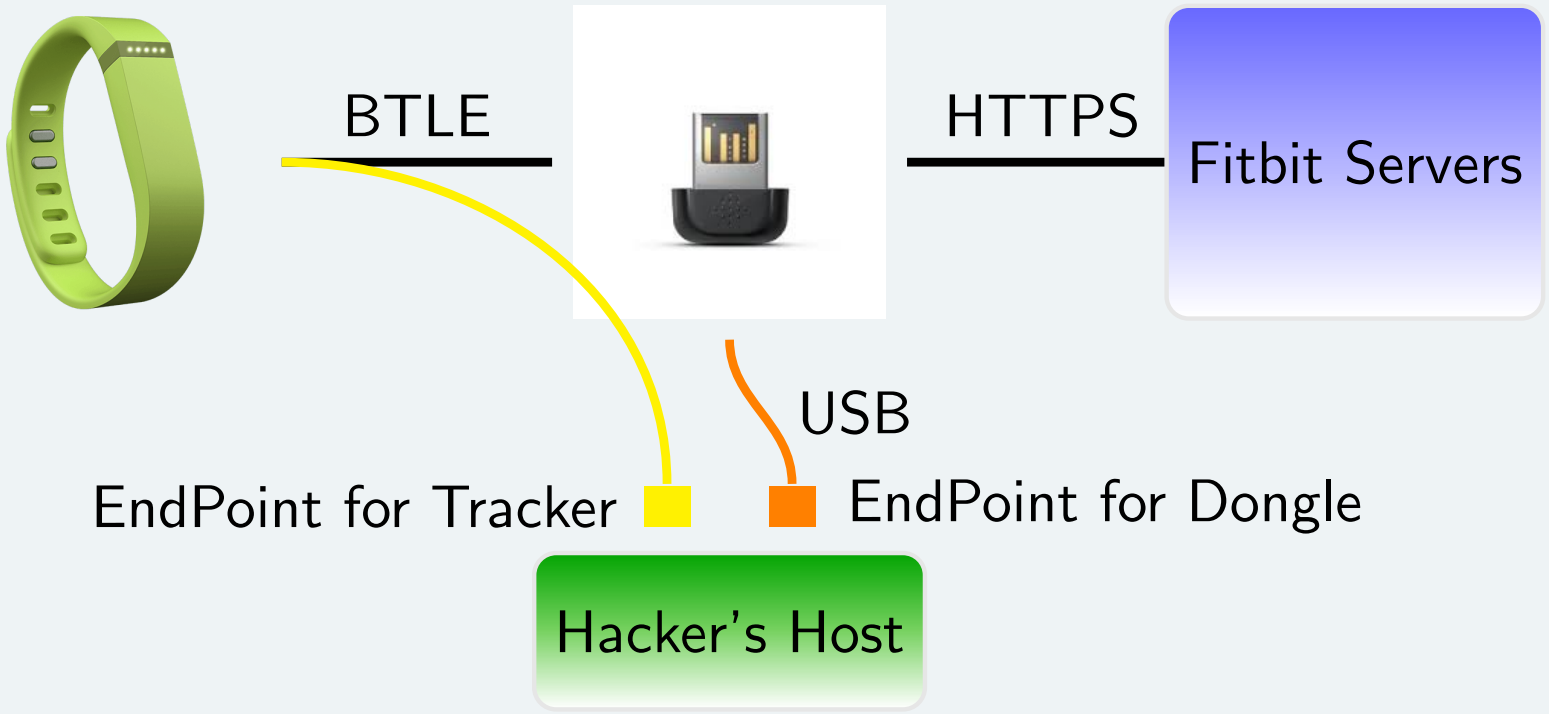


Encryption

Data is encrypted on the **tracker**
Decrypted on Fitbit Servers
Dongle does not encrypt/decrypt

Actors

Encrypted Data ⟶ Encapsulated in XML

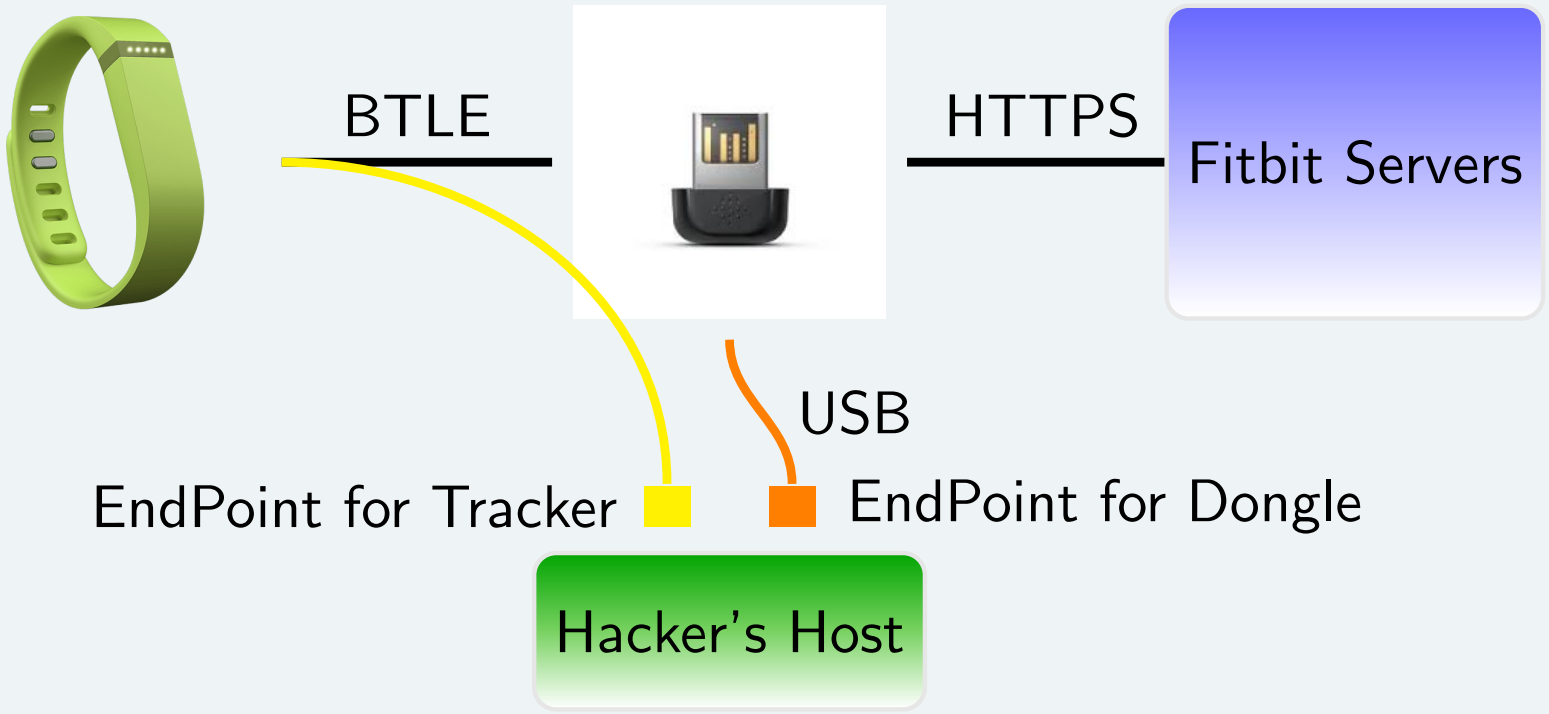


Encryption

Data is encrypted on the **tracker**
Decrypted on Fitbit Servers
Dongle does not encrypt/decrypt

Actors

Encrypted Data → Encapsulated in XML → Decrypted Data



Encryption

Data is encrypted on the **tracker**
Decrypted on Fitbit Servers
Dongle does not encrypt/decrypt

Current Status

Reversed

- 16 dongle messages
- 24 tracker messages
- XML communication between client and server

Current Status

Reversed

- 16 dongle messages
- 24 tracker messages
- XML communication between client and server

What we don't have

Encryption algorithm used between tracker and server

Outline

Linux tools: unclaiming the device

- USB HID claims the dongle (`lsusb`)
- You need to **unclaim** it:

```
import usb.core
import usb.util
import sys

idVendor=0x2687
idProduct=0xfb01
dev = usb.core.find(idVendor=idVendor, idProduct=idProduct)
for interface in range(0,2):
  usb.util.release_interface(dev, interface)
```

Linux tools: Galileo

https://bitbucket.org/benallard/galileo

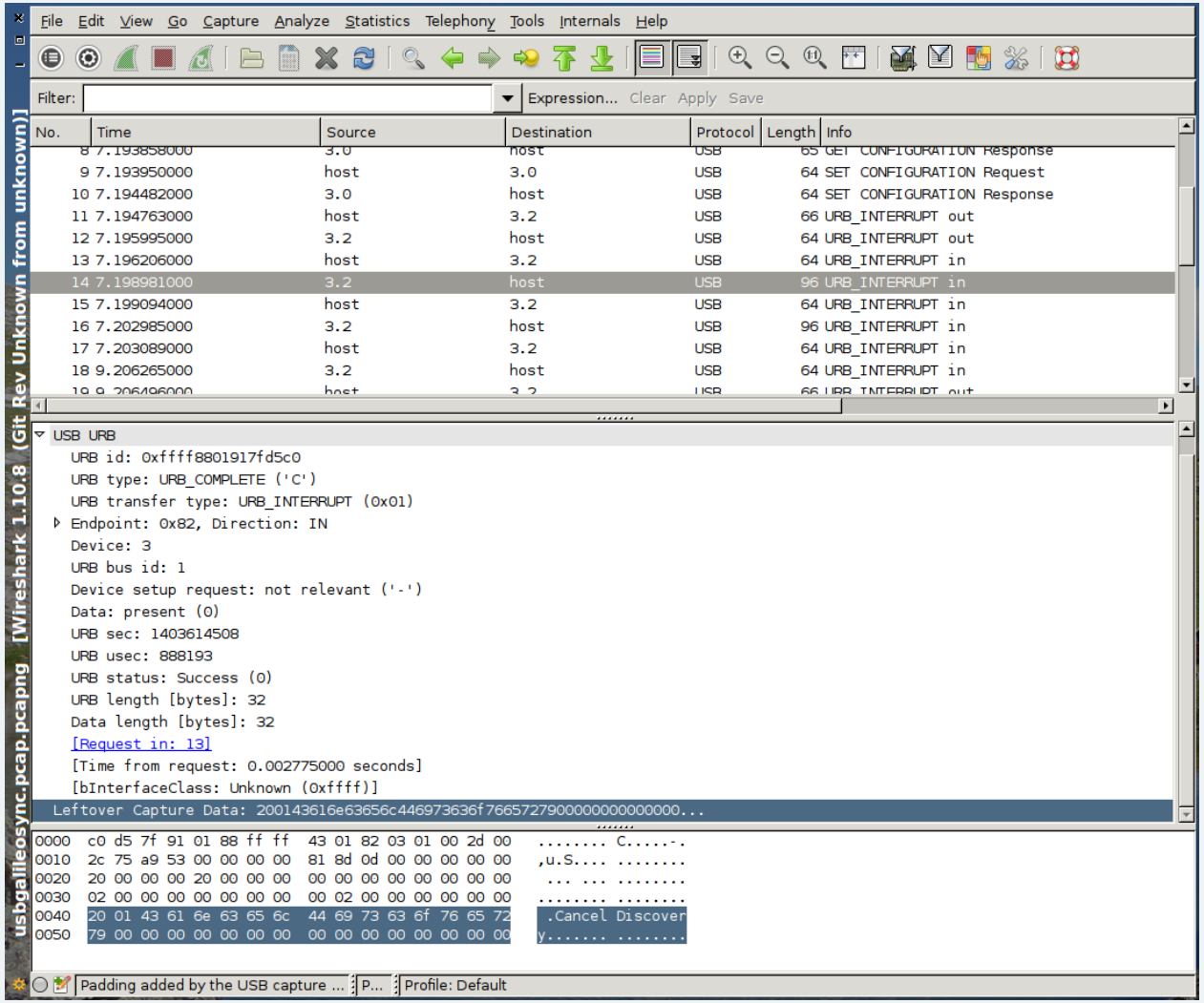
Python utility to synchronize the tracker

```
./run --no-upload --dump --force --no-https-only -d
```

Small interactive mode

```
./run interactive
> c ; d ; l ; tx 1 ; al; => c0 10 03;
<=
```

Linux tools: Wireshark + usbmon




```
tshark -r file.pcap -T fields -e usb.capdata
```

Your own Python code

- **Connect device**.

```
device = usb.core.find(idVendor=0x2687, \
      idProduct=0xfb01)
```

- **Send data** to tracker/dongle.

```
device.write(endpoint, data, timeout)
```

- **Read responses** from tracker/dongle. 32 bytes at most.

```
response = device.read(endpoint, length, timeout)
```

- **Exhaust in pipe:** loop on read until USB error occurs (must not be timeout error)
- **Be patient**: some requests are slow. Set correct timeouts.
- **Handle exceptions/errors**: e.g usb.core.USBError

Fuzzing the tracker

Reverse engineering and fuzzing

- Fuzz all possible **command identifiers** with dummy payload length 0-30
- **Invalid messages** like get bad dump type, or bad toggle pipe

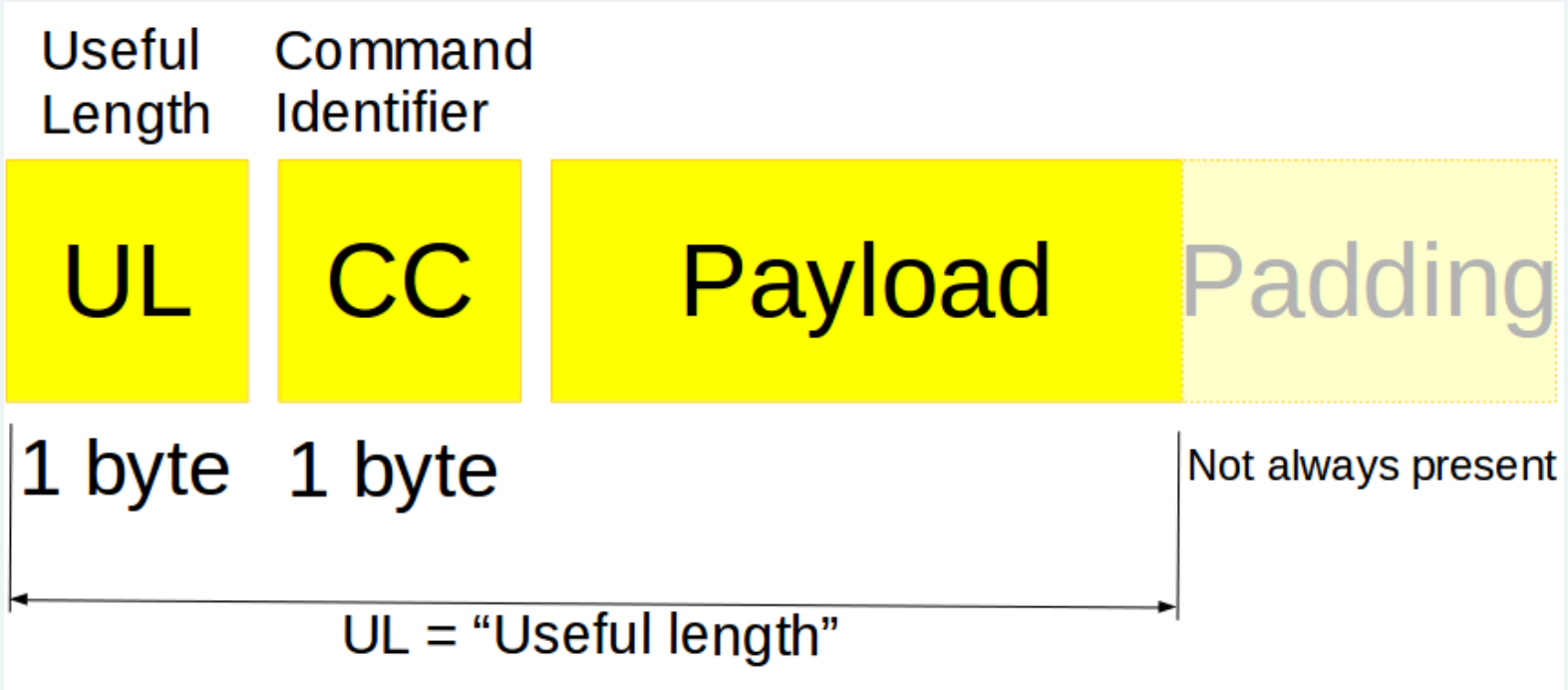
What did we find?

- Many details on packets' format: command identifier on 7 bits only, error code...
- Unknown commands
- Vulnerabilities (Responsible Disclosure)

Outline

Dongle messages

- Endpoints 0x02 (incoming for dongle), 0x82 (outgoing)
- Variable length



- Padding with zeros may or may not be present.
- Useful Length (UL) does not consider padding.

Known dongle requests (EP 0x02)

02 01	Get Dongle Information Request
02 02	Disconnect
1a 04 PP ..	Start Discovery
02 05	Cancel Discovery Request
0b 06 PP ..	Establish Link Request
02 07	Terminate AirLink Request
03 08 PP	Toggle Tx Pipe Request
11 12 PP ..	Establish Link Ex

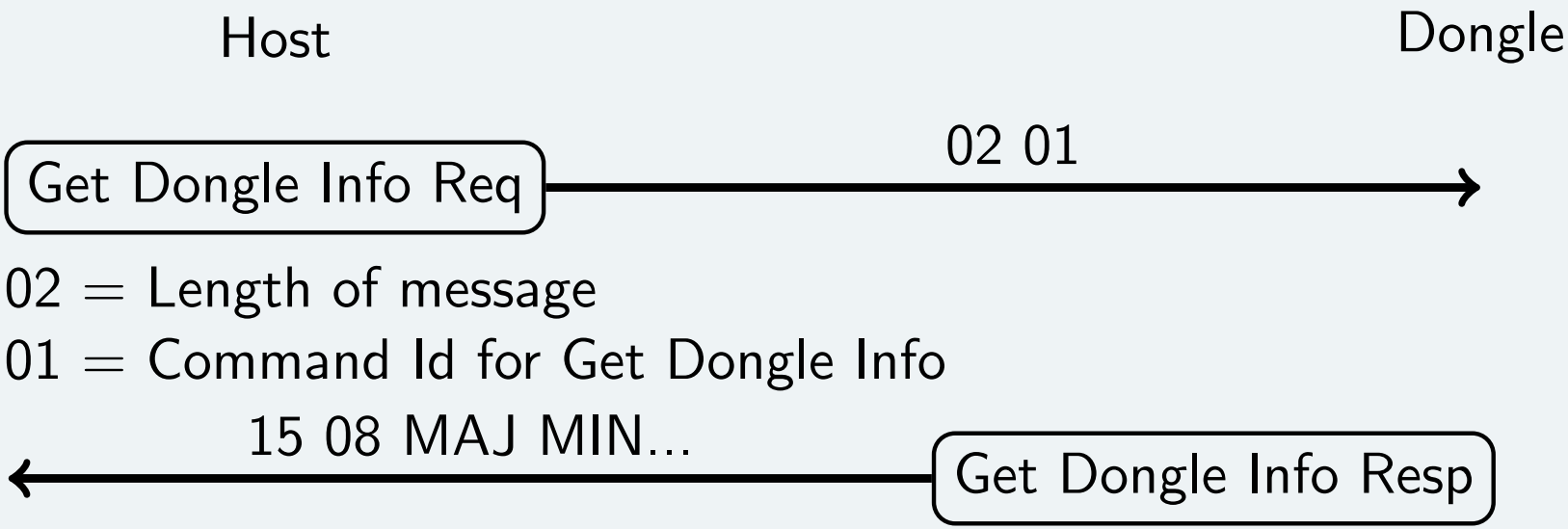
Known dongle responses (EP 0x82)

Format	Padded?	Description
20 01 PP ..	√	Information message
03 02 PP	√	Finished Discovering Trackers
13 03 PP ..	√	Tracker Discovered
03 04 PP ..		Establish Link Response
03 05 PP ..		Link Terminated Response
08 06 PP ..	√	Test Air Link Response
02 07		(Establish Link?) Confirmed
15 08 PP ..	√	Dongle Information Response

We won't detail all of them ;)

Drowsing detected ;)

Example of dongle message flow

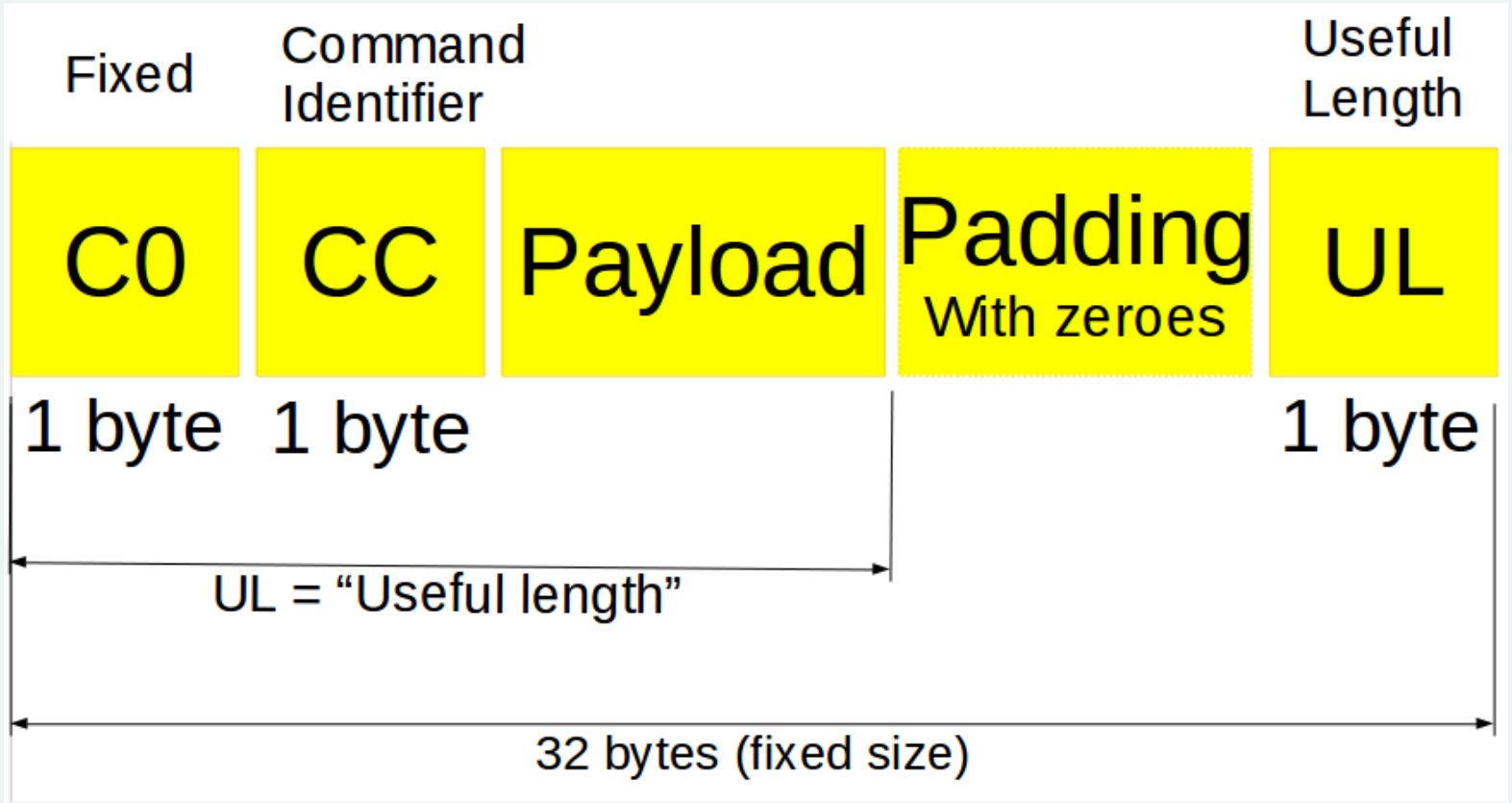



```
15 08 MAJ MIN dd dd dd dd
dd dd 74 04 00 02 00 00
ff e7 01 00 02 Zeroes
```

- Major (1 byte)
- Minor (1 byte)
- Dongle MAC address (6 bytes)
- Remaining seems to be fixed?

Tracker messages

- Endpoints 0x01 (received from tracker), 0x81 (sent to tracker)
- Fixed length
- Padded with zeroes up to 32 bytes



Messages sent to tracker (EP 0x01)

C0 01 -- ... 02	Reset Link
C0 04 PP ... 03	Handle secret (clear or display)
C0 05 -- ... 02	Alert user request
C0 06 -- ... 02	Display code (on the tracker)
C0 09 PP ... UL	Echo Request
C0 0a PP ... 0c	Initialize Air Link
C0 10 PP ... 03	Get Dump Request
C0 24 PP ... 09	Start Transmission
C0 50 PP ... 0a	Client Challenge
C0 52 PP ... 0a	Authentication Response

Messages received from tracker (EP 0x81)

C0 01 -- ... 02	Reset Link Response
C0 02 -- ... 02	Ack Response
C0 03 PP ... 04	Error Code Response
C0 05 -- ... 02	Alert user response
C0 08 -- ... 02	User Activity
C0 09 PP ... UL	Echo Response
C0 0b -- ... 02	Toggle Pipe Response
C0 12 PP ... 05	First Ack Block Response
C0 13 PP ... 05	Ack Block Response
C0 14 PP ... 0c	Air Link Initialized Response
C0 40 PP ... UL	Single block packet response
C0 41 PP ... 03	Start of Dump
C0 42 PP ... 09	End of Dump
C0 51 PP ... 0e	Tracker Challenge

We'll detail every each of them

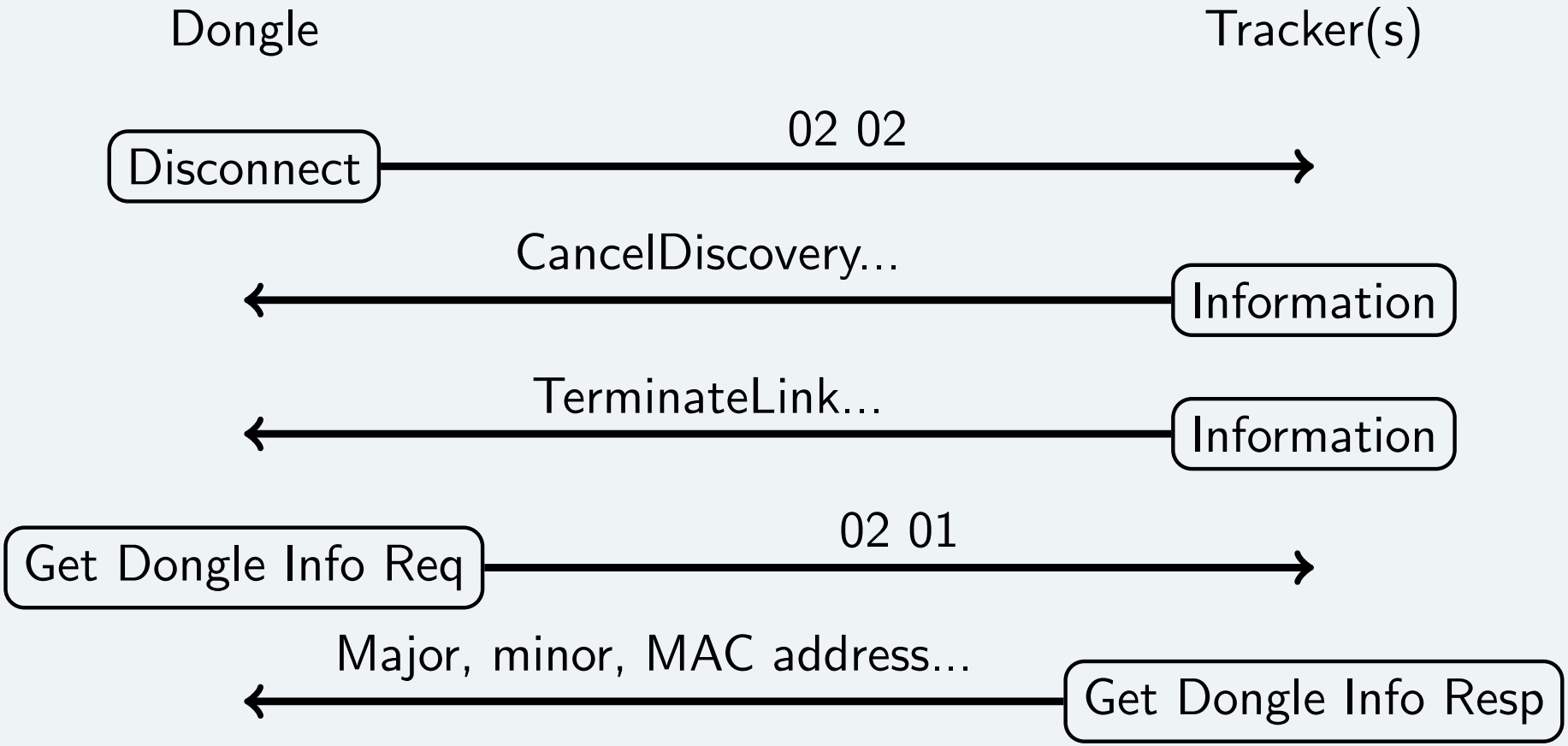
We'll detail every each of them

No - I'm joking :)
Let's only have a look at how to sync (Get Dump)

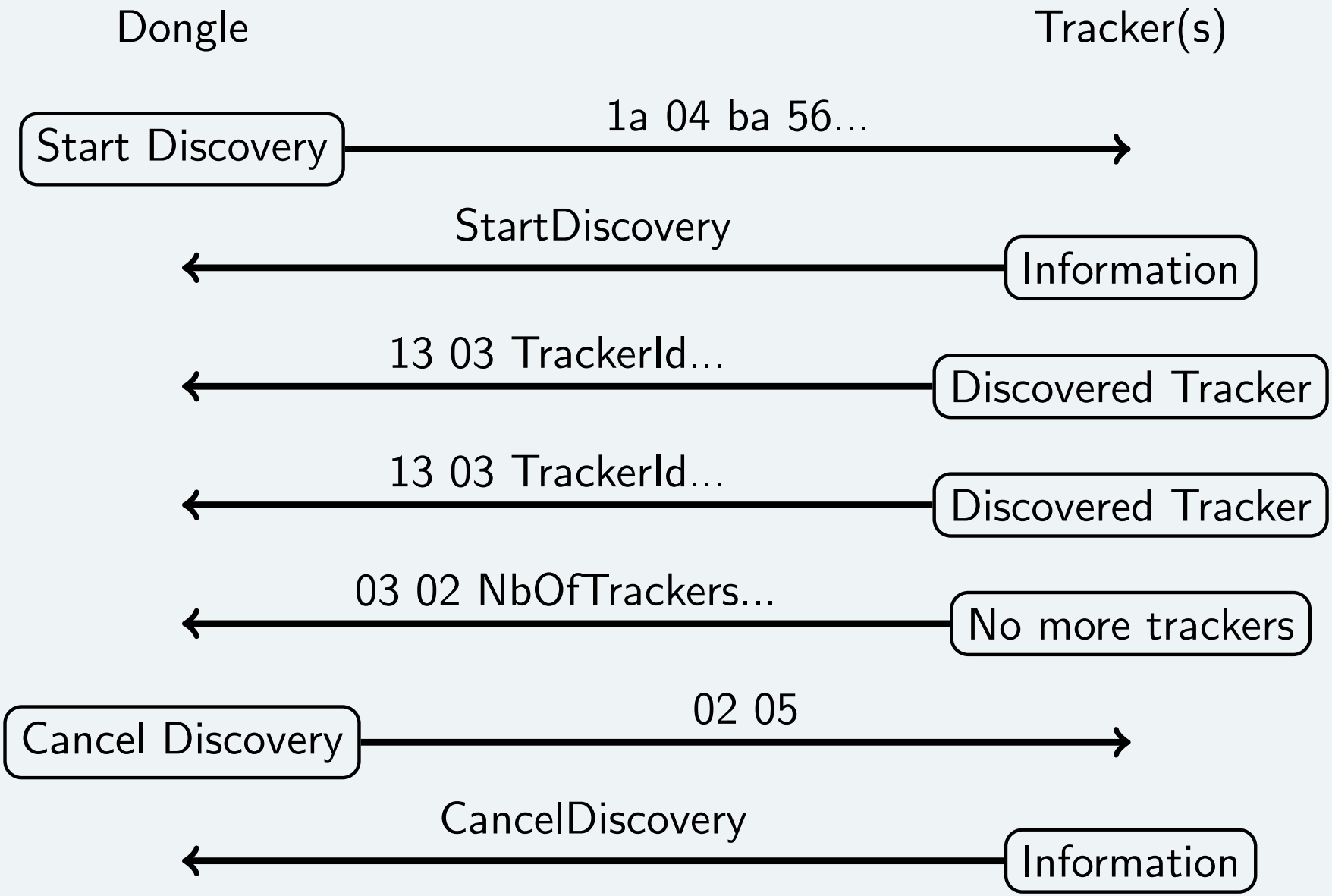
I'm watching those who drowse ;)

Setting up communication with tracker 1/4

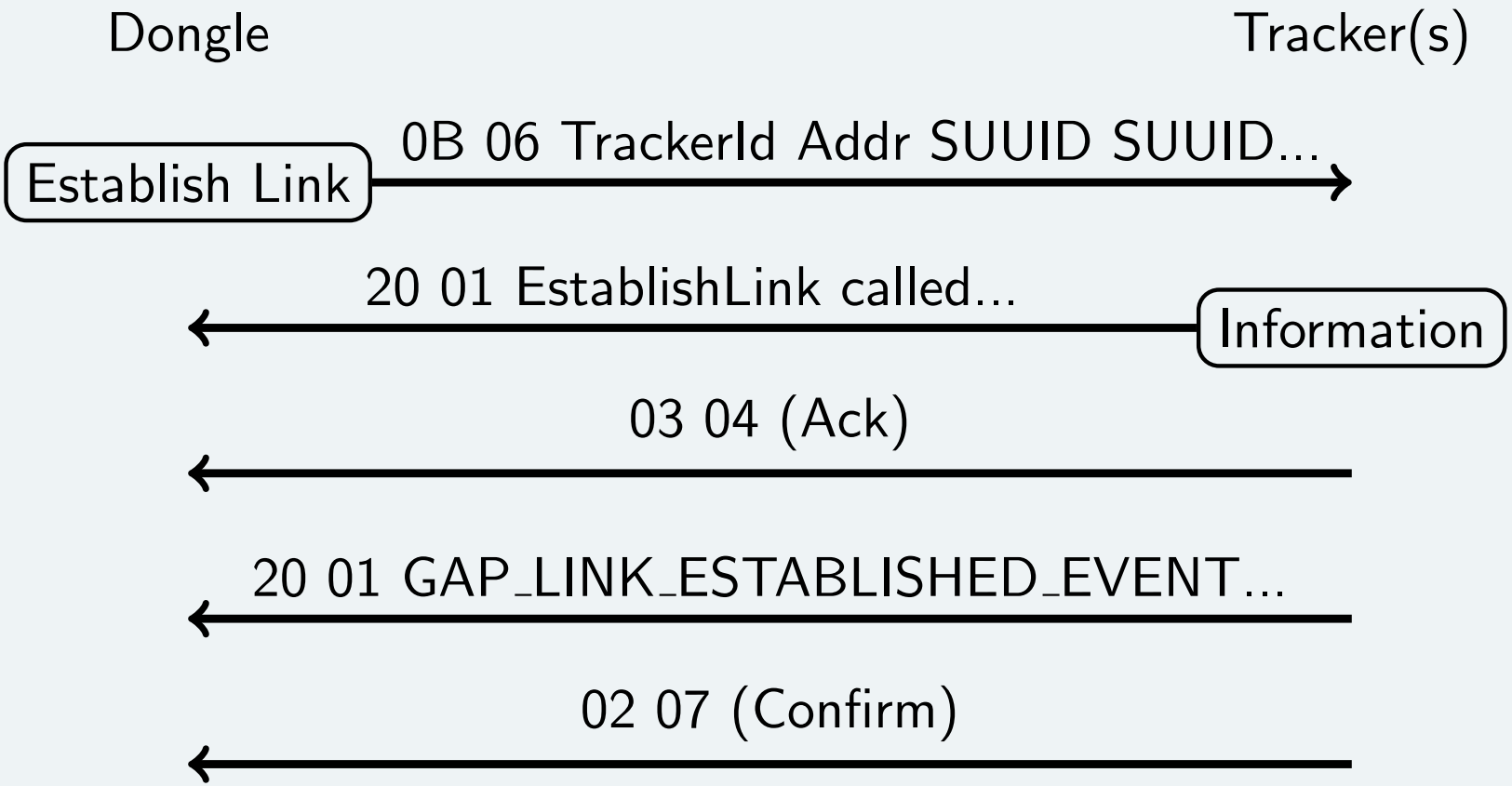
- Disconnect. Clean up current state
- What's my dongle? Get Dongle Information



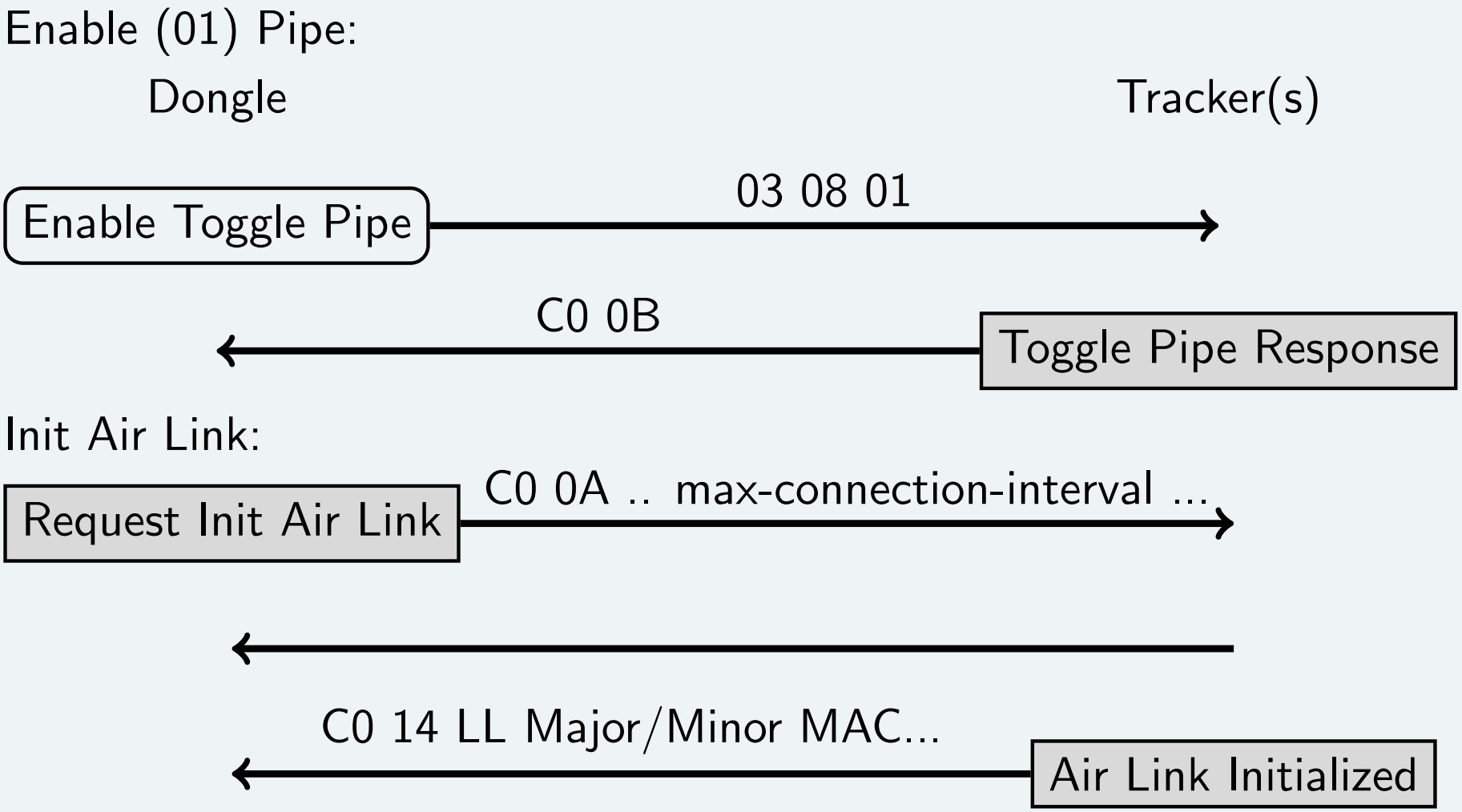
Communication setup 2/4: discovery



Communication setup 3/4: establish link



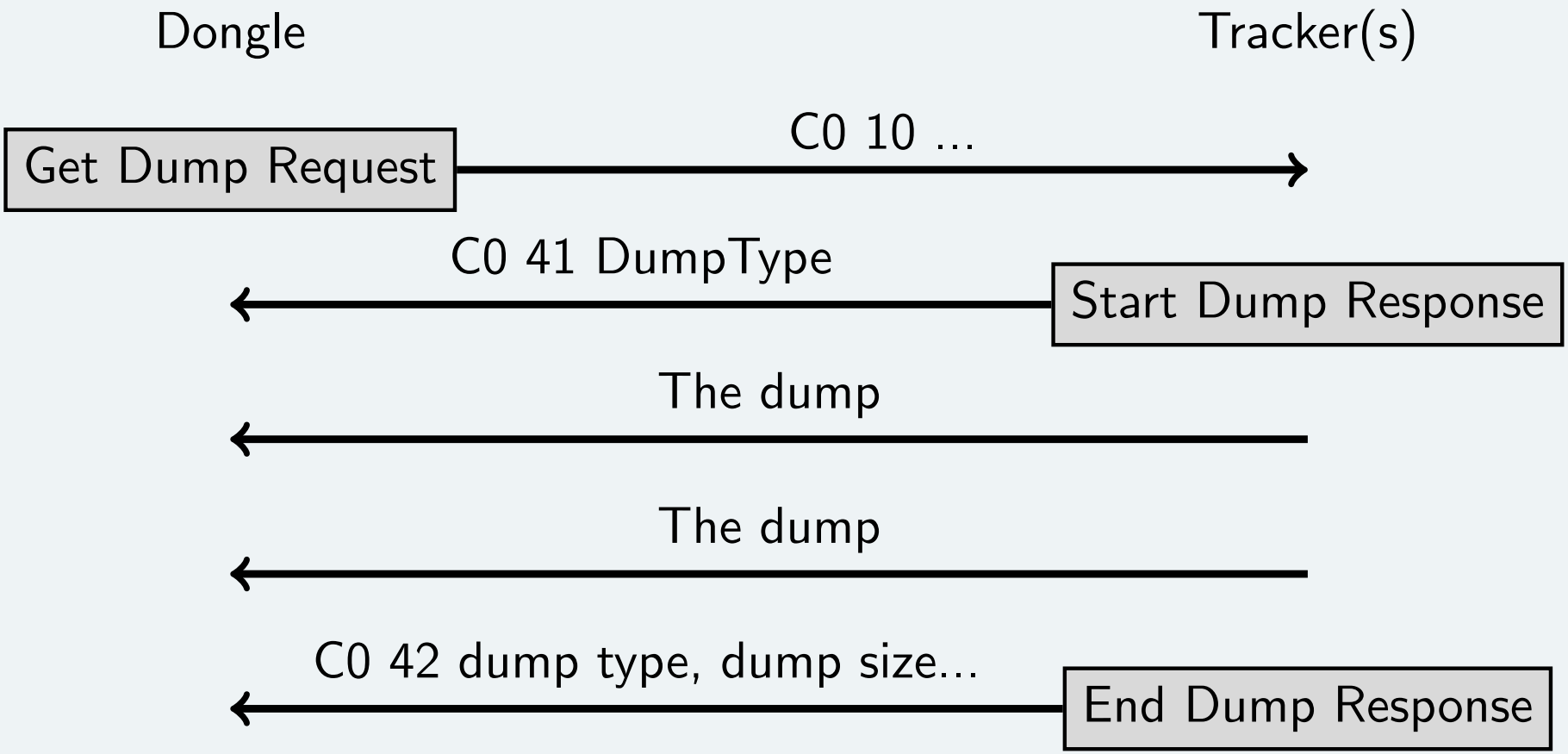
Communication setup 4/4: switch to tracker



Setting up communication with tracker: Recap

- **Disconnect** - to clean up.
- What's my dongle? get dongle info.
- **Discover** trackers.
- **Establish link** with a given tracker.
- **Switch** to communication with tracker
- Initialize **air link** with tracker
- Optional: authentication handshake

Get Dump



If dump holds in a single packet, `c0 40 DumpType Dump ... UL`

Mega / micro dumps

Different messages

- Megadump (0xd): tracker data sync
- Microdump (0x3): e.g requesting an update, pairing
- Megadump responses (0x4) (sent by server) e.g alarms sync
- Microdump responses (0x1) (sent by server) e.g updates

Megadump format

```
TT 02 00 00 01 00 SS SS SS SS MM MM MM MM MM MM
Encrypted ...
```

- **Device type**. Flex 0x28, 0xf4 is Zip, 0x26 is One
- **Sequence counter**. 4 bytes. Little Endian.
- **Model identifier** 6 bytes. Flex: D2 C0 56 2E 15 07 or C9 9B F8 2D 15 07 One: 7B 2E 9F 2B 2D 05 or 7B 16 E4 2A 2A 05 (old).
- Encrypted blob starts at offset 17 (0x11)

Example of megadump

```
28 02 00 00 01 00 7B 11 00 00 D2 C0 56 2E 15 07 08 5E E7 FC
93 83 83 D1 AE 8C 4D E7 D8 F6 B0 32 F5 41 29 EC D5 26 D5 A9
27 F1 A2 17 26 BC 51 4B 66 FA 08 39 6A 7B 07 60 02 5B CC 05
5D E7 3B F8 8B 33 28 6A EA 3B C7 9E C8 22 D5 A7 EA E4 4D A4
68 AB 09 8E D0 AE A7 04 6D CB 42 CC 8C 73 D1 05 47 C8 CD D3
11 01 E6 A3 EA EC 0F F8 44 43 0A 29 E3 24 4F B7 6A 14 BC 0E
33 DD B5 7C 7D F4 6F 49 84 CC BE B2 DE B6 B6 AA 99 8F F5 26
4A 5B 5C 74 65 16 B0 78 87 2A BA 4E 13 FD 75 C4 3B 8A D2 E5
90 60 67 2D 46 07 7A 4F EE 7E 19 56 29 91 D0 98 B8 FF 2A 5A
88 16 4D 9F 79 56 72 BE CD AC CE F0 B1 D0 46 06 24 CD 82 EB
88 AC D6 2A 3A 72 E1 BE 05 E3 A6 5A C8 E8 85 F6 F1 43 55 A5
A7 13 98 E7 F9 4E B3 87 FF 61 9E 94 94 A3 00 4A AE A9 0D D9
87 3D B2 D8 A4 7C 08 35 D7 50 17 9A 9A F4 4F D4 AD D2 09 5C
51 05 36 AE 61 DB 4A FB 60 9E 1E D9 6D 90 65 D5 1D 79 85 A1
02 B4 91 A2 ED E2 B0 60 60 1A 12 9F EC 91 FD CC 21 AE 80 A9
02 C7 C8 18 E8 3F 8C F7 DC 90 99 32 14 01 00
C0 42 0D B9 03 3B 01 00 00
```

Communication with Fitbit Server

```
<?xml version="1.0" encoding="UTF-8" standalone=yes?>
 <galileo-client version="2.0">
 ...
  <tracker tracker-id="DEADBEEF0000" type="megadump">
   <data>
    Base 64 dump ...
   </data>
  </tracker>
 </galileo-client>
```

Encryption algorithm between tracker and server

What algorithm???
(and what key...)

- Megadump size different at each sync (1836 bytes, 1859(+23), 1877(+18), 1900(+23), 1915(+15)...)
- Differential cryptanalysis: ciphertext-only attack
- Hardware inspection? JTAG?
- Any other idea?

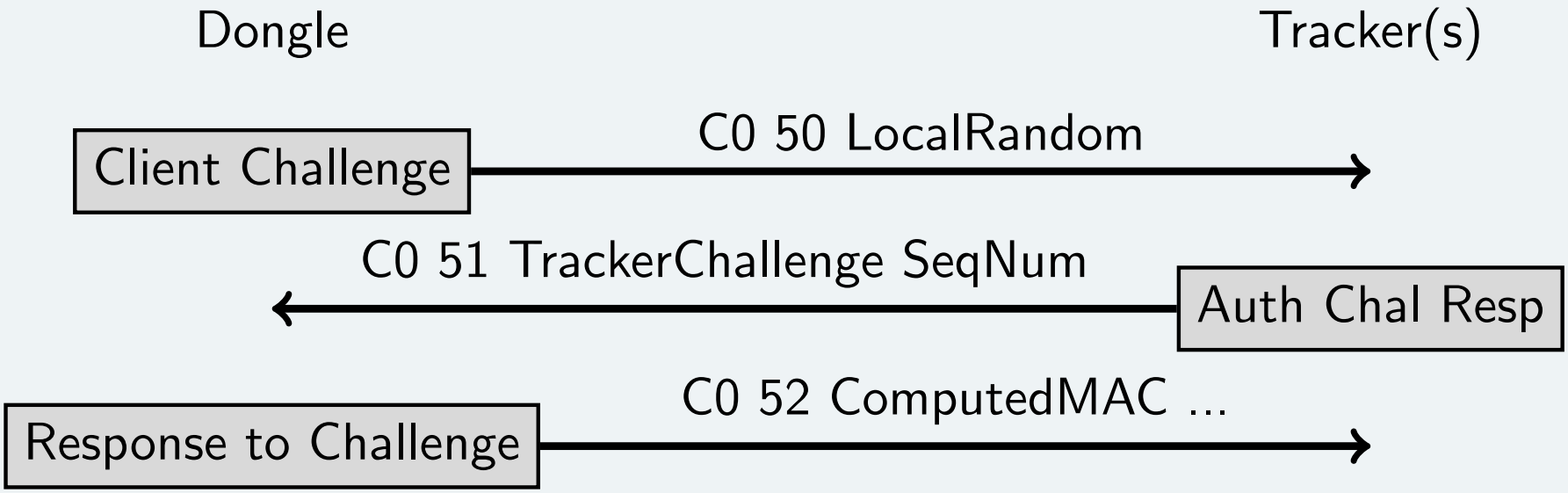
Guessing the algorithm

- STM32L151 does not support AES, only STM32L162...
- Yet, AES (or other) can be used and not accelerated by hardware
- Authenticity: XTEA-CMAC or AES-CMAC (Surge)
- Encryption: XTEA?

Outline

Fitbit tracker as source of entropy?

Use authentication packets:



- Send a dummy local random (C0 50)
- Wait for tracker's response: 8-byte challenge
- Never send last message (C0 52)

Example

Getting random bytes

```
$ python rndflex.py -b 256
e3 57 5a d0 00 14 4a b2
25 d3 91 0b 21 5b c1 e4
fd 9e c9 8d e8 c4 9e 90
76 ba 01 1f ba 56 95 19
...
```

- Entropy 7.72: ok
- Source code: https://github.com/cryptax/fittools

Outline

Thanks for your attention!

Contact info

@cryptax or `aapvrille` (at) `fortinet` (dot) `com`

More fun

There's still lots to do:

- Control the LEDs on the tracker
- Make the tracker vibrate
- Fuzz the dongle, fuzz the server
- Algorithm for dumps?