Bluetooth Low Energy Protocol, Security & Attacks

Online Beer-Talk
07.05.2020 17:00

Emanuel Duss <emanuel.duss@compass-security.com>

Speaker

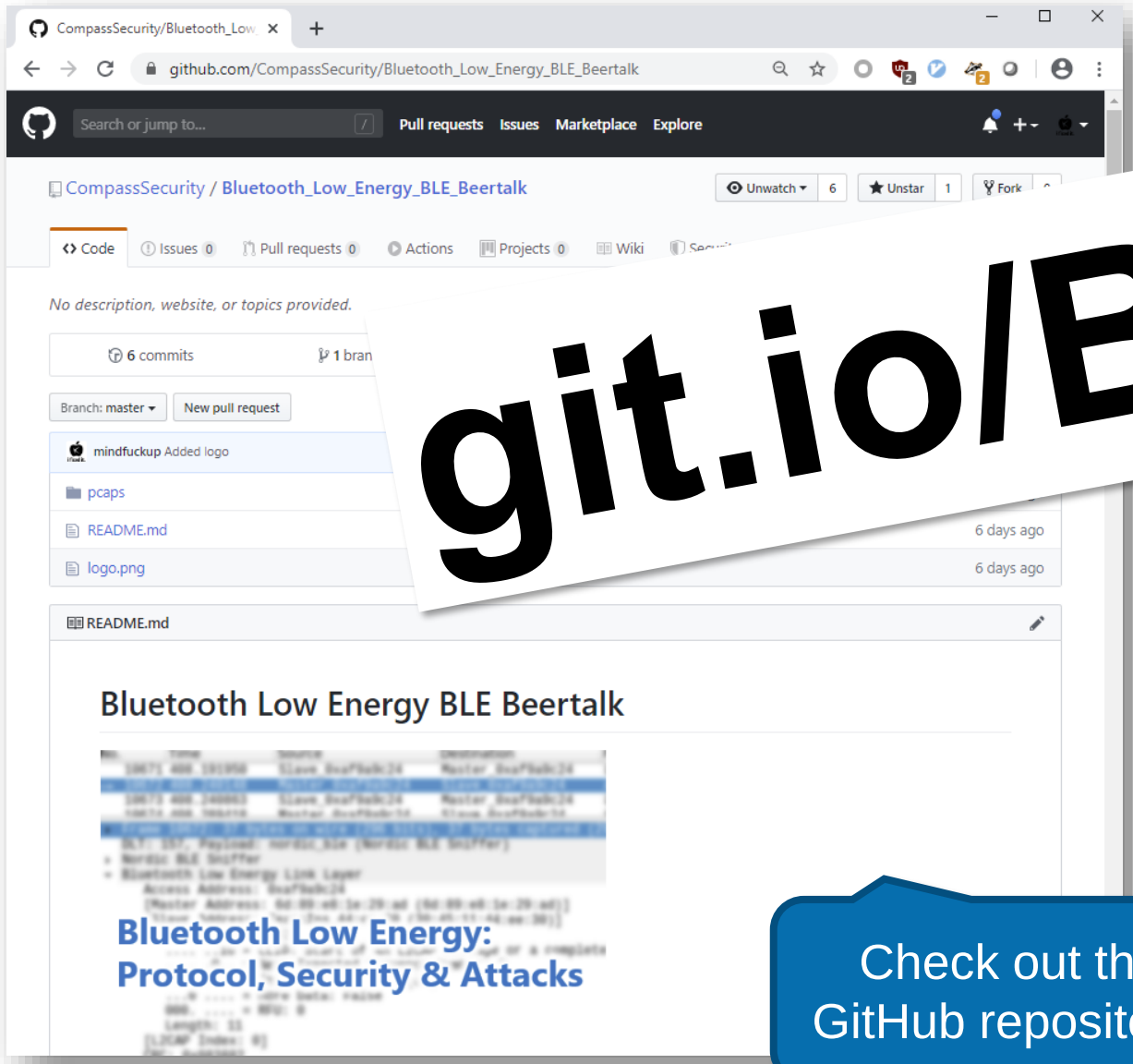
- Emanuel Duss
- Bachelor Thesis @ HSR: «SAML Raider» Burp Extension in collaboration with Compass
- IT Security Analyst @ Compass Security since 2016
- Focus: Pentesting web apps, external/internal networks, mobile apps, hardening reviews, also a course teacher
- I like Linux, shells, CLI, networks, protocols, lockpicking and generally when things break 😉
- Bluetooth Low Energy: Research Topic 2019
- Online
  - Twitter: @mindfuckup
  - GitHub: https://github.com/mindfuckup
  - Web: https://emanuelduss.ch
  - Mail: emanuel.duss@gmail.com, emanuel.duss@compass-security.com

Agenda

- Introduction to Bluetooth Low Energy (BLE)
- BLE Security Mechanisms
- BLE Sniffing
- BLE Interaction
- BLE Man-in-the-Middle
- BLE Hijacking
- Example BLE Attacks
- BLE 5

I'll skip some stuff in this Beer-Talk so I can finish in ≤ 1h 😉.

Resources @ https://github.com/CompassSecurity



git.io/BLE

Check out the GitHub repository!

- Slides (full version!)
- Links to demo videos
- Links to this Beer-Talk video
- Links to software / hardware
- Example PCAPs
- Links to further resources

Introduction to Bluetooth Low Energy

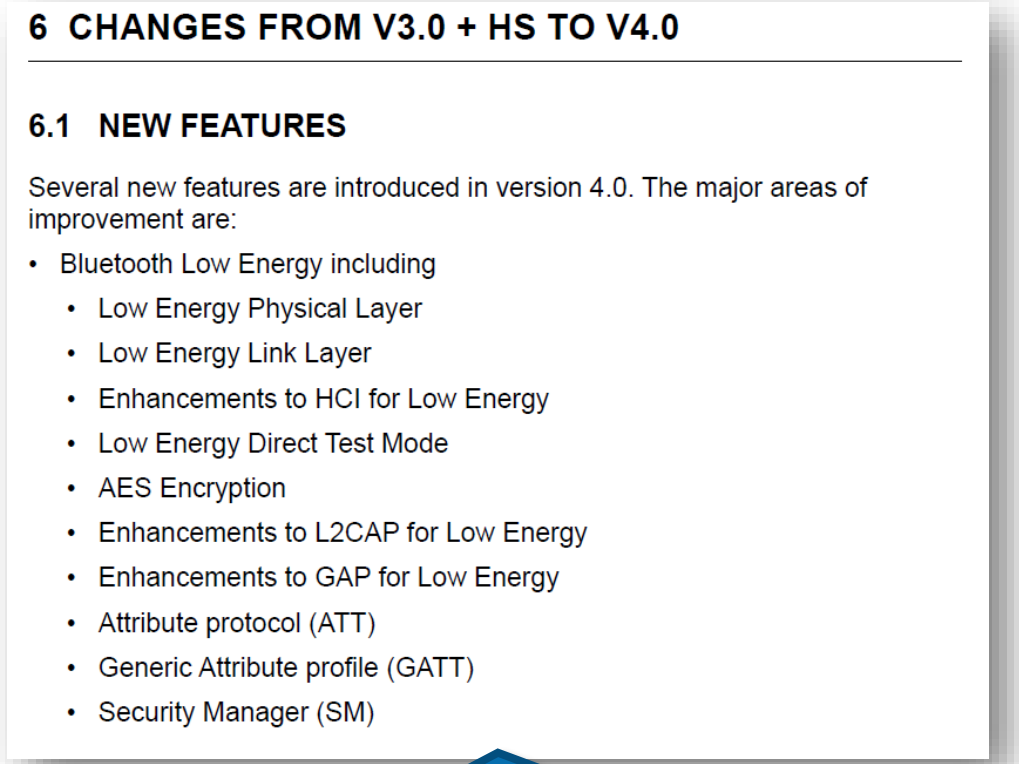
Bluetooth

- Short-range wireless communication system intended to replace cables
- Key Features: Robustness, low power consumption, low cost
- Many features, many are optional
- Basic Rate (BR)
  - 721.2 kbit/s Basic Rate (BR) & 2.1 Mbit/s Enhanced Data Rate (EBR)
  - Up to 54 Mbit/s with 802.11 AMP
  - Up to 100 meter distance
- Low Energy (LE)
  - Not compatible with BR/EBR
  - High-level protocols are reused
- Both forms include device discovery, connection establishment & connection mechanisms

Standardized by Bluetooth SIG (Special Interest Group)

Bluetooth Low Energy

- Part of Bluetooth 4.0 core specification (2010)
  - Also known as Bluetooth Smart until 2016
- Low…
  - Lower complexity
  - Lower power consumption / duty cycles
  - Lower cost
  - Lower data rates (1 Mbit/s or 2 Mbit/s in BLE 5.0)
- But also up to 100 meter distance (400 m in BLE 5.0)
- Connectionless Model: Not necessarily a cable replacement (short-term connections, fast connection setup)
- Versions
  - Version 4.2: More secure pairing
  - Version 5.0: 2 MB/s, longer range, changes in advertising
  - Version 5.1: GATT caching, changes in advertising



6 CHANGES FROM V3.0 + HS TO V4.0

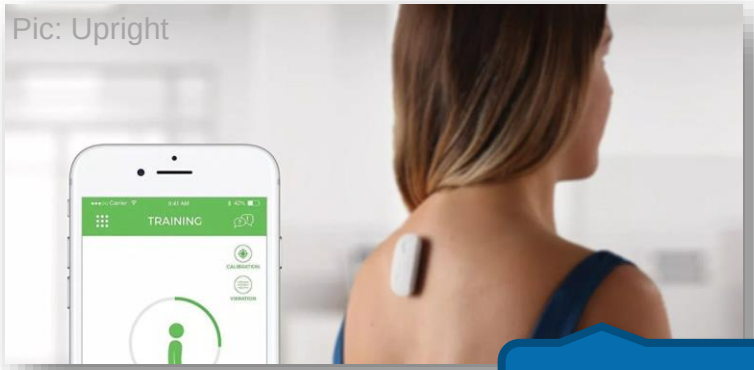
6.1 NEW FEATURES

Several new features are introduced in version 4.0. The major areas of improvement are:

- Bluetooth Low Energy including
  - Low Energy Physical Layer
  - Low Energy Link Layer
  - Enhancements to HCI for Low Energy
  - Low Energy Direct Test Mode
  - AES Encryption
  - Enhancements to L2CAP for Low Energy
  - Enhancements to GAP for Low Energy
  - Attribute protocol (ATT)
  - Generic Attribute profile (GATT)
  - Security Manager (SM)

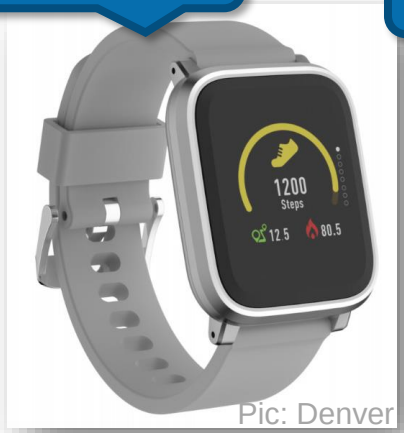
Bluetooth 4.0 Core Specification

BLE Devices

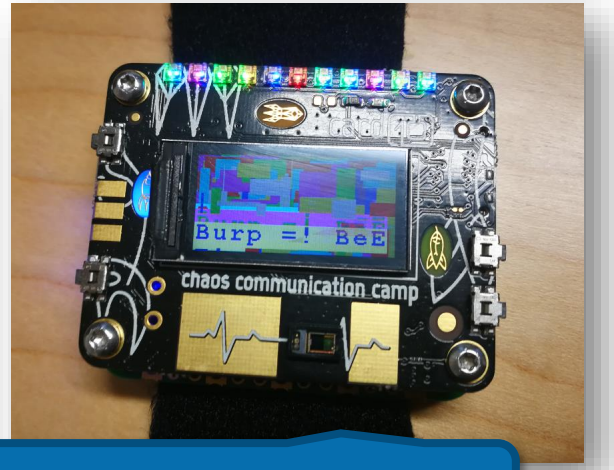


Monitoring

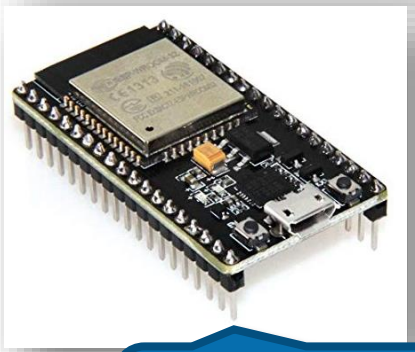
Wearables



Smart Home



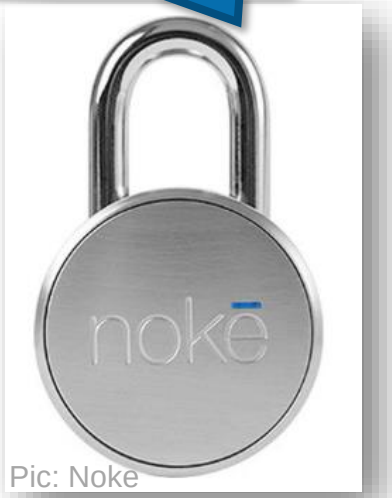
Conference Badges



ESP32
BLE CTF Game

Coffee Machine @ Office
Bern (THX Ivano!)

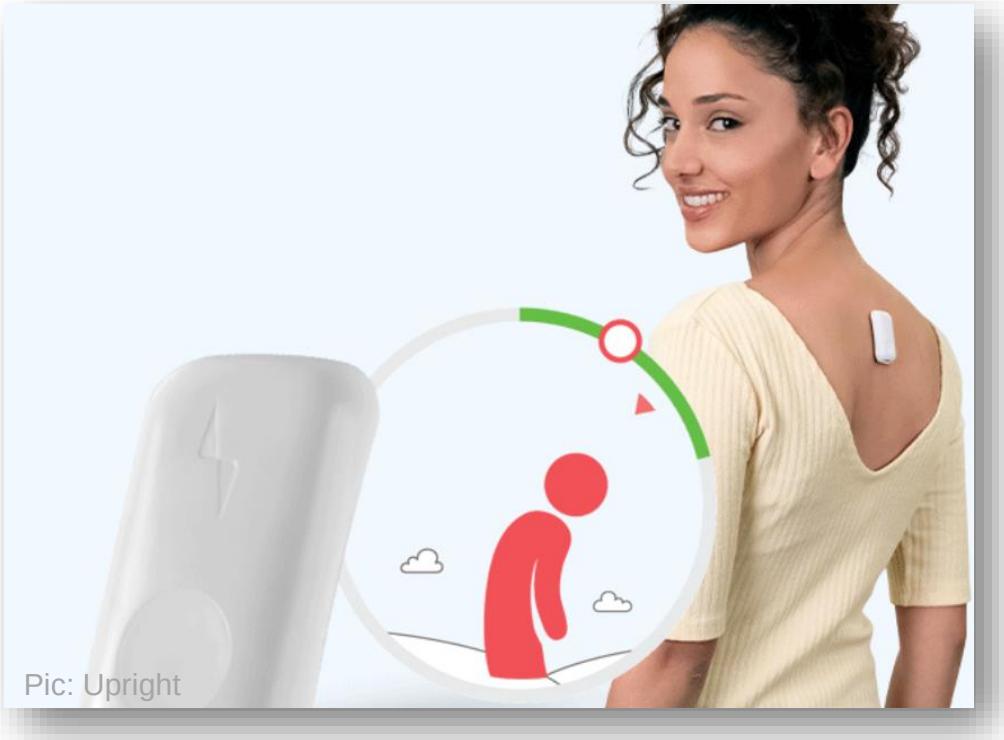
Smart Locks



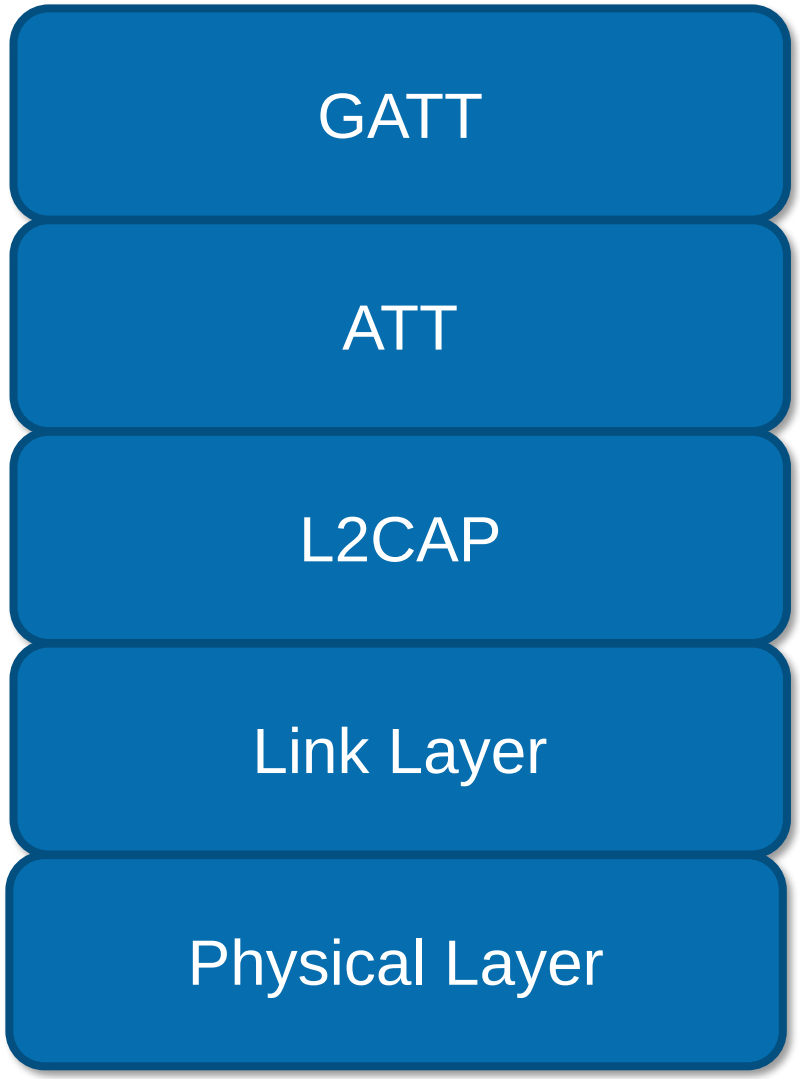
Vehicles

Showcase: UprightGo

- This gadget can be attached to your neck
- It measures your posture and tells the phone via BLE
- The phone let's the device vibrate if your posture is bad
- Project Page: https://www.uprightpose.com


Pic: Upright

Bluetooth Low Energy Protocol Stack




```
▸ Frame 16: 52 bytes on wire (416 bits), 52 bytes captured (416 bits)
▾ Nordic BLE Sniffer
    Board: 220
  ▸ Header Version: 1, Packet counter: 0
    Length of packet: 28
  ▸ Flags: 0x01
    Channel: 8
    RSSI (dBm): 0
    Event counter: 11
    Delta time (µs end to start): 0
    [Delta time (µs start to start): 144]
▾ Bluetooth Low Energy Link Layer
    Access Address: 0xaf9a83d2
    [Master Address: 5b:e3:cc:ea:83:81 (5b:e3:cc:ea:83:81)]
    [Slave Address: ca:4d:10:ba:09:73 (ca:4d:10:ba:09:73)]
  ▸ Data Header: 0x1a02
    [L2CAP Index: 10]
    CRC: 0x000000
▾ Bluetooth L2CAP Protocol
    Length: 22
    CID: Attribute Protocol (0x0004)
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Read By Group Type Response (0x11)
    Length: 20
  ▾ Attribute Data, Handle: 0x0920, Group End Handle: 0xffff, UUID128: Unknown
    ▸ Handle: 0x0920 (Unknown)
      Group End Handle: 0xffff
      UUID: 42234223422342234223422300022342
    [UUID: GATT Primary Service Declaration (0x2800)]
    [Request in Frame: 15]
```

These layers can also be seen in Wireshark

Physical Layer

- Operates on the unlicensed 2.5 GHz ISM Band
- 40 times 2 MHz channels (2402 MHz to 2480 MHz)
- Access Scheme (Sharing the same medium)
  - Frequency Division Multiple Access (FDMA)
  - Time Division Multiple Access (TDMA)
- Different frequencies on different time slots

Physical Layer

- 3 Advertising Channels
  - 37, 38, 39
- 37 Data Channels
  - 0 to 36

Channel number

PHY Channel	RF Center Frequency	Channel Index	Physical Channel Type	
			Primary Advertising	All others
0	2402 MHz	37	●	
1	2404 MHz	0		●
2	2406 MHz	1		●
...	...	...	...	...
11	2424 MHz	10		●
12	2426 MHz	38	●	
13	2428 MHz	11		●
14	2430 MHz	12		●
...	...	...	...	...
38	2478 MHz	36		●
39	2480 MHz	39	●	

Table 1.2: Mapping of PHY channel to physical channel index and channel type

Link Layer

- Responsible for advertising, scanning, creating/maintaining connections
- Package format for LE Uncoded physical layer

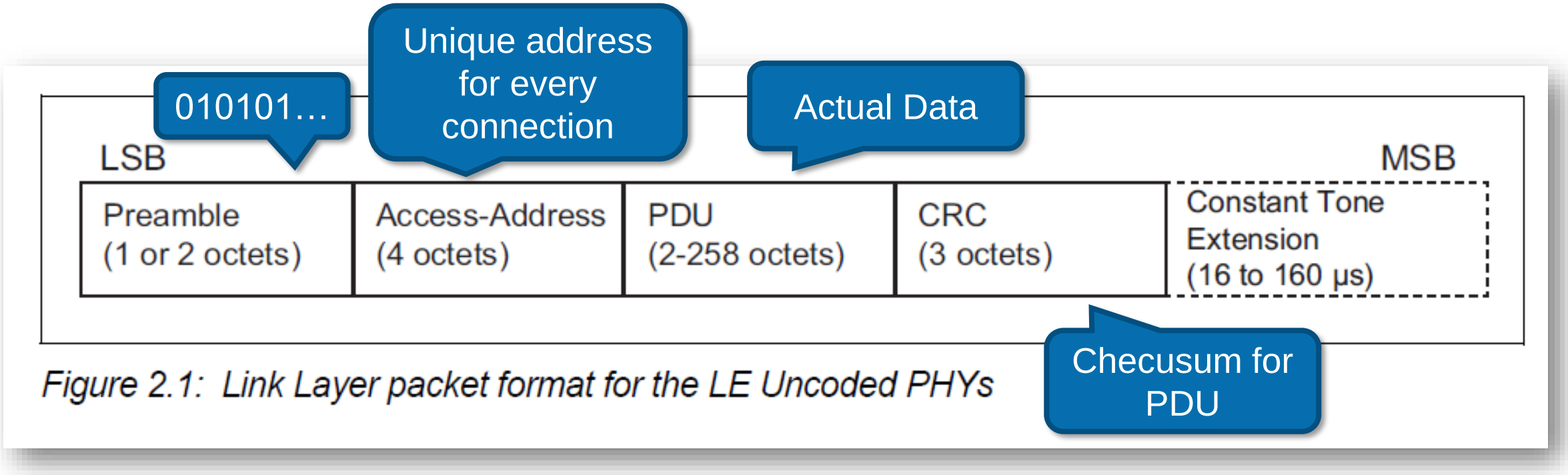
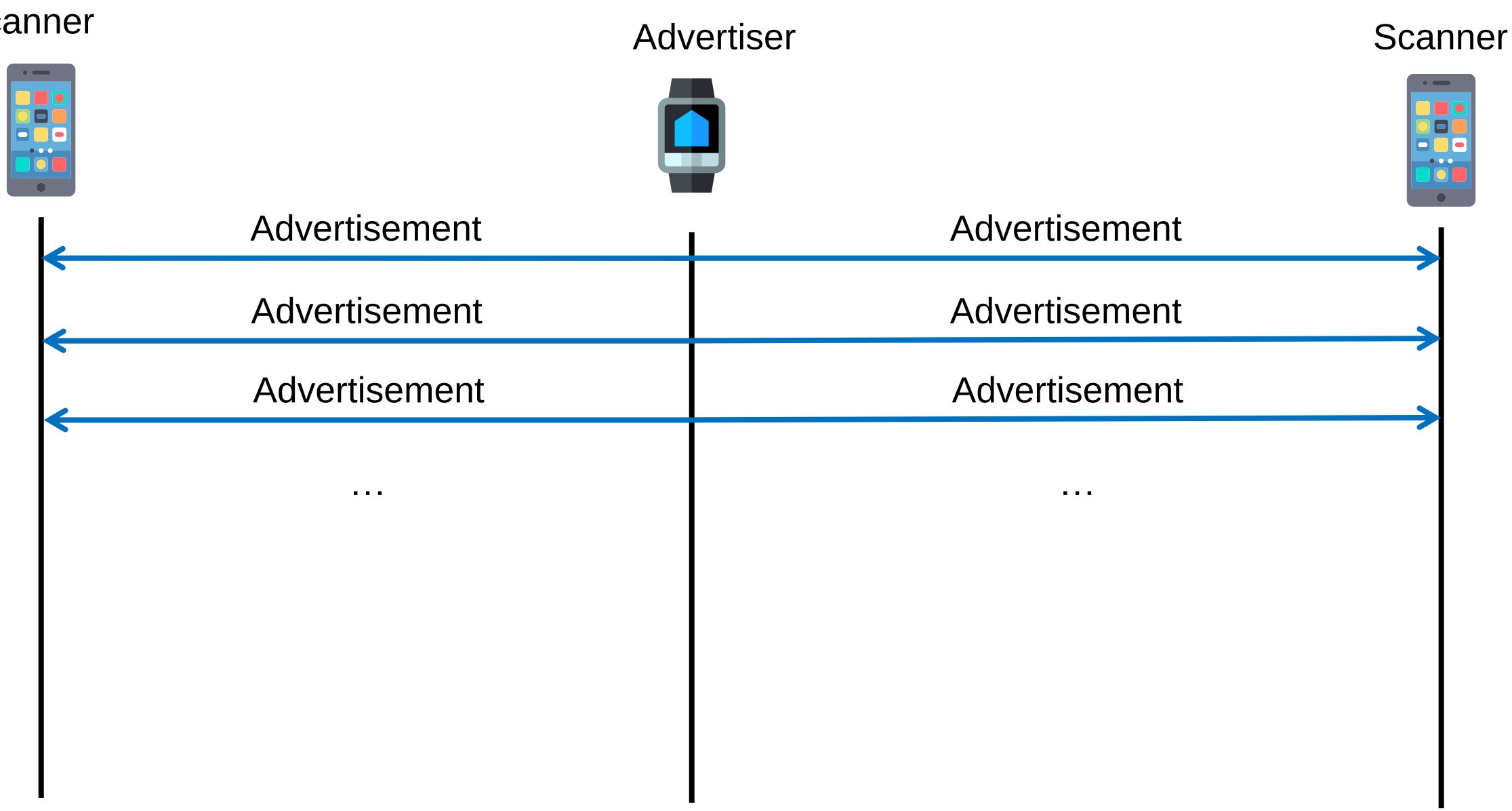


Figure 2.1: Link Layer packet format for the LE Uncoded PHYs

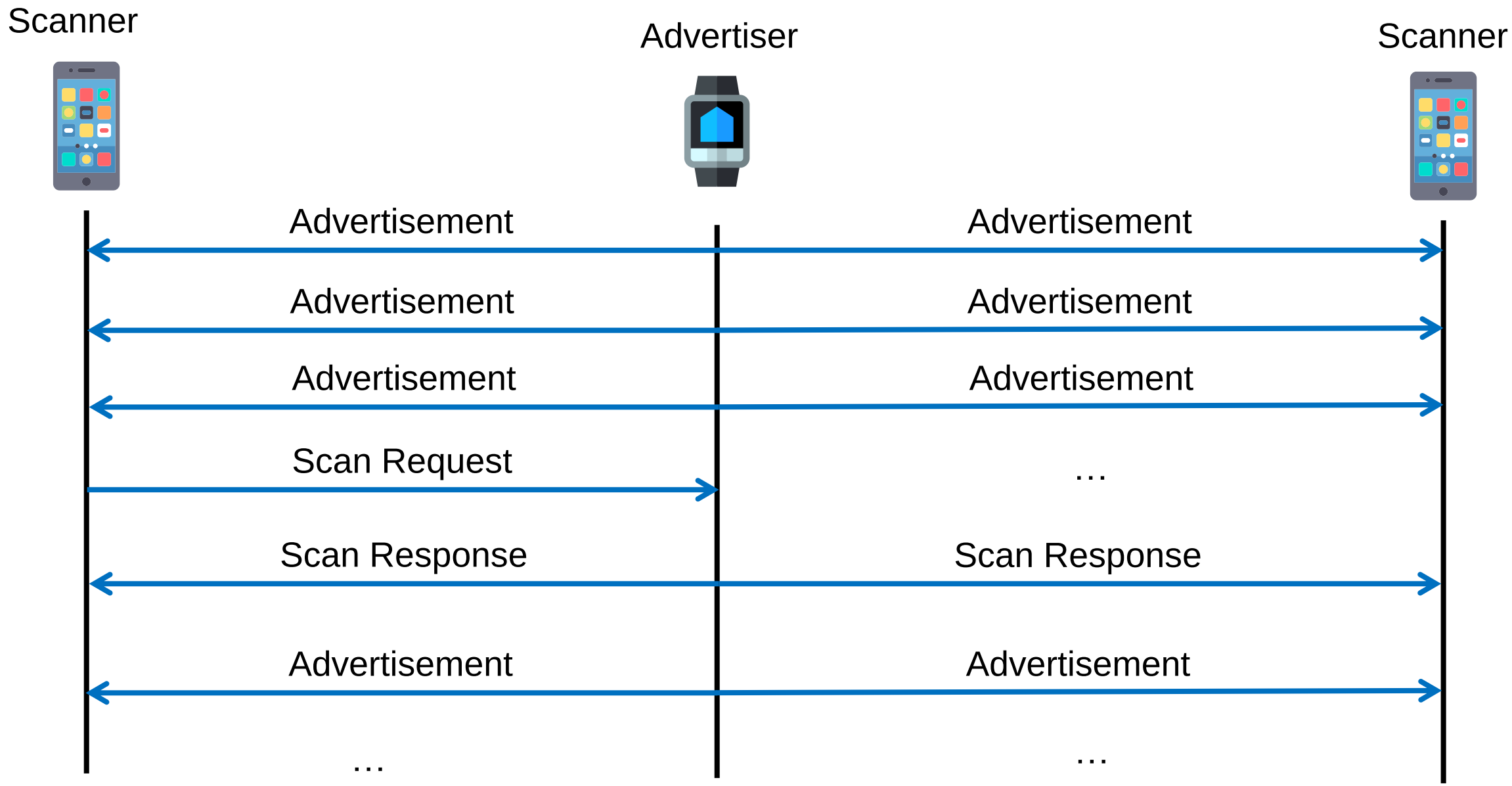
Link Layer

- Conflict if multiple devices send on the same channel at the same time
- Frequency hopping to combat interference and fading
- One data packet per channel at a given time
- Channel Selection Algorithm (CSA #1)
- Frequency hopping scheme (sent in connection request)
  - Channel Matrix: Which frequencies will be used? (e.g. use all 37 data channels)
  - Hop Increment: Next channel = Channel + Hop Increment (mod 37)
  - Hop Interval: Time between Hops

Passive Scanning



Active Scanning



Packet: Scan Request

No.	Time	Source	Destination	Protocol	Length	Comment	Info
10498	365.649479	TexasIns_44:ee:30	Broadcast	LE LL	45		ADV_IND
10499	365.751589	TexasIns_44:ee:30	Broadcast	LE LL	45		ADV_IND
10500	365.756044	6d:89:e8:1e:29:ad	TexasIns_44:ee:30	LE LL	38		SCAN_REQ
10501	365.756802	TexasIns_44:ee:30	Broadcast	LE LL	52		SCAN_RSP

```
▸ Frame 10500: 38 bytes on wire (304 bits), 38 bytes captured (304 bits)
  DLT: 157, Payload: nordic_ble (Nor
▸ Nordic BLE Sniffer
▾ Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  ▾ Packet Header: 0x0c43 (PDU Type: SCAN_REQ, ChSel: #1, TxAdd: Random, RxAdd: Public)
        .... 0011 = PDU Type: SCAN_REQ (0x3)
        ...0 .... = RFU: 0
        ..0. .... = Channel Selection Algorithm: #1
        .1.. .... = Tx Address: Random
        0... .... = Rx Address: Public
        Length: 12
    Scanning Address: 6d:89:e8:1e:29:ad (6d:89:e8:1e:29:ad)
    Advertising Address: TexasIns_44:ee:30 (30:45:11:44:ee:30)
    CRC: 0x453ad6
```

Fixed access address for advertising packets

Bluetooth MAC Address

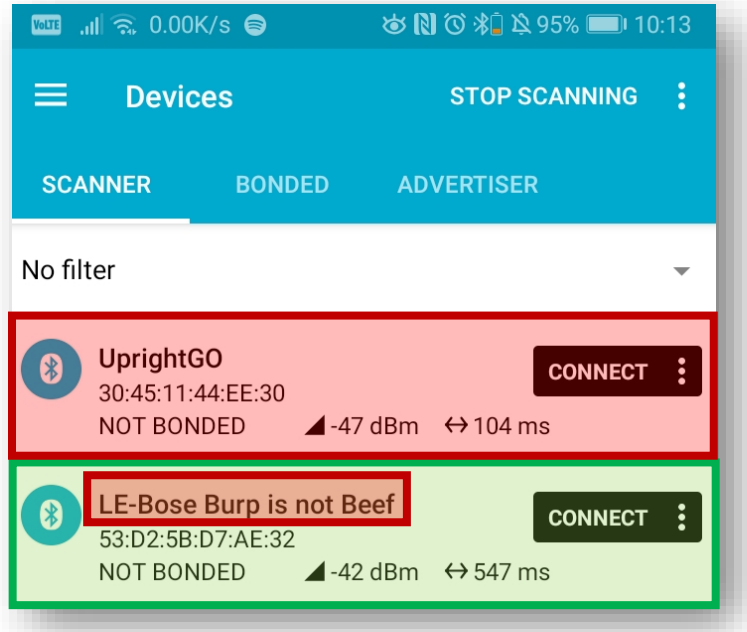
Modern Bluetooth stacks randomize the MAC address for privacy reasons (user tracking)

Bluetooth LE Privacy

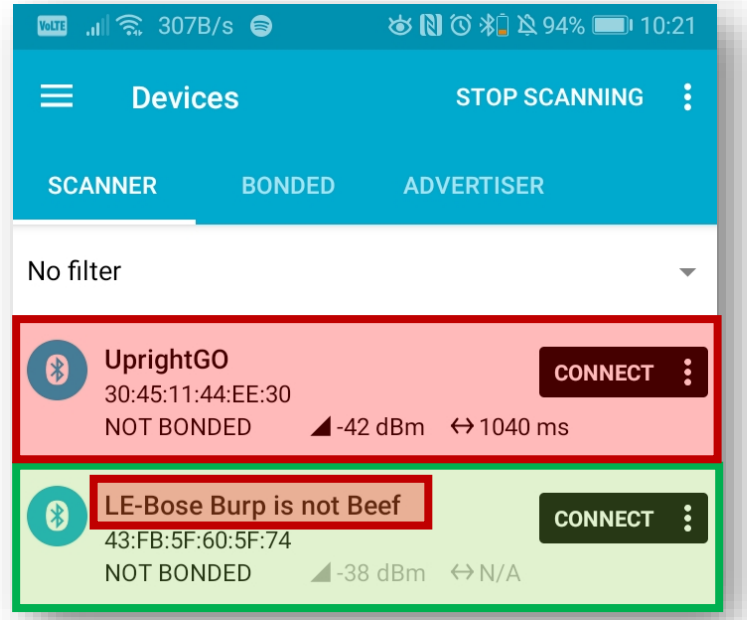
- Bluetooth LE Privacy exists since Bluetooth 4.0
- Implemented to avoid user tracking
- Random MAC address is used
- Changes the MAC address at a time interval specified by the manufacturer
- Identity Resolution Key (IRK) is exchanged during pairing / bonding process
- Paired devices can convert random MAC addresses back to real MAC address

nRF Connect App

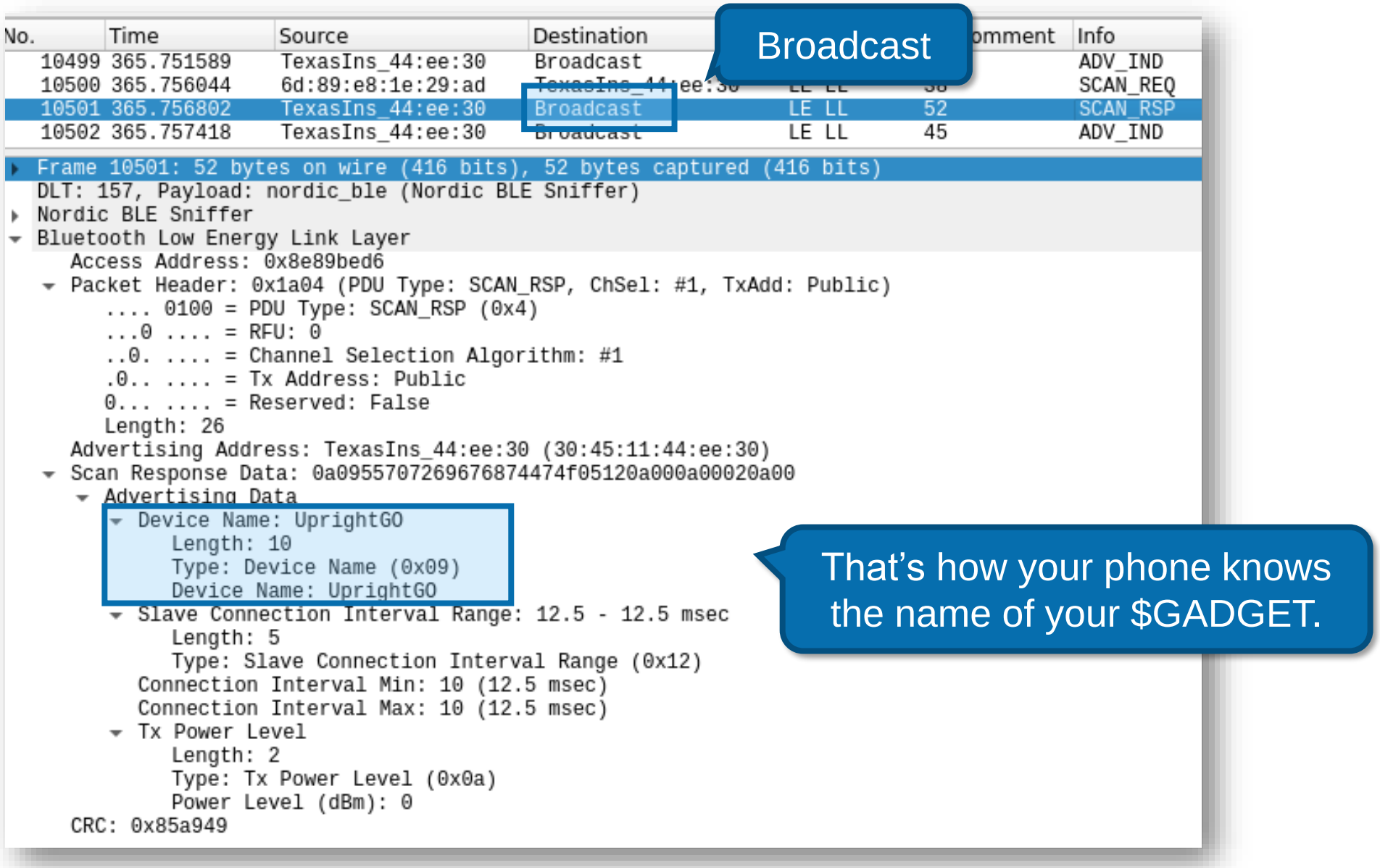
Bose headphones can be controlled via BLE



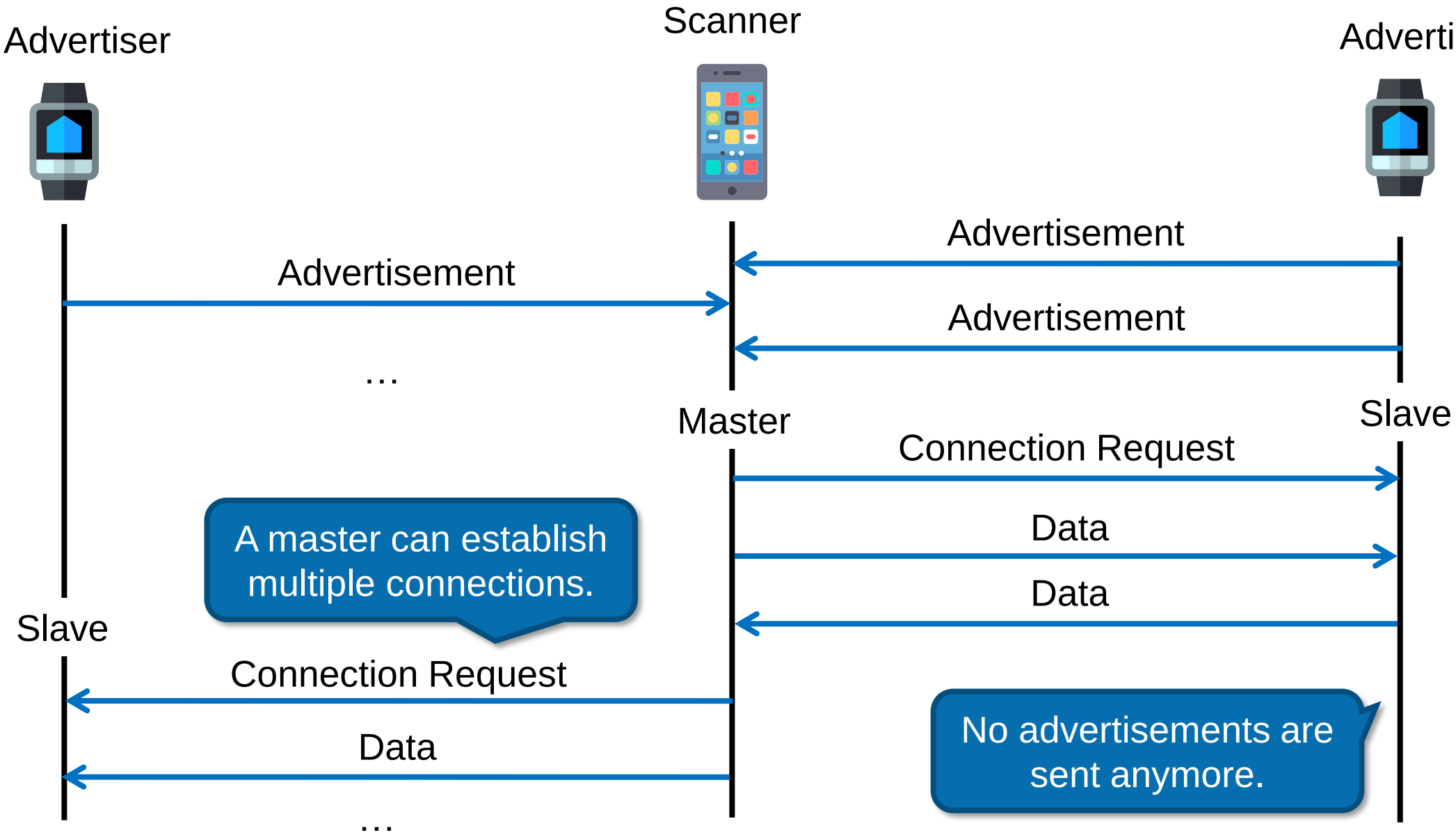
Other information could be used for user tracking like the device name.



Packet: Scan Response



Connection Establishement



Packet: Connection Request

No.	Time	Source	Destination	Protocol	Length	Comment	Info
10656	407.794530	TexasIns_44:ee:30	Broadcast	LE LL	45		ADV_IND
10657	407.906622	TexasIns_44:ee:30	Broadcast	LE LL	45		ADV_IND
10658	407.907744	6d:89:e8:1e:29:ad	TexasIns_44:ee:30	LE LL			NNECT_REQ
10659	407.908525	Master_0xaf9a9c24	Slave_0xaf9a9c24	LE LL			pty PDU
10660	407.947269	Master_0xaf9a9c24	Slave_0xaf9a9c24	LE LL	26		Empty PDU

Unicast

```
Frame 10658: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
DLT: 157, Payload: nordic_ble (Nordic BLE Sniffer)
Nordic BLE Sniffer
Bluetooth Low Energy Link Layer
   Access Address: 0x8e89bed6
   Packet Header: 0x2245 (PDU Type: CONNECT_REQ, ChSel: #1, TxAdd: Random, RxAdd: Public)
      .... 0101 = PDU Type: CONNECT_REQ (0x5)
      ...0 .... = RFU: 0
      ..0. .... = Channel Selection Algorithm: #1
      .1.. .... = Tx Address: Random
      0... .... = Rx Address: Public
      Length: 34
   Initator Address: 6d:89:e8:1e:29:ad (6d:89:e8:1e:29:ad)
   Advertising Address: TexasIns_44:ee:30 (30:45:11:44:ee:30)
   Link Layer Data
      Access Address: 0xaf9a9c24
      CRC Init: 0xdceebf
      Window Size: 3 (3.75 msec)
      Window Offset: 1 (1.25 msec)
      Interval: 39 (48.75 msec)
      Latency: 0
      Timeout: 500 (5000 msec)
      Channel Map: ffffffff1f
      ...0 0111 = Hop: 7
      001. .... = Sleep Clock Accura                (1)
   CRC: 0xe91210
```

Access Address

Hop Interval

Channel Map

Hop Increment

Frequency Hopping Scheme

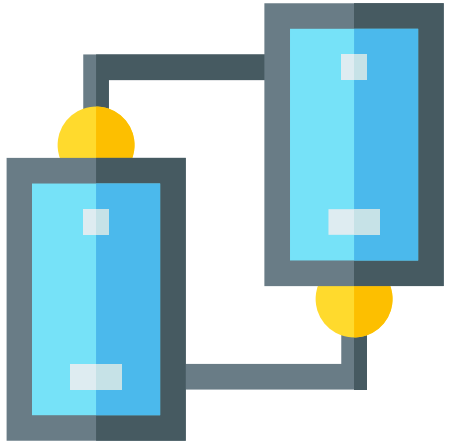
Packet: Data (Empty)

No.	Time	Source	Destination	Protocol	Length	Comment	Info
10656	407.794530	TexasIns_44:ee:30	Broadcast	LE LL	45		ADV_IND
10657	407.906622	TexasIns_44:ee:30	Broadcast	LE LL	45		ADV_IND
10658	407.907744	6d:89:e8:1e:29:ad	TexasIns_44:ee:30	LE LL	60		CONNECT_REQ
10659	407.908525	Master_0xaf9a9c24	Slave_0xaf9a9c24	LE LL	26		Empty PDU
10660	407.947269	Master_0xaf9a9c24	Slave_0xaf9a9c24	LE LL	26		Empty PDU
10661	407.947714	Slave_0xaf9a9c24	Master_0xaf9a9c24	LE LL	26		Empty PDU

```
▸ Frame 10659: 26 bytes on wire (208 bits), 26 bytes captured (208 bits)
  DLT: 157, Payload: nordic_ble (Nordic BLE Sniffer)
▸ Nordic BLE Sniffer
▾ Bluetooth Low Energy Link Layer
    Access Address: 0xaf9a9c24
    [Master Address: 6d:89:e8:1e:29:ad (6d:89:e8:1e:29:ad)]
    [Slave Address: TexasIns_44:ee:30 (30:45:11:44:ee:30)]
  ▾ Data Header: 0x0001
      .... ..01 = LLID: Continuation fragment of an L2CAP message, or an Empty PDU (0x1)
      .... .0.. = Next Expected Sequence Number: 0
      .... 0... = Sequence Number: 0 [OK]
      ...0 .... = More Data: False
      000. .... = RFU: 0
    Length: 0
    CRC: 0x659579
```

Access Address is used for further communication.

Attribute Protocol (ATT)



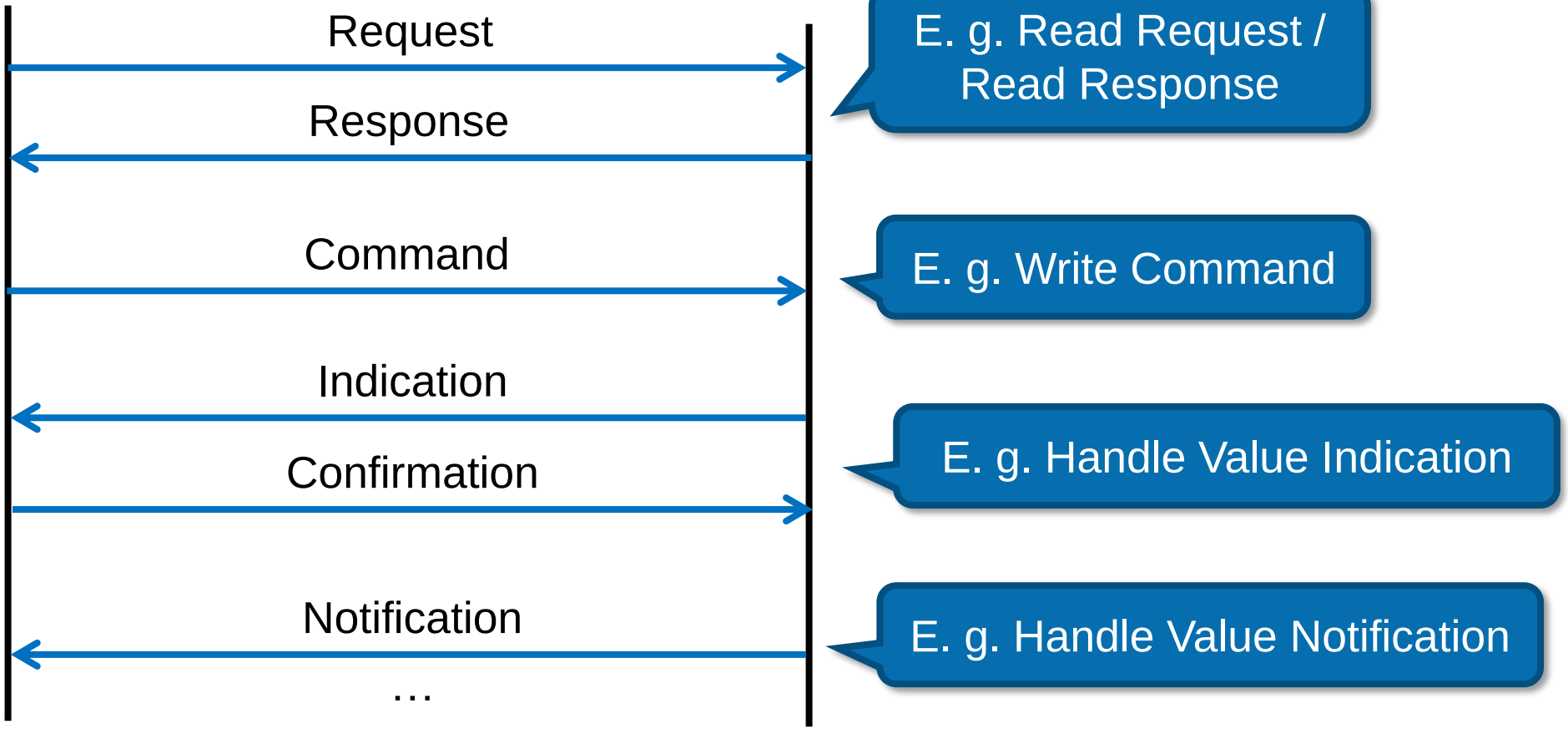
- Peer-to-peer protocol between attribute server and attribute client
- The master is the ATT client
  - The ATT client can send ATT commands, requests and confirmations
- The slave is the ATT server
  - The ATT server can send ATT sends responses, notifications and indications
- Based on attributes
  - Attribute Type (16 or 128 bit UUID)
  - 16 bit handle
  - Length + Value

The handle is used to address an attribute

Attribute Protocol (ATT)

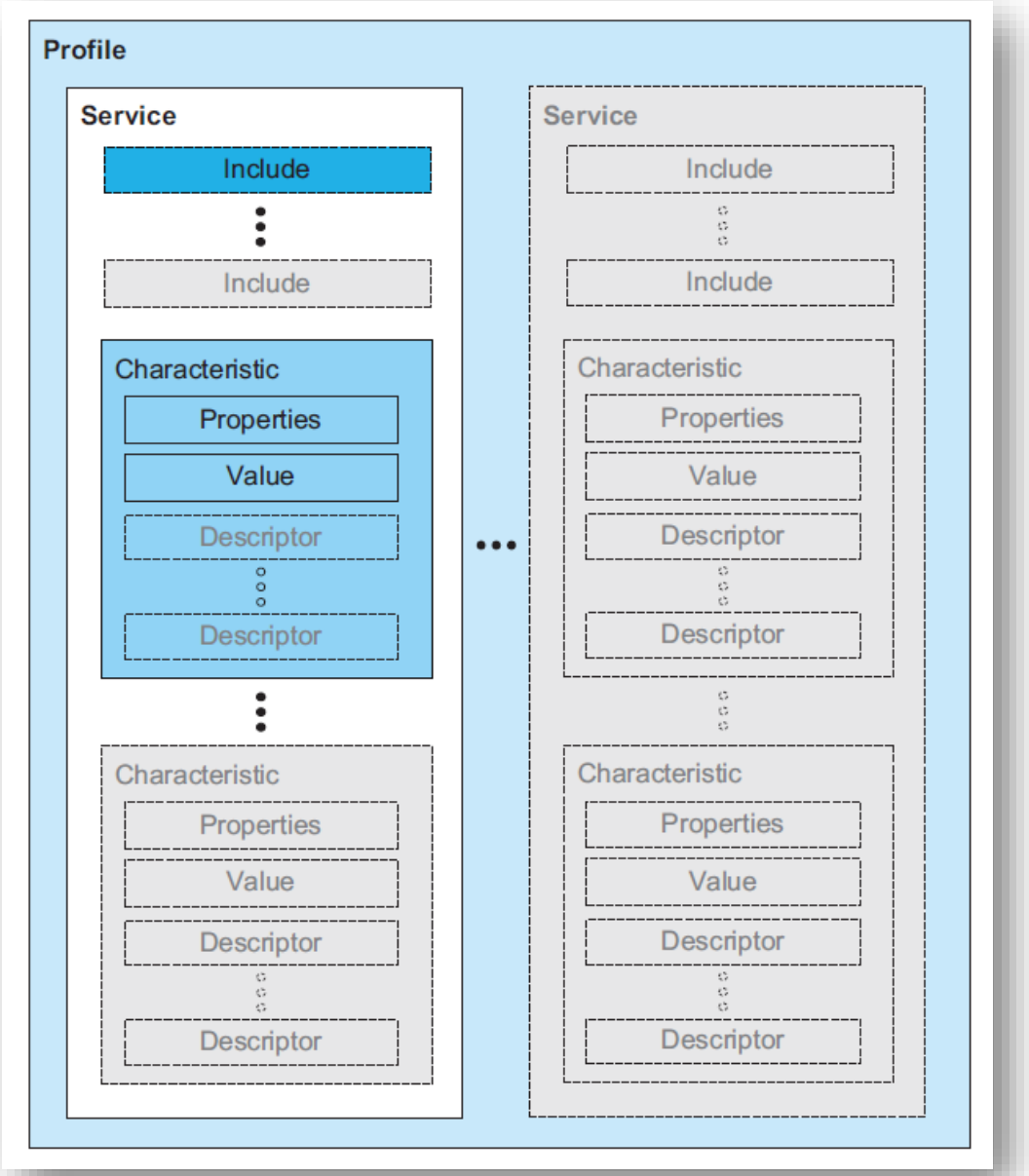
Master (ATT Client)

Slave (ATT Server)



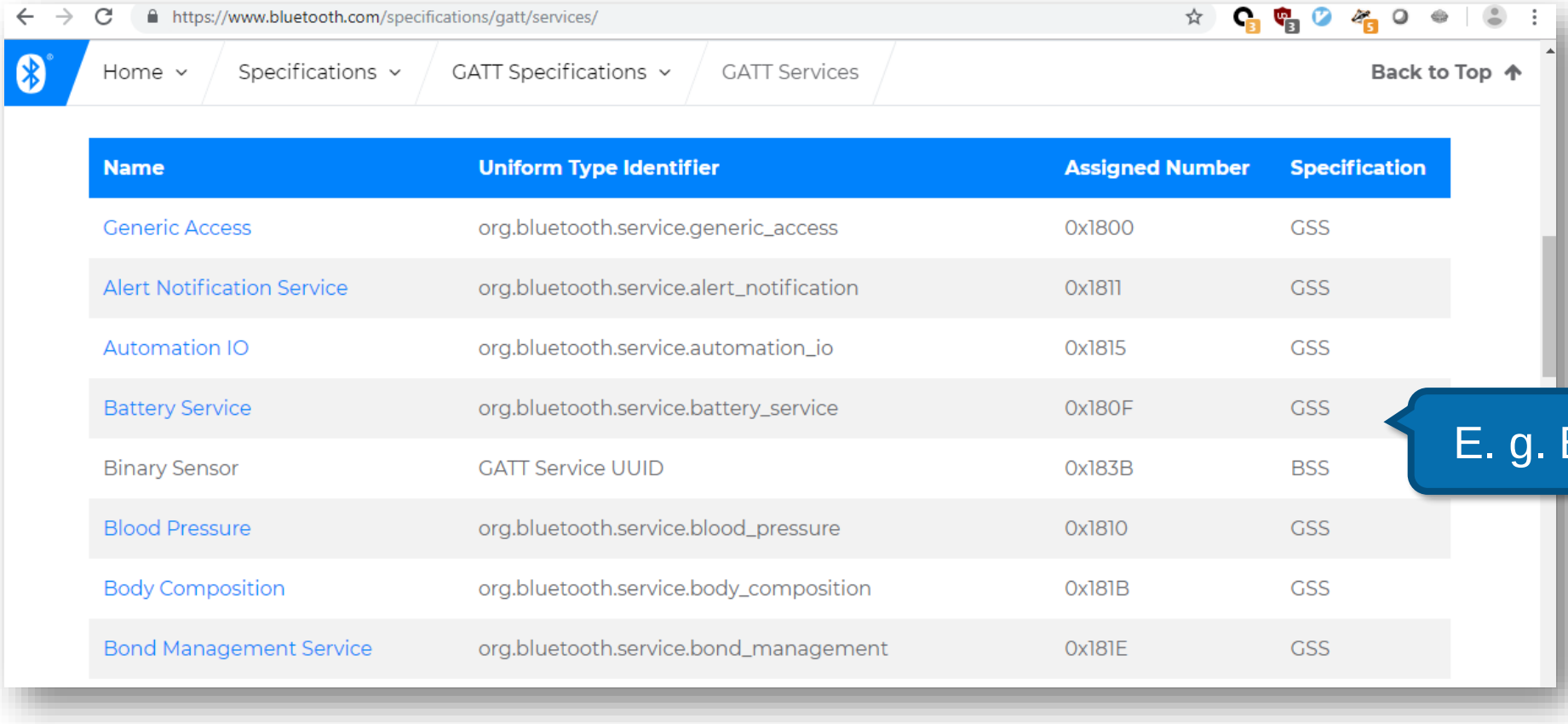
Generic Attribute Profile (GATT)

- Functionality of the ATT server and optionally the ATT client
- Hierarchy of services & characteristics
- Interface for discovering, reading, writing and indicating services
- Multiple services containing multiple characteristics
- Think of a «web service»
  - Profile / Service ≈ Description
  - Characteristics ≈ Webservice Endpoints



GATT Services

- Collection of characteristics (think of «categories»)
- Identified by a UUID (Standardised services: 16 bit; custom services: 128 bit)
- Standardised services: https://www.bluetooth.com/specifications/gatt/services

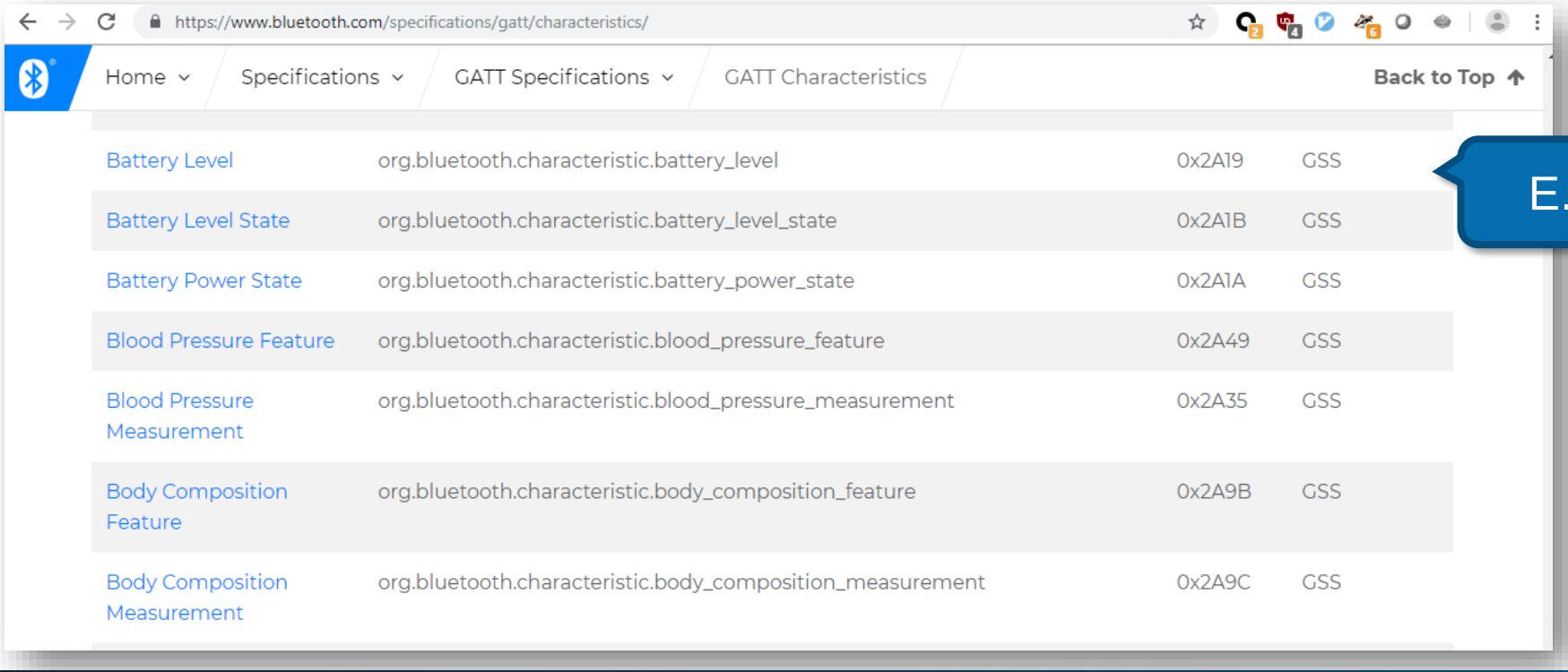


Name	Uniform Type Identifier	Assigned Number	Specification
Generic Access	org.bluetooth.service.generic_access	0x1800	GSS
Alert Notification Service	org.bluetooth.service.alert_notification	0x1811	GSS
Automation IO	org.bluetooth.service.automation_io	0x1815	GSS
Battery Service	org.bluetooth.service.battery_service	0x180F	GSS
Binary Sensor	GATT Service UUID	0x183B	BSS
Blood Pressure	org.bluetooth.service.blood_pressure	0x1810	GSS
Body Composition	org.bluetooth.service.body_composition	0x181B	GSS
Bond Management Service	org.bluetooth.service.bond_management	0x181E	GSS

E. g. Battery Service

GATT Characteristics

- Actual data
- Read/Write/...
- Identified by a UUID (Standardised characteristics: 16 bit; custom characteristics: 128 bit)
- Defined in the service specification



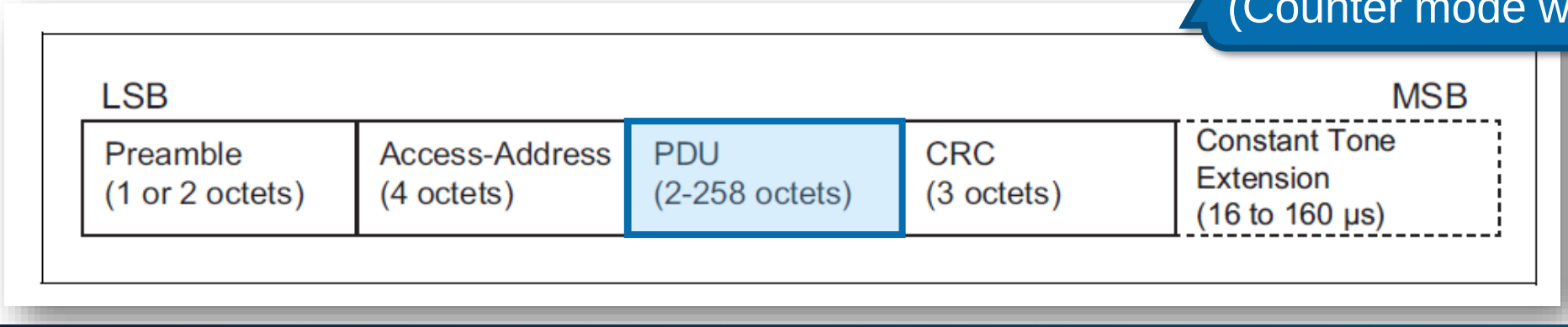
E. g. Battery Level

BLE Security Mechanisms

BLE Security

- Security is optional!
  - By default, there is no authentication & no encryption!
- Authentication & encryption is possible
- Authentication
  - Used to ensure that the connection is established to the correct device
  - Protects against active Man-in-the-Middle attacks
- Encryption
  - Used to ensure that noone can read the transmitted data
  - Protects against passive Man-in-the-Middle attacks

AES-128 CCM
(Counter mode with CBC-MAC)



Security Manager (SM)

- Defines pairing, authentication, encryption, key exchange/distribution, …
- Security Manager Protocol (SMP): Peer-to-peer protocol used to generate encryption keys
- Custom Key Exchange Protocol (3 phases)

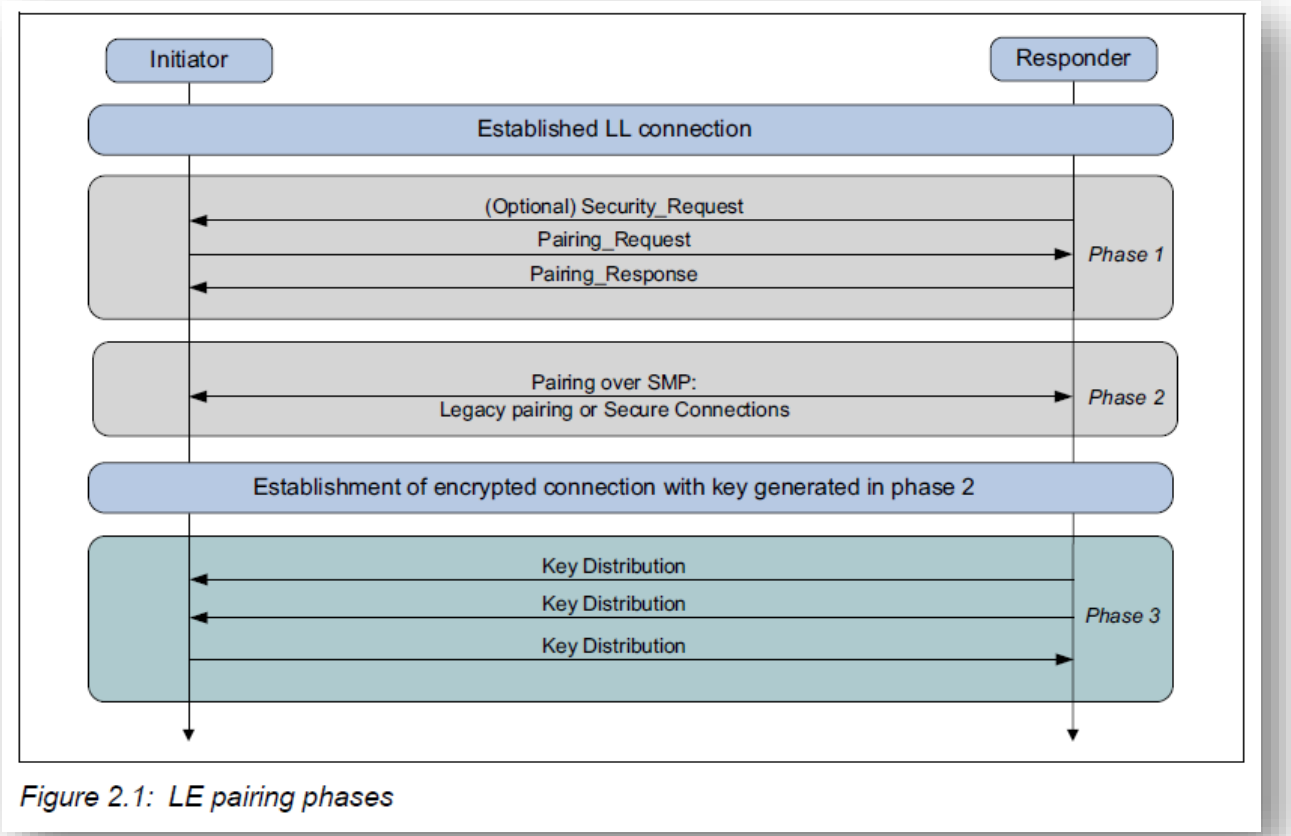
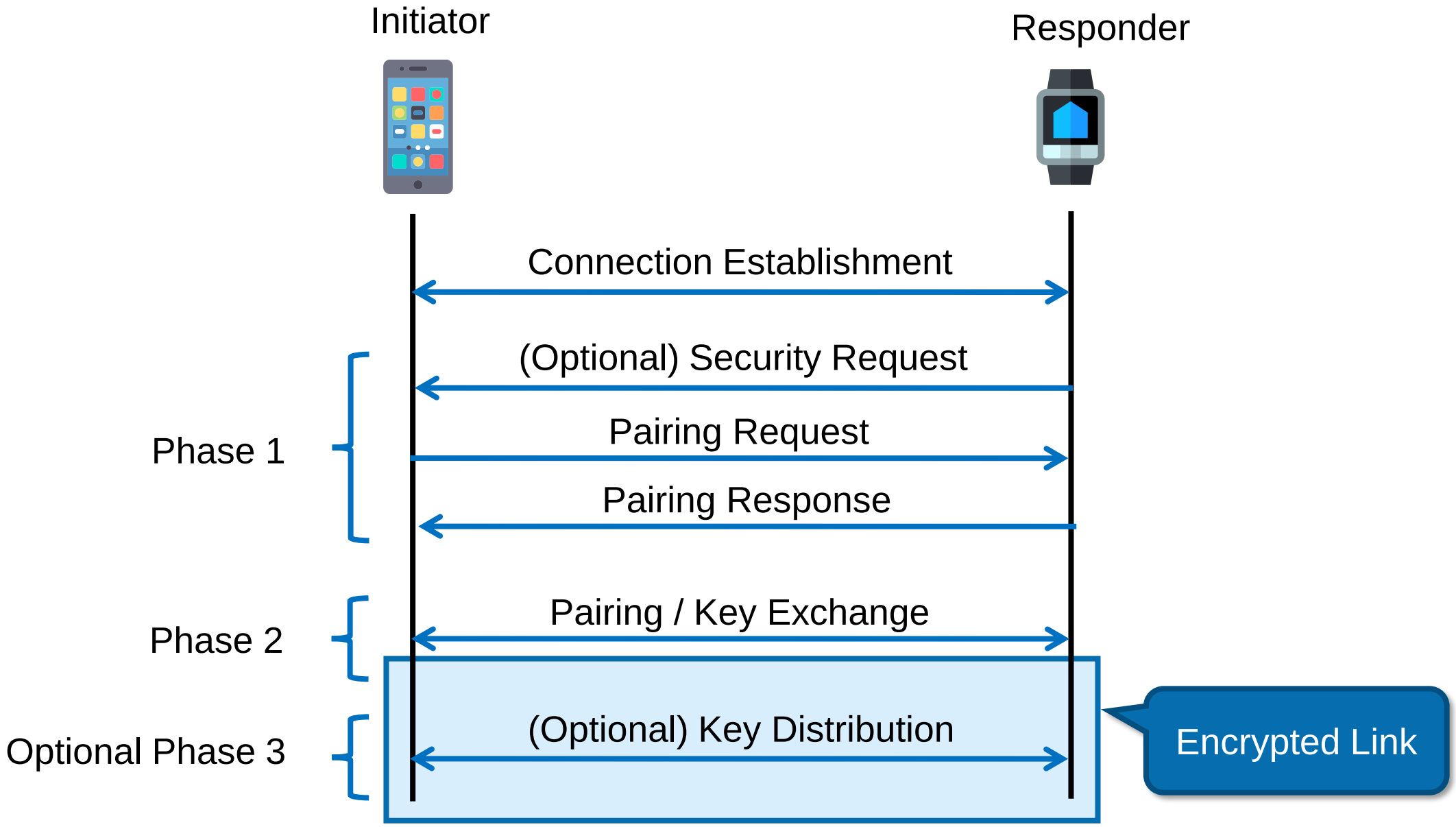


Figure 2.1: LE pairing phases

Pairing = Key Exchange Mechanisms

Pairing Phases



Pairing Phases

- Phase 1
  - Which key generation / pairing method is used?
- Phase 2
  - Key Generation
  - «LE Legacy Pairing»
    - Both devices generate a Short Term Key (STK)
    - Key generation method depends on the pairing method
  - «LE Secure Connections»
    - Long Term Key (LTK) Generation
- Phase 3 (Optional)
  - Transport Specific Key Distribution
  - Used for Bonding

Since Bluetooth 4.2 (2014)

Pairing Phase 1: Pairing Feature Exchange

The devices tell each other which pairing features they support.

Capability Flags	Description
No input	No method to indicate yes or no
Yes / No	There is a method to indicate yes or no
Keyboard	There is a keyboard with the number 0 to 9 and a method to indicate yes or no
No output	Not possible to display a 6 digit number
Numeric output	Possible to display a 6 digit number

Other Flags	Description
OOB	Flag whether out-of-band authentication data is present or not
Bonding	Flag whether long-term key should be saved for later use
MITM	Flag whether man-in-the-middle protection is requested or not (request Authenticated security property for the legacy pairing STK / Secure Connection LTK)
SC	Flag whether LE Secure Connections can be used
KC	Keypress flag used for the Passkey Entry pairing method (generate keypress notifications and send via SMP)

Pairing Phase 1: Pairing Method Selection

- Choose key generation method

LE Legacy Pairing

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Check MITM		
	OOB Not Set	Check MITM	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

Table 2.6: Rules for using Out-of-Band and MITM flags for LE legacy pairing

LE Secure Connections

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Use OOB		
	OOB Not Set	Use OOB	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

Table 2.7: Rules for using Out-of-Band and MITM flags for LE Secure Connections pairing

Pairing Phase 1: Pairing Method Selection

	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Display Only	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated
Display YesNo	Just Works Unauthenticated	Just Works (For LE Legacy Pairing) Unauthenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): responder displays, initiator inputs Authenticated
		Numeric Comparison (For LE Secure Connections) Authenticated			Numeric Comparison (For LE Secure Connections) Authenticated

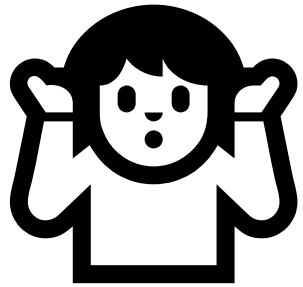
Table 2.8: Mapping of IO capabilities to key generation method

	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Keyboard Only	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator and responder inputs Authenticated	Just Works Unauthenticated	Passkey Entry: initiator displays, responder inputs Authenticated
NoInput NoOutput	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
Keyboard Display	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated
		Numeric Comparison (For LE Secure Connections) Authenticated			Numeric Comparison (For LE Secure Connections) Authenticated

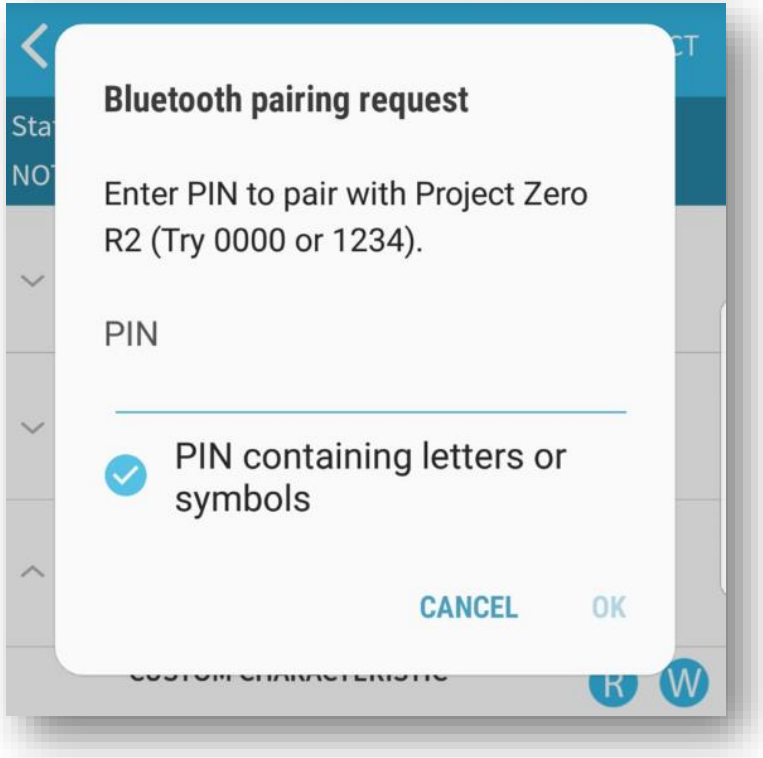
Table 2.8: Mapping of IO capabilities to key generation method

Pairing Methods

- Just Works
  - It just works, no user interaction needed
  - Unauthenticated!
  - No protection against active MITM

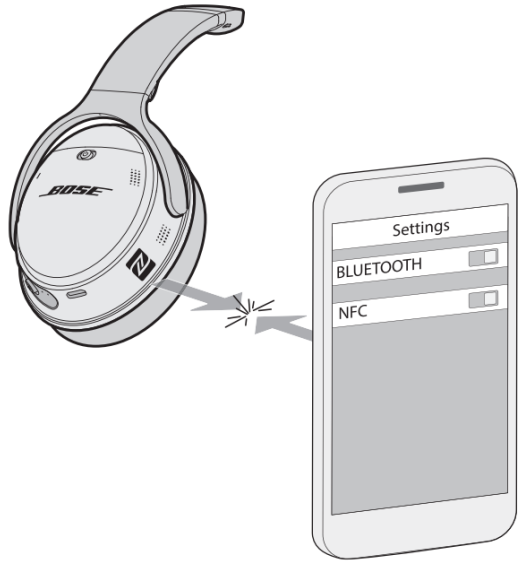


- Passkey Entry
  - One device generates and displays a number between 000000 and 999999
  - This number must be entered on the other device
  - Protects against active MITM (0.000001 succeeding probability)



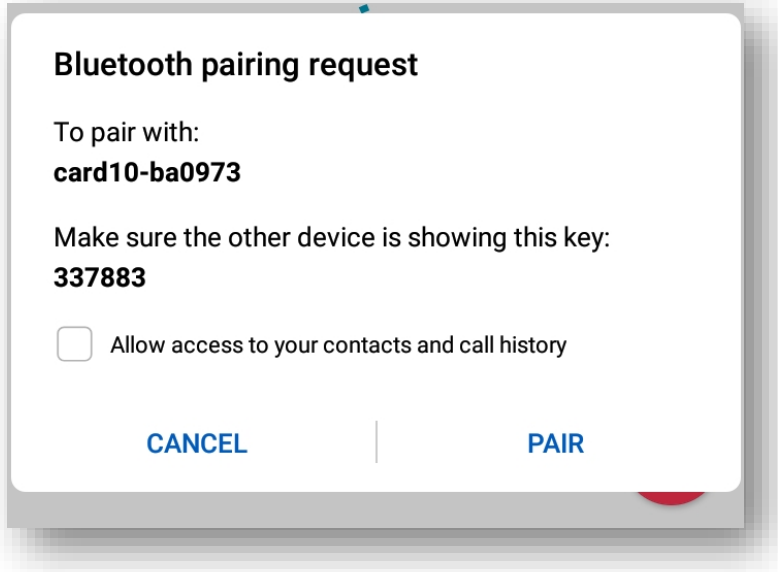
Pairing Methods

- Out of Band
  - Exchange of the key material out of band
  - E.g. via NFC, QR Codes, …
  - Protects against active MITM if the OOB mechanism is also MITM resistant



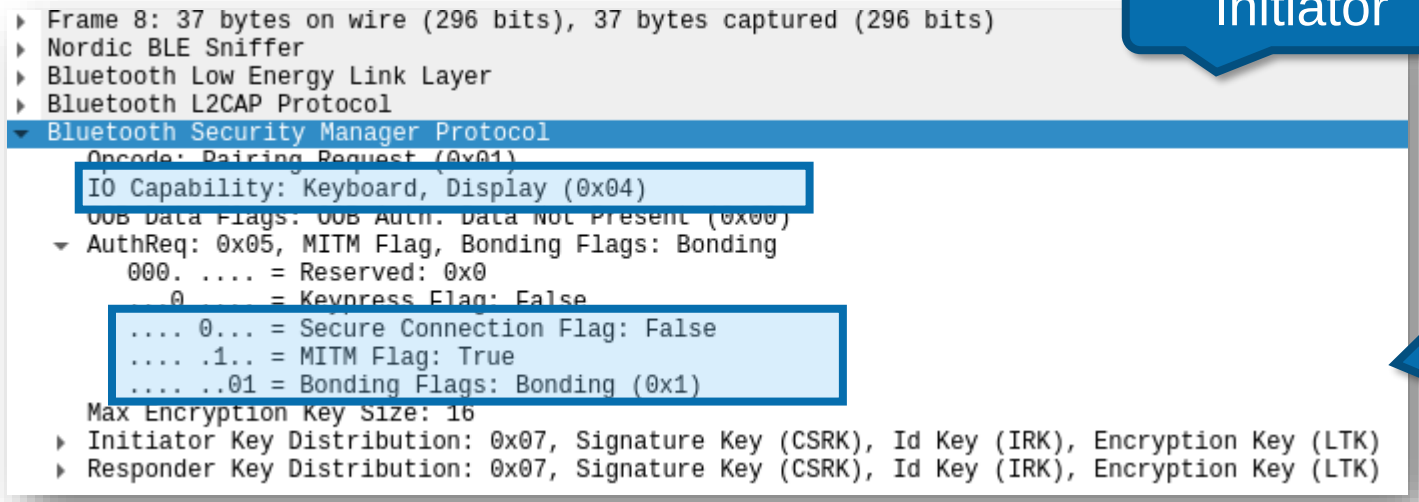
Pic: Bose

- Numeric Comparison
  - Only for LE Secure Connections
  - Both devices display the same agreed number that has to be acknowledged on both devices
  - Protects against active MITM



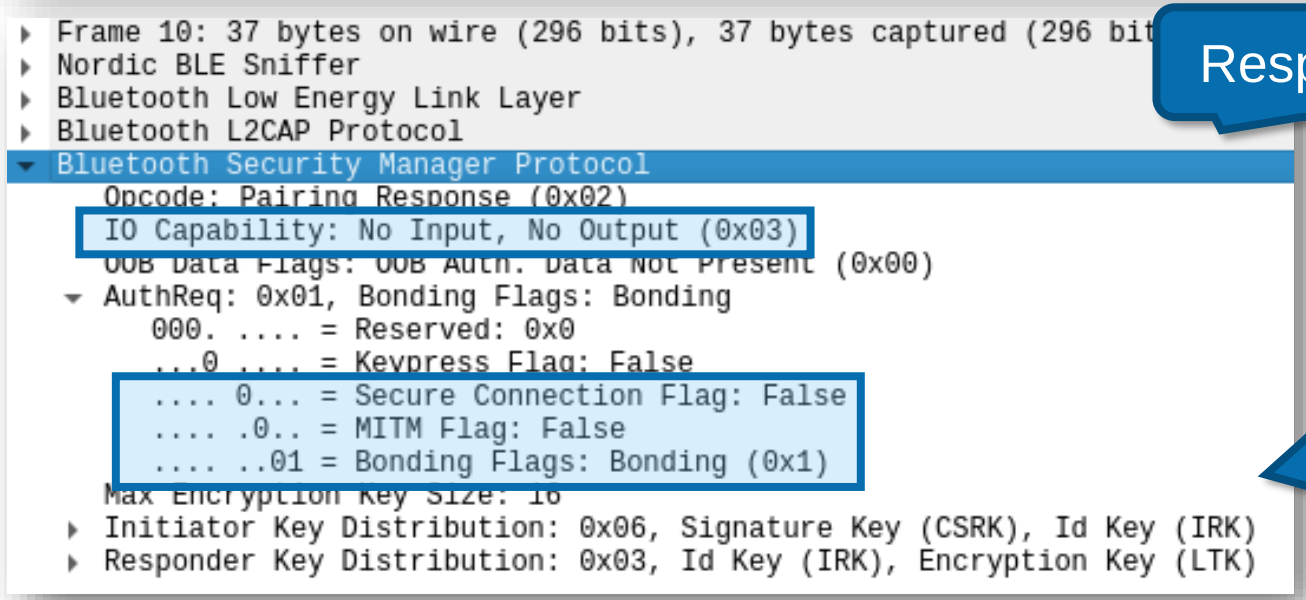
Pairing Phase 1: Example #1

Initiator



IO Capability: Keyboard, Display
Secure Connection: False
MITM Flag: True
Bonding Flag: True

Responder



IO Capability: No Input, No Output
Secure Connection: False
MITM Flag: False
Bonding Flag: True

Pairing Phase 1: Example #1

		Initiator			
		OOB Set	OOB Not Set	MITM Set	MITM Not Set
Responder	OOB Set	Use OOB	Check MITM		
	OOB Not Set	Check MITM	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

Table 2.6: Rules for using Out-of-Band and MITM flags for LE legacy pairing

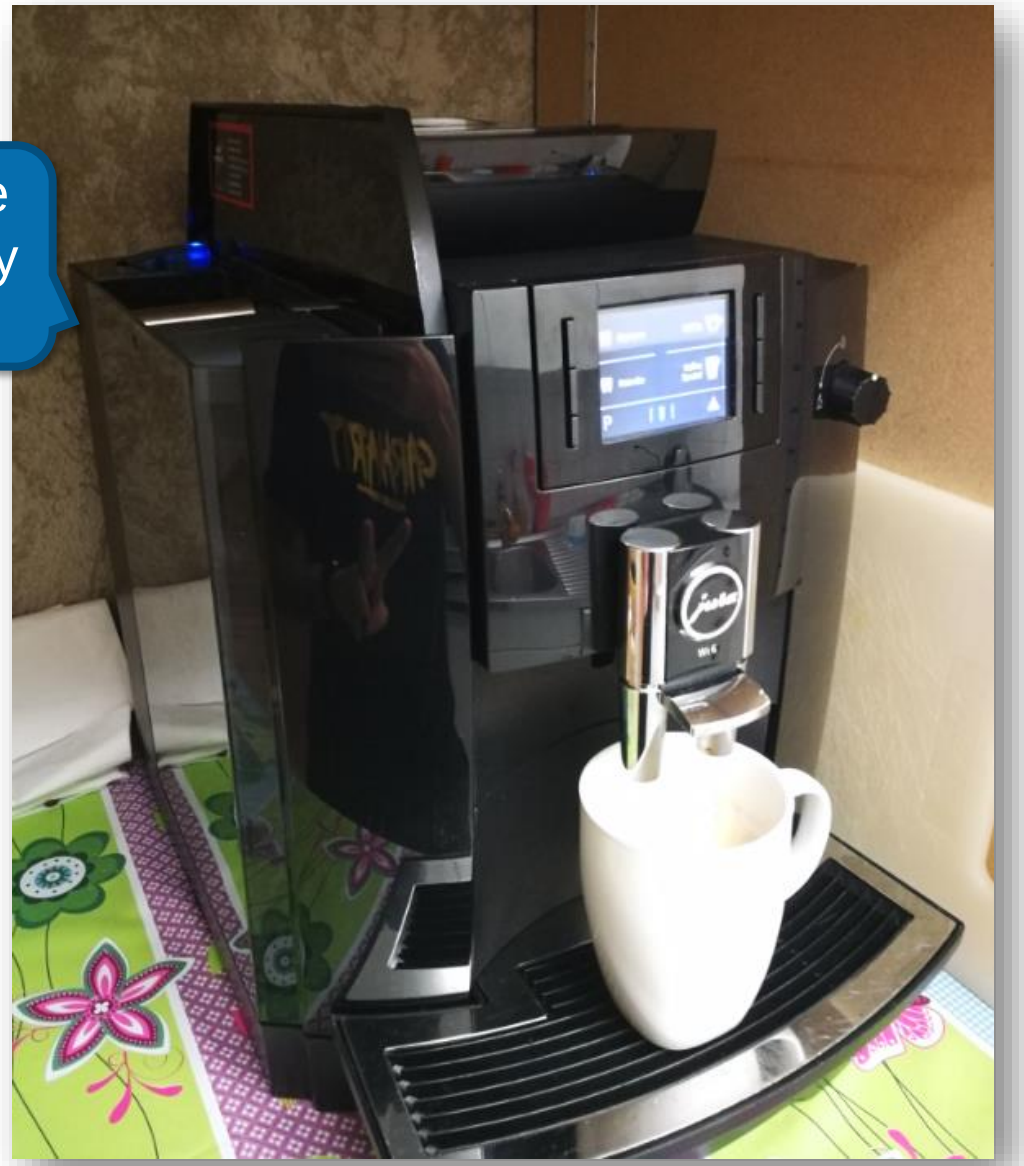
Selected Key Generation Method: Just Works

	Initiator				
Responder	DisplayOnly	Display YesNo	Keyboard Only	NoInput NoOutput	Keyboard Display
Keyboard Only	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry: initiator and responder inputs Authenticated	Just Works Unauthenticated	Passkey Entry: initiator displays, responder inputs Authenticated
NoInput NoOutput	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
Keyboard Display	Passkey Entry: initiator displays, responder inputs Authenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated	Passkey Entry: responder displays, initiator inputs Authenticated	Just Works Unauthenticated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenticated
		Numeric Comparison (For LE Secure Connections) Authenticated			Numeric Comparison (For LE Secure Connections) Authenticated

Table 2.8: Mapping of IO capabilities to key generation method

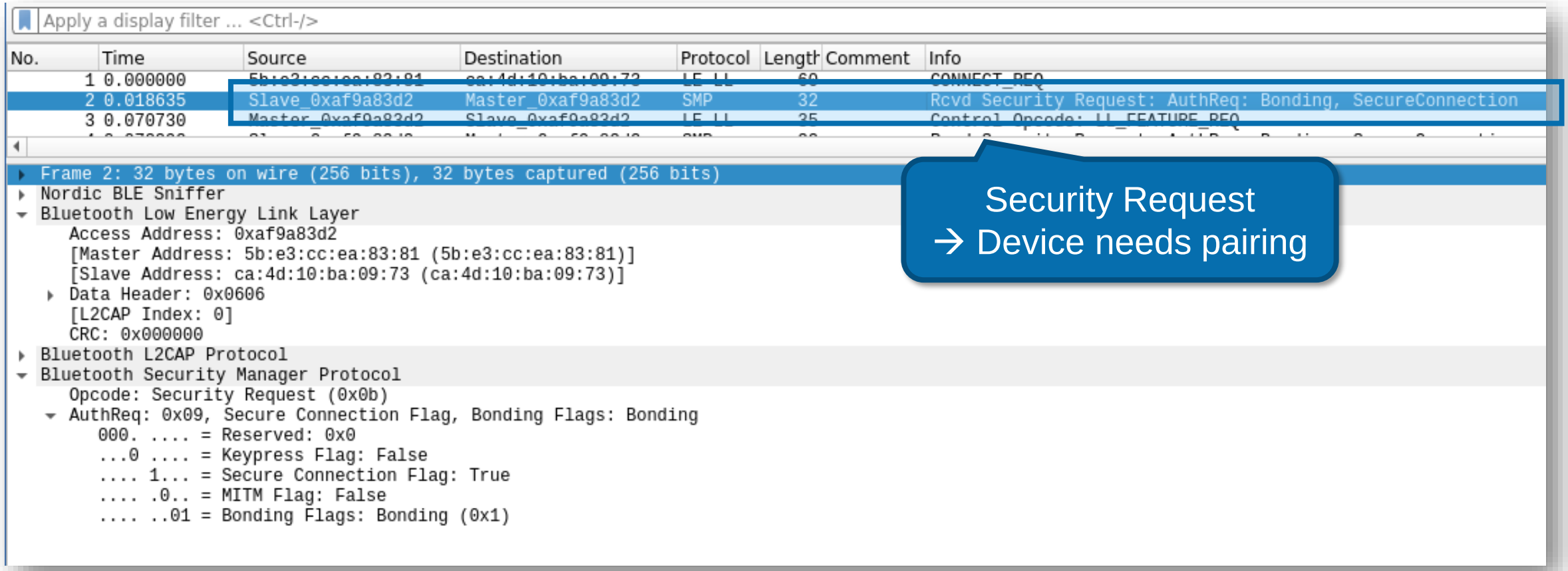
Pairing Phase 1: Example #1

In fact, our machine does not require any pairing at all 😒.

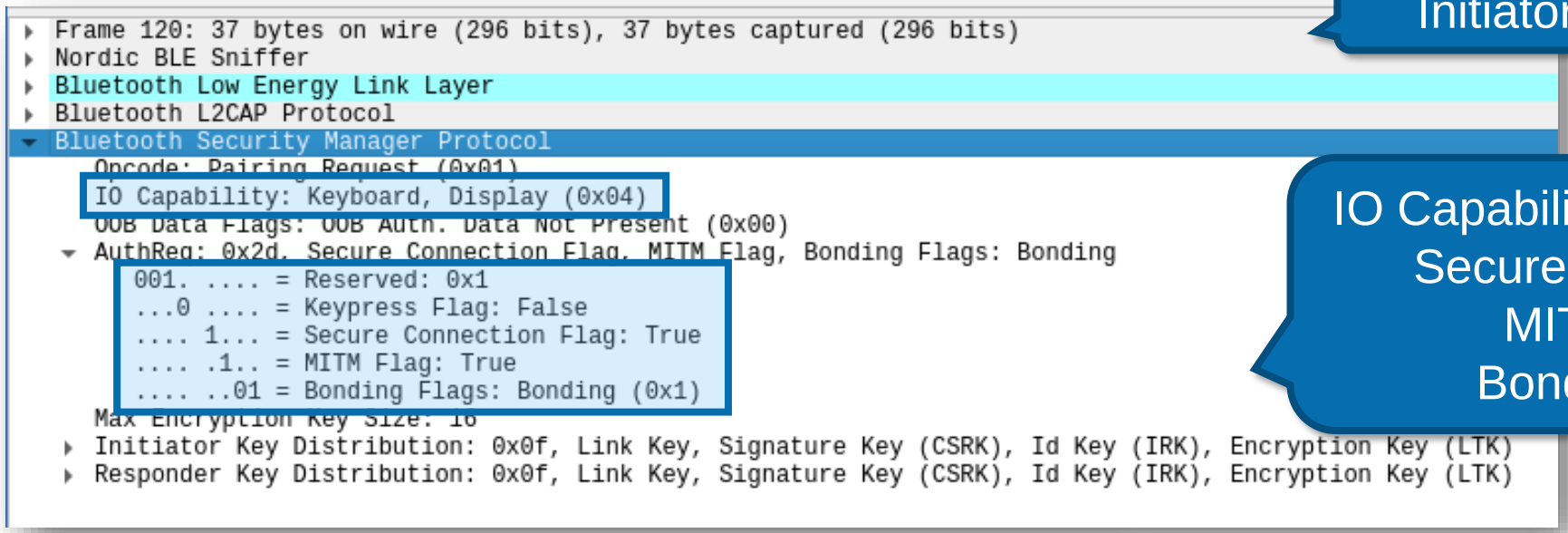


Unauthenticated 0xc0ffee!

Pairing Phase 1: Example #2

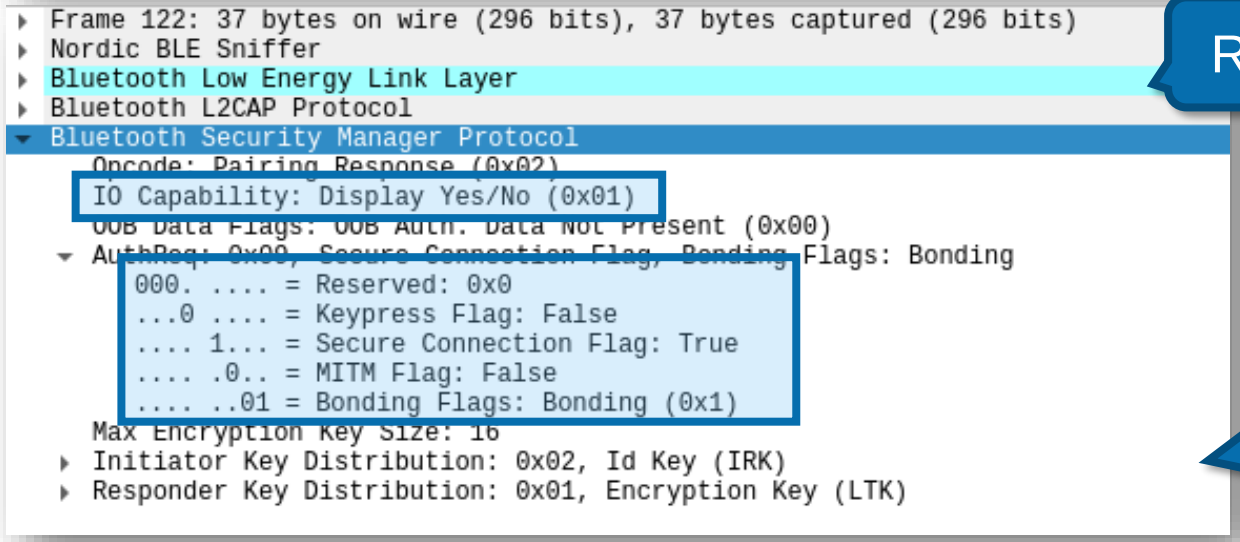


Pairing Phase 1: Example #2



Initiator

IO Capability: Keyboard, Display
Secure Connection: True
MITM Flag: True
Bonding Flag: True



Responder

IO Capability: Display Yes/No
Secure Connection: True
MITM Flag: False
Bonding Flag: True

Pairing Phase 1: Example #2

		Initiator			
		OOB Set	**OOB Not Set**	**MITM Set**	**MITM Not Set**
Responder	**OOB Set**	Use OOB	Use OOB		
	OOB Not Set	Use OOB	Check MITM		
	MITM Set			Use IO Capabilities	Use IO Capabilities
	MITM Not Set			Use IO Capabilities	Use Just Works

Table 2.7: Rules for using Out-of-Band and MITM flags for LE Secure Connections pairing

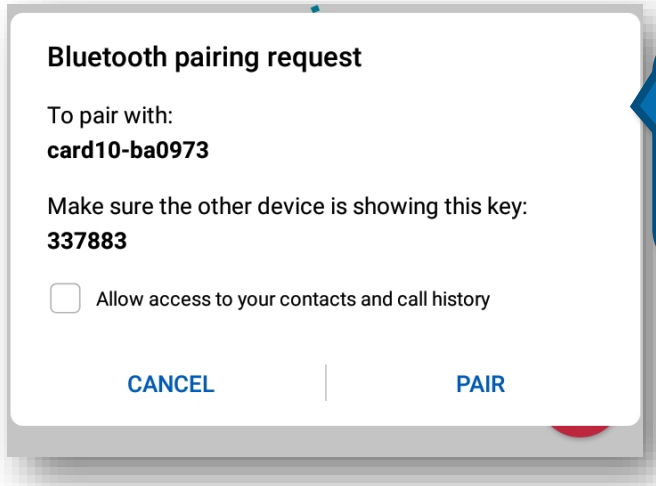
	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Display Only	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, ini-tiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, ini-tiator inputs Authenti-cated
Display YesNo	Just Works Unauthenti-cated	Just Works (For LE Legacy Pairing) Unauthenti-cated	Passkey Entry: responder displays, ini-	Just Works Unauthenti-	Passkey Entry (For LE Legacy Pairing): responder displays, ini-tiator inputs Authenti-cated
		Nu Co (Fo Se ne Authenti-cated			Numeric Comparison (For LE Secure Con-nections) Authenti-cated

Table 2.8: Mapping of IO capabilities to key generation method

LE Secure Connections

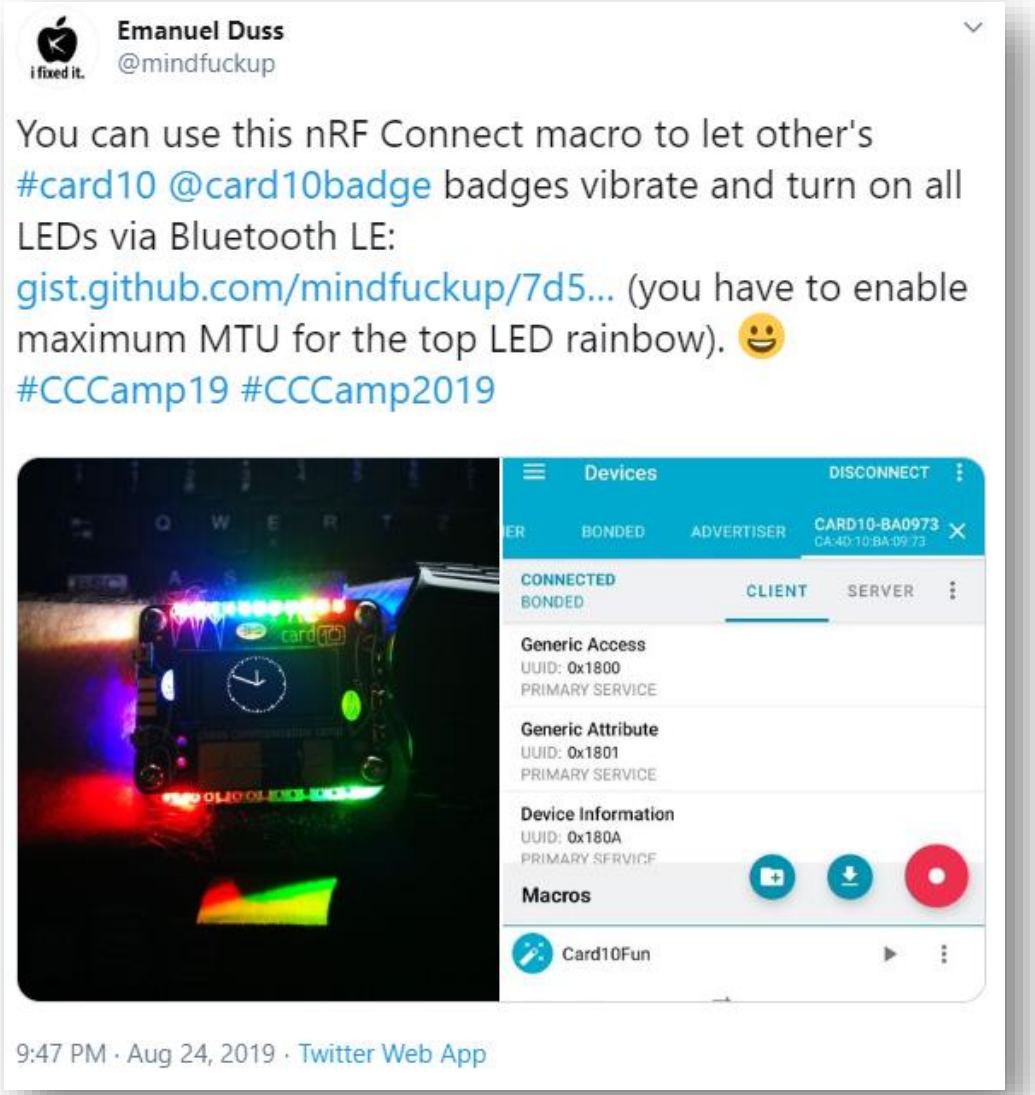
Selected: Numeric Comparison

Pairing Phase 1: Example #2



Confirm on both devices that the same key is shown!

Implementation may be incomplete



Pairing Phase 2: Key Generation

LE Legacy Pairing

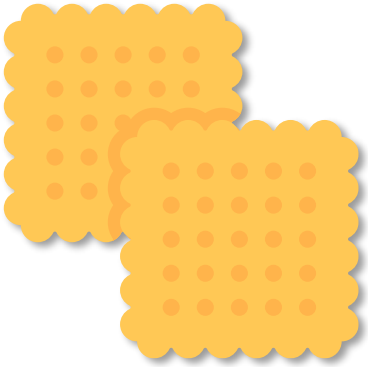
- Temporary Key (TK) → Short Term Key (STK) → Long Term Key (LTK) → Session Key (SK)
- TK is generated from the selected key exchange method
  - Just Works: TK = 0
  - Passcode Entry: TK = Entered PIN (000000-999999)
  - Out of Band: TK = OOB Exchanged Key (128 bits)

The arrows «→» mean «some cryptographic algorithm defined in the spec».

LE Secure Connections

- No Temporary Keys (TK) or Short Term Key (STK)
- Long Term Key (LTK) generated using Elliptic Curve Diffie-Hellman (ECDH)
- Long Term Key (LTK) → Session Key (SK)

Key Cracking



LE Legacy Pairing is easy to crack:

- Just Works
  - TK is always 0 → Always the same static key

- Passkey Entry: 6 digits = 1’000’000 possibilities
  - Provides 20 bits of security (log2(1000000) ≈ 20) → Can be cracked immediately

- Out of Band
  - Depends on the generated key → this can be strong!

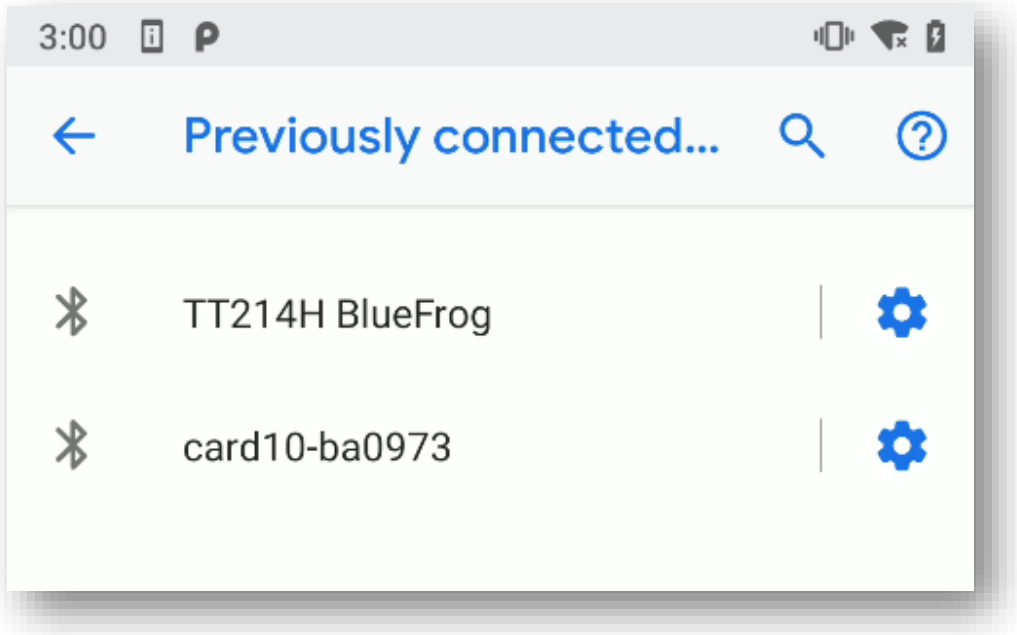
LE Secure Connections cannot be cracked:

- Elliptic Curve Diffie-Hellman (ECDH) key exchange is used.

Phase 3: Bonding / Transport Specific Key Distribution

- Bonding is the exchange of a Long Term Key (LTK) after pairing
- No pairing is required for the next session
- Exchanged in Pairing Phase 3
- Creates relationship and permanent security between two devices
- Link key as an identifier
- Link key stored on both devices
- Link key used for further authentication
- Long Term Key (LTK) stored on both devices

The bonded devices can be seen in Android in the Bluetooth menu



BLE Sniffing

BLE Sniffing

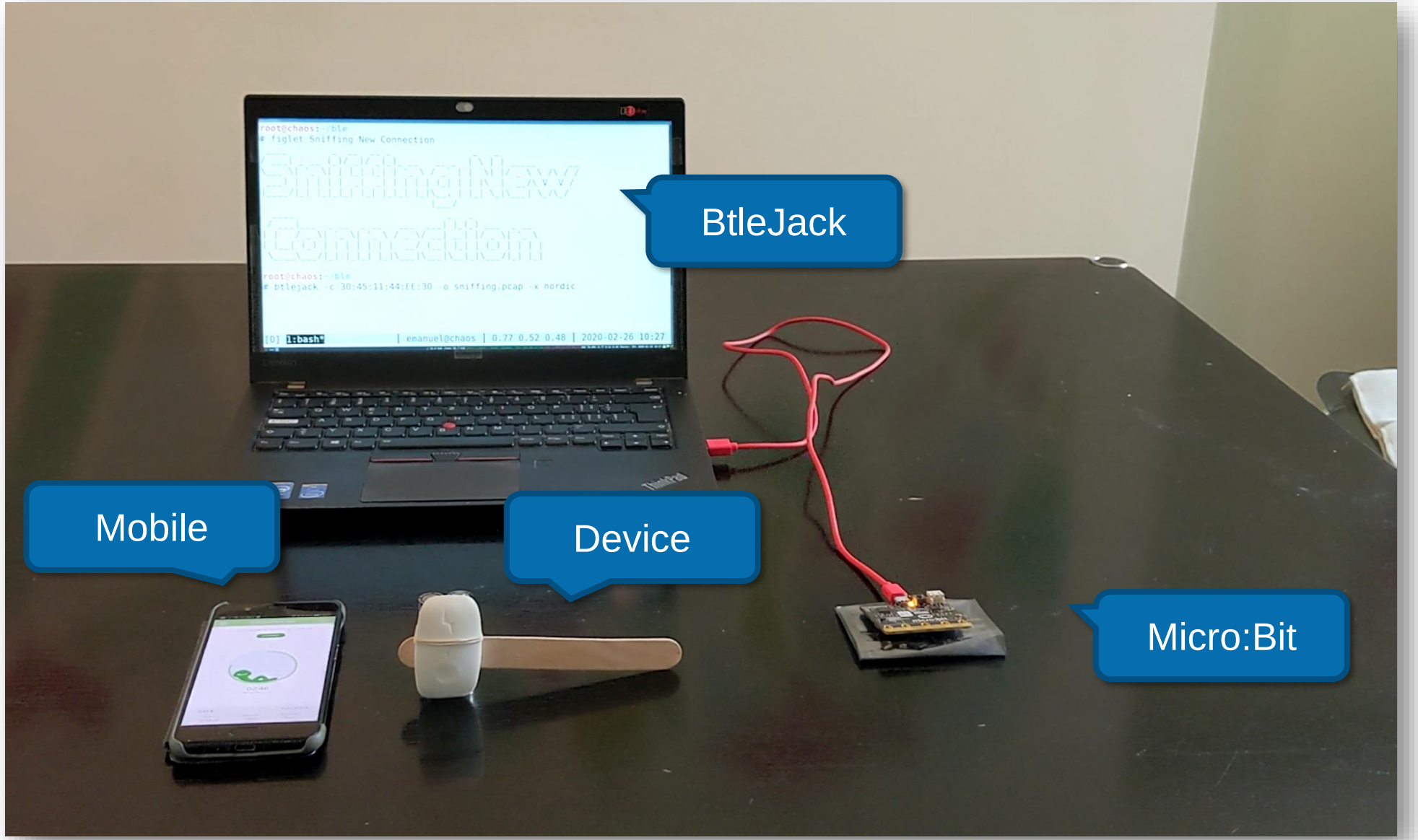
- Blackbox Approach: Capture the packets in the air
  - Ubertooth
  - Adafruit Bluefruit LE Sniffer
  - Micro:Bit / BtleJack
- Whitebox Approach: Sniff directly on the used BLE interface
  - Android HCI Snoop Log
  - Linux HCI Snoop Log

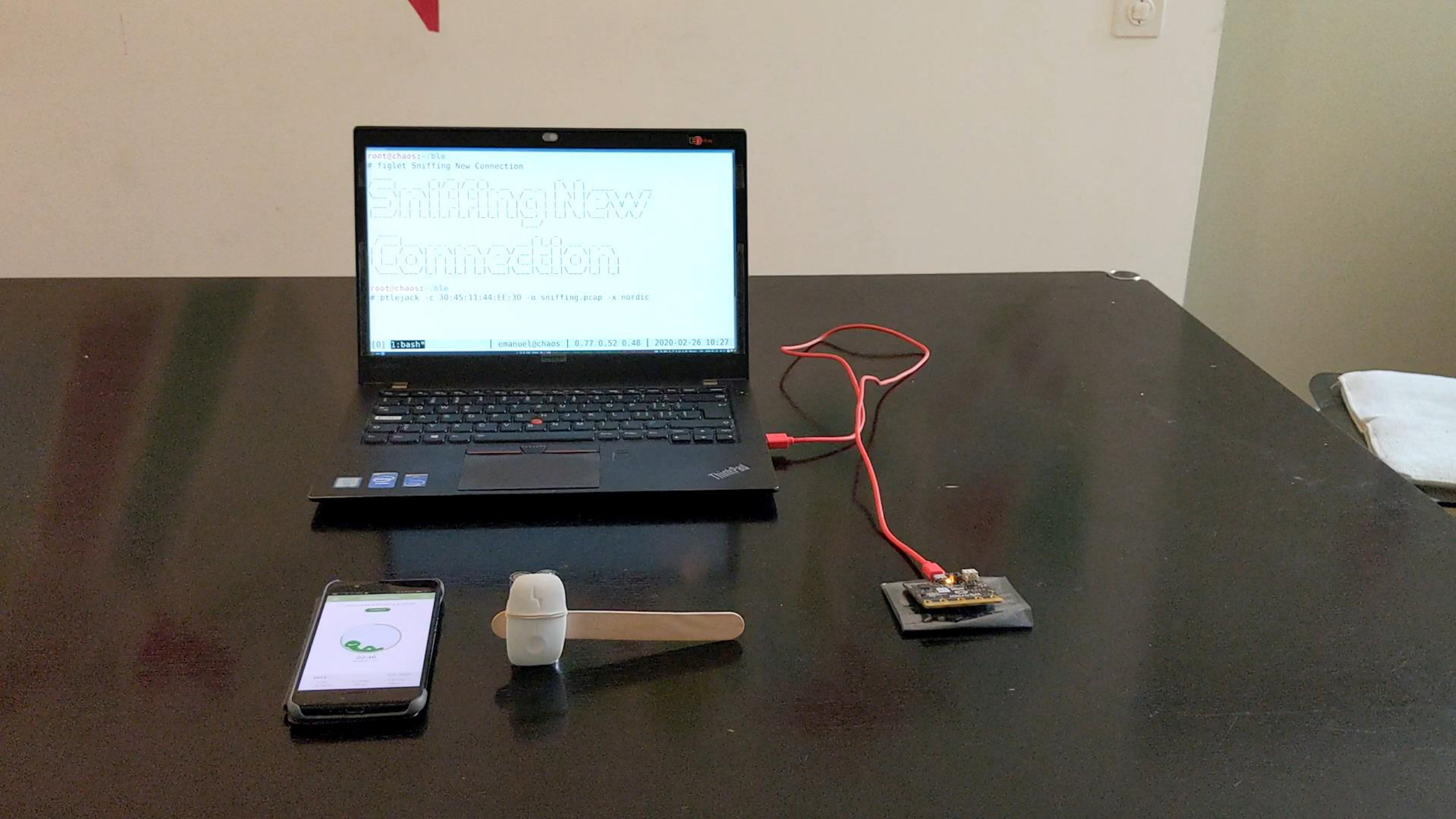
BtleJack

- Bluetooth LE Swiss-Army Knife Software by Damien Cauquil
- Firmware for various devices: BBC Micro:Bit, Adafruit Bluefruit LE sniffer, nRF51822 Eval Kit
- Micro:Bit is an OpenSource ARM Hardware created for teaching programming
- It has an Nordic nRF51822 chip → Bluetooth Low Energy
- Features: Sniffing, Jamming, Hijacking
- Supports multiple devices to sniff all 3 advertisement channels
- Respects the channel map
- More reliable than Adafruit Bluefruit LE Sniffer
- Project Page: https://github.com/virtualabs/btlejack

The way to go!

Demo Time: Sniffing New Connection




root@chaos:~/ble
figlet Sniffing New Connection
root@chaos:~/ble
btlejack -c 30:45:11:44:EE:30 -o sniffing.pcap -x nordic
[0] 1:bash*
emanuel@chaos | 0.77 0.52 0.48 | 2020-02-26 10:27

PCAP Analysis

No.	Time	Source	Destination	Protocol	Length	Value	Info
394	44.824232	Slave_0x50657412	Master_0x50657412	ATT	35	8900	Rcvd Handle Value Notification, Handle: 0x004b (Unknown: Unknown)
395	45.330332	Slave_0x50657412	Master_0x50657412	ATT	35	5d00	Rcvd Handle Value Notification, Handle: 0x004b (Unknown: Unknown)
396	45.432084	Slave_0x50657412	Master_0x50657412	ATT	35	4b00	Rcvd Handle Value Notification, Handle: 0x004b (Unknown: Unknown)
397	45.938163	Slave_0x50657412	Master_0x50657412	ATT	35	7600	Rcvd Handle Value Notification, Handle: 0x004b (Unknown: Unknown)
398	46.039331	Slave_0x50657412	Master_0x50657412	ATT	35	9100	Rcvd Handle Value Notification, Handle: 0x004b (Unknown: Unknown)
399	46.646395	Slave_0x50657412	Master_0x50657412	ATT	35	a300	Rcvd Handle Value Notification, Handle: 0x004b (Unknown: Unknown)

```
+ Frame 396: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
+ Nordic BLE Sniffer
+ Bluetooth Low Energy Link Layer
+ Bluetooth L2CAP Protocol
- Bluetooth Attribute Protocol
  - Opcode: Handle Value Notification (0x1b)
      0... .... = Authentication Signature: False
      .0.. .... = Command: False
      ..01 1011 = Method: Handle Value Notification (0x1b)
  - Handle: 0x004b (Unknown: Unknown)
      [Service UUID: Unknown (0xaac0)]
      [UUID: Unknown (0xaaca)]
    Value: 4b00
```

Notification for device angle

PCAP Analysis

No.	Time	Source	Destination	Protocol	Length	Info
420	53.723458	Slave_0x50657412	Master_0x50657412	ATT	38	Rcvd Read Response, Handle: 0x0016 (Device Information: Firmware Revision String)
421	56.309847	Master_0x50657412	Slave_0x50657412	ATT	36	Sent Write Request, Handle: 0x0028 (Unknown: Unknown)
422	56.344767	Slave_0x50657412	Master_0x50657412	ATT	31	Rcvd Write Response, Handle: 0x0028 (Unknown: Unknown)
423	56.411022	Master_0x50657412	Slave_0x50657412	ATT	33	Sent Read Request, Handle: 0x0016 (Device Information: Firmware Revision String)

```
+ Frame 421: 36 bytes on wire (288 bits), 36 bytes captured (288 bits)
+ Nordic BLE Sniffer
+ Bluetooth Low Energy Link Layer
+ Bluetooth L2CAP Protocol
- Bluetooth Attribute Protocol
  - Opcode: Write Request (0x12)
      0... .... = Authentication Signature: False
      .0.. .... = Command: False
      ..01 0010 = Method: Write Request (0x12)
  - Handle: 0x0028 (Unknown: Unknown)
      [Service UUID: Unknown (0xaaa0)]
      [UUID: Unknown (0xaaa5)]
    Value: 030001
```

Write Request

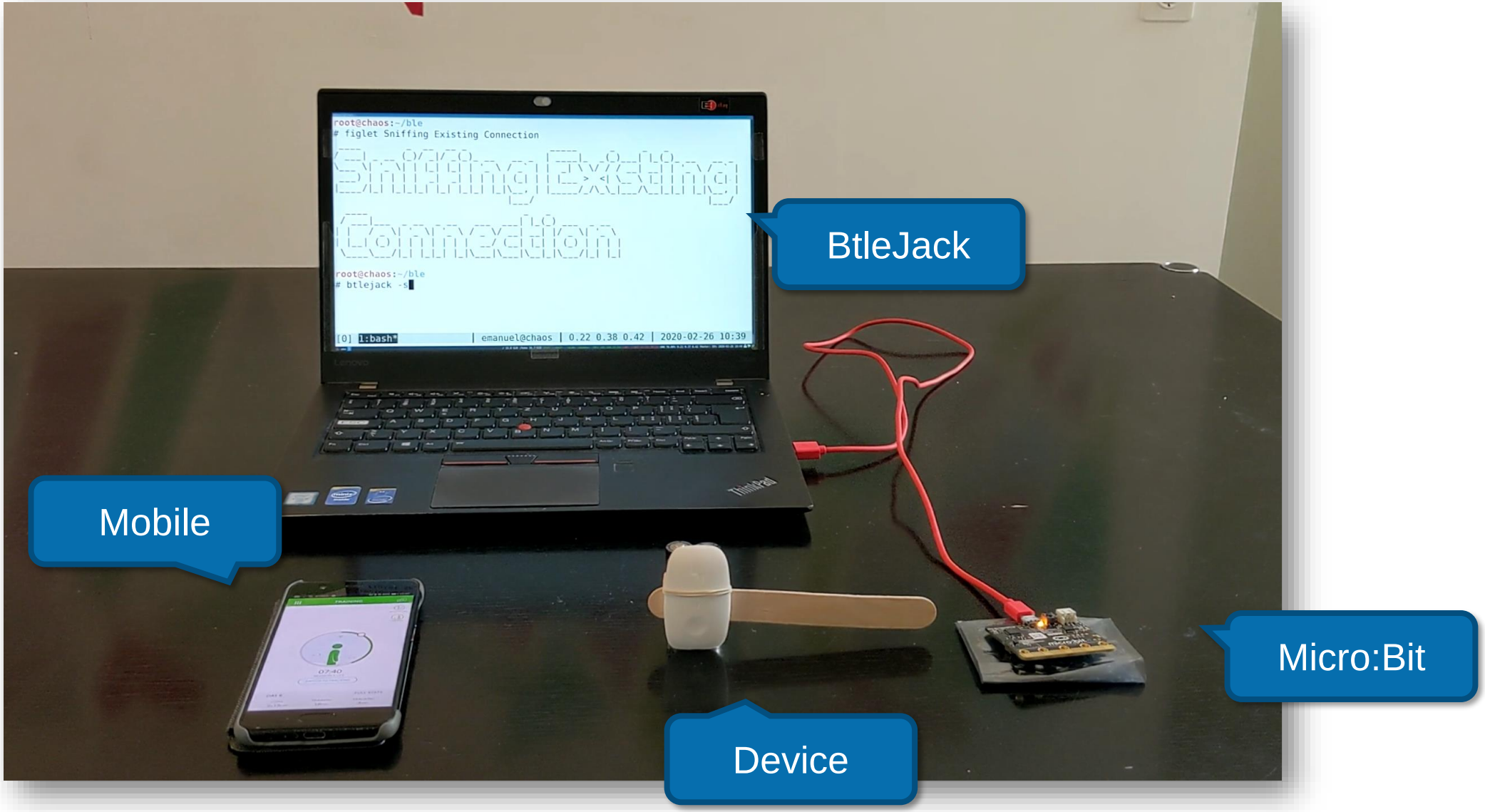
Change Vibration Mode:

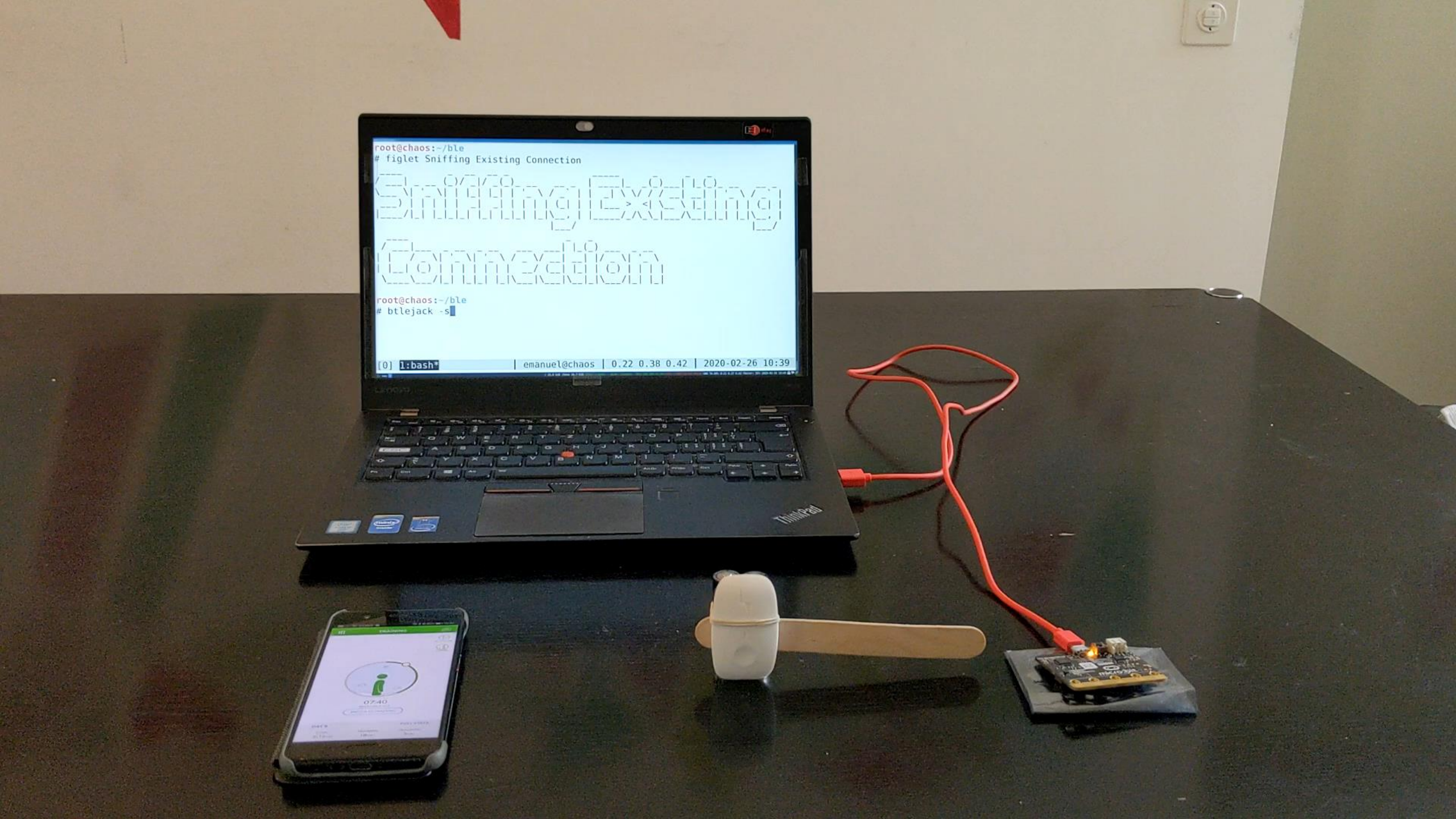
- Handle: 0x0028
- Value: 0x030001
- UUID: 0xaaa5

Encrypted Connections

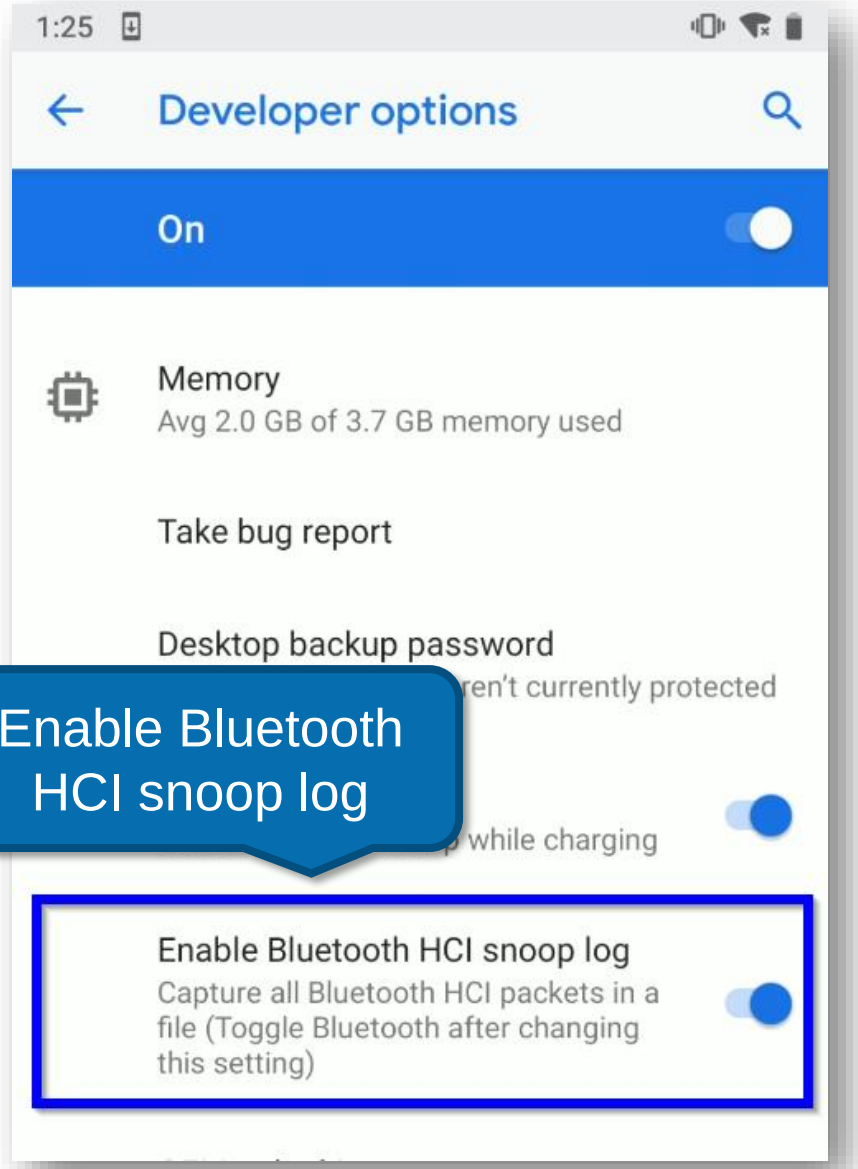
- Crackle brute forces the TK used during BLE Legacy Pairing
- 6 digit PIN (Pinentry Pairing) or 000000 (JustWorks Pairing) is used as a TK (added to 128 Bits)
- Easy to brute force
- Pairing handshake must be captured
- BtleJack's ll_phdr format is supported
- Project Page: https://github.com/mikeryan/crackle

Demo Time: Sniffing Existing Connection




root@chaos:~/ble
figlet Sniffing Existing Connection
root@chaos:~/ble
btlejack -s
[0] 1:bash*
emanuel@chaos | 0.22 0.38 0.42 | 2020-02-26 10:39
ThinkPad

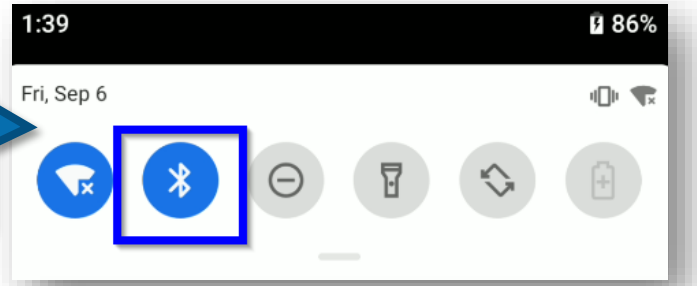
Android Bluetooth HCI Snoop Log



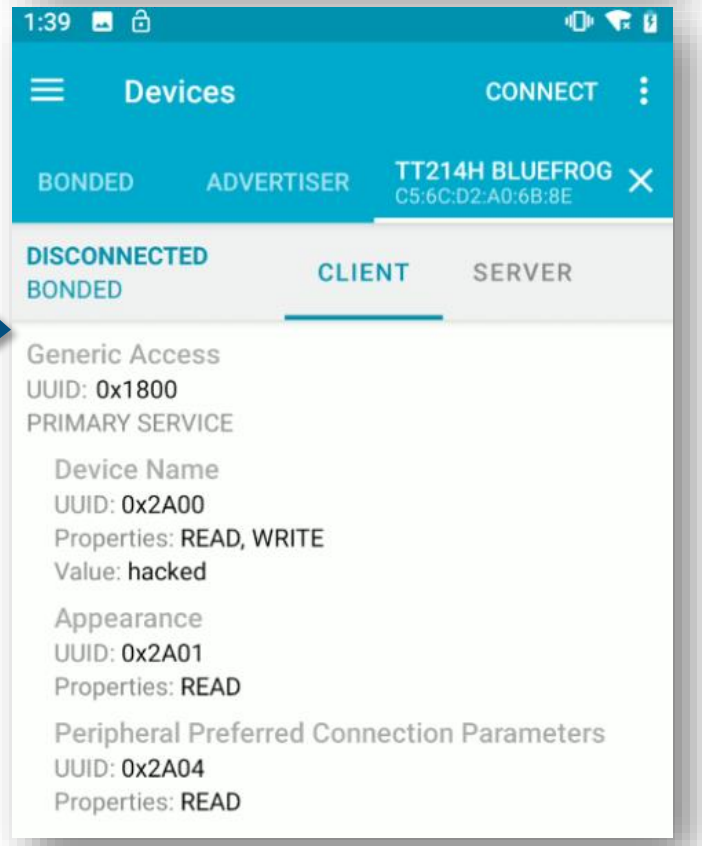
Enable Bluetooth HCI snoop log

HCI = Host Controller Interface

Toggle the Bluetooth adapter



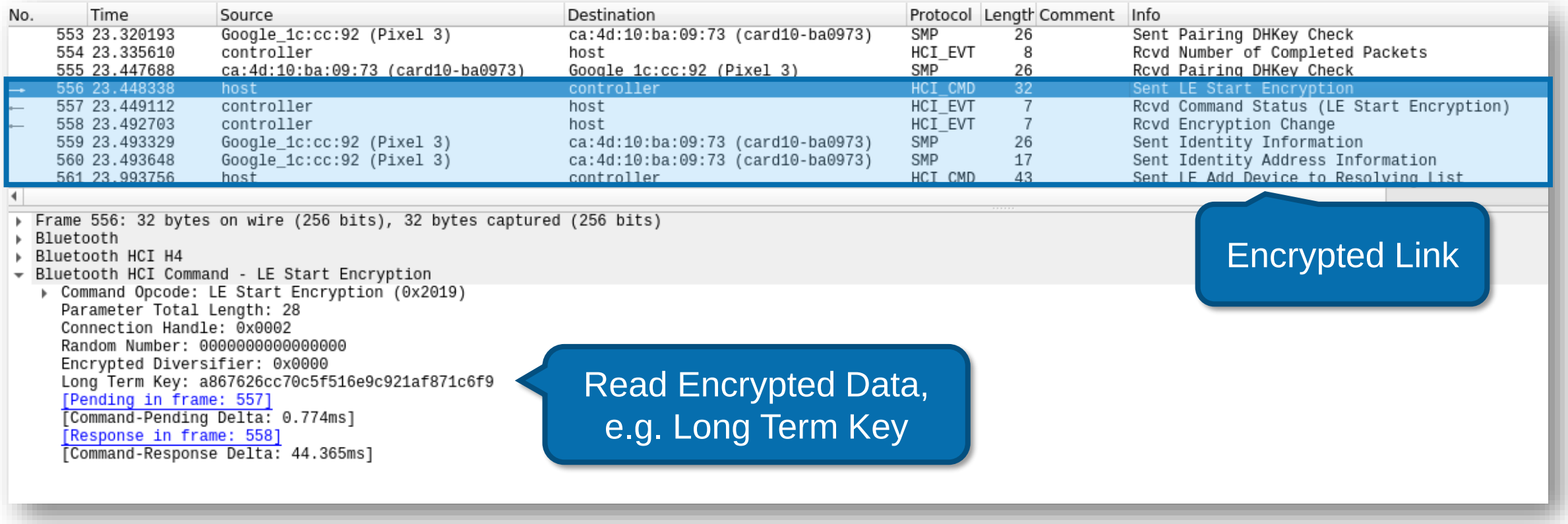
Start device interaction



Android Bluetooth Snoop Log

```
# adb shell su -c cat /data/misc/bluetooth/logs/btsnoop_hci.log > btsnoop_hci.log
# file btsnoop_hci.log
btsnoop_hci.log: BTSnoop version 1, HCI UART (H4)
```

Location for Pixel 3, Android 9



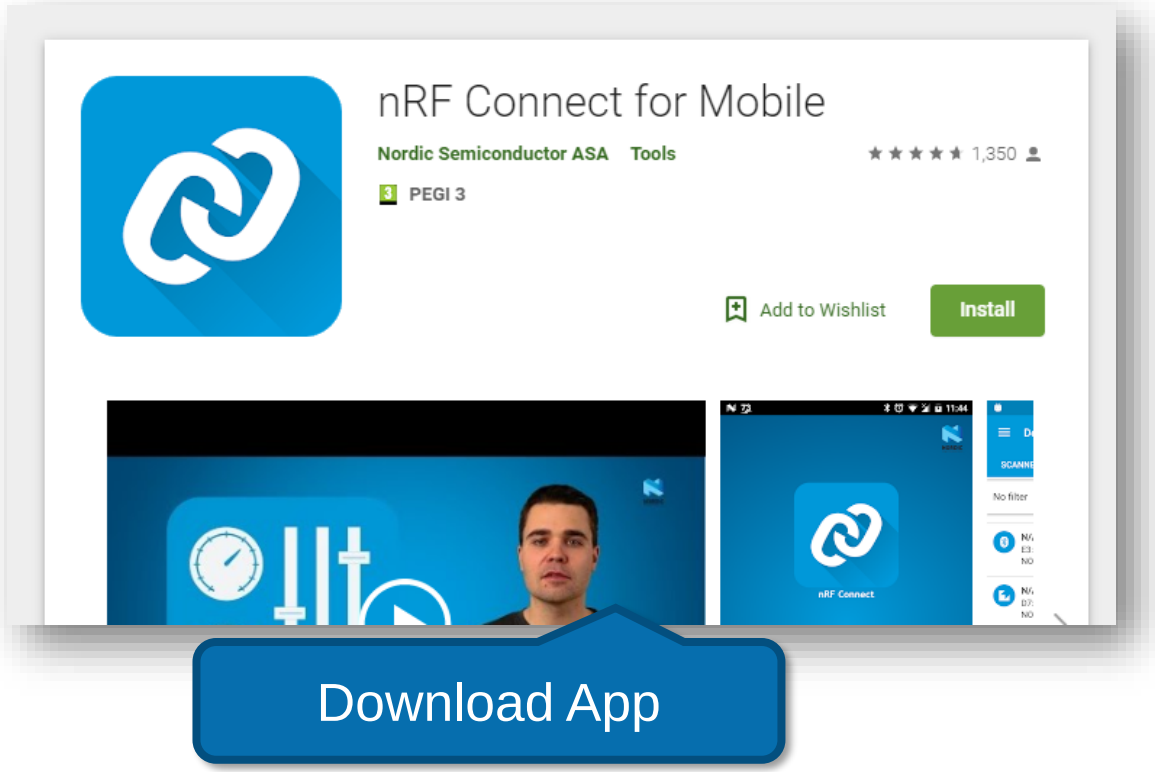
Encrypted Link

Read Encrypted Data, e.g. Long Term Key

BLE Interaction

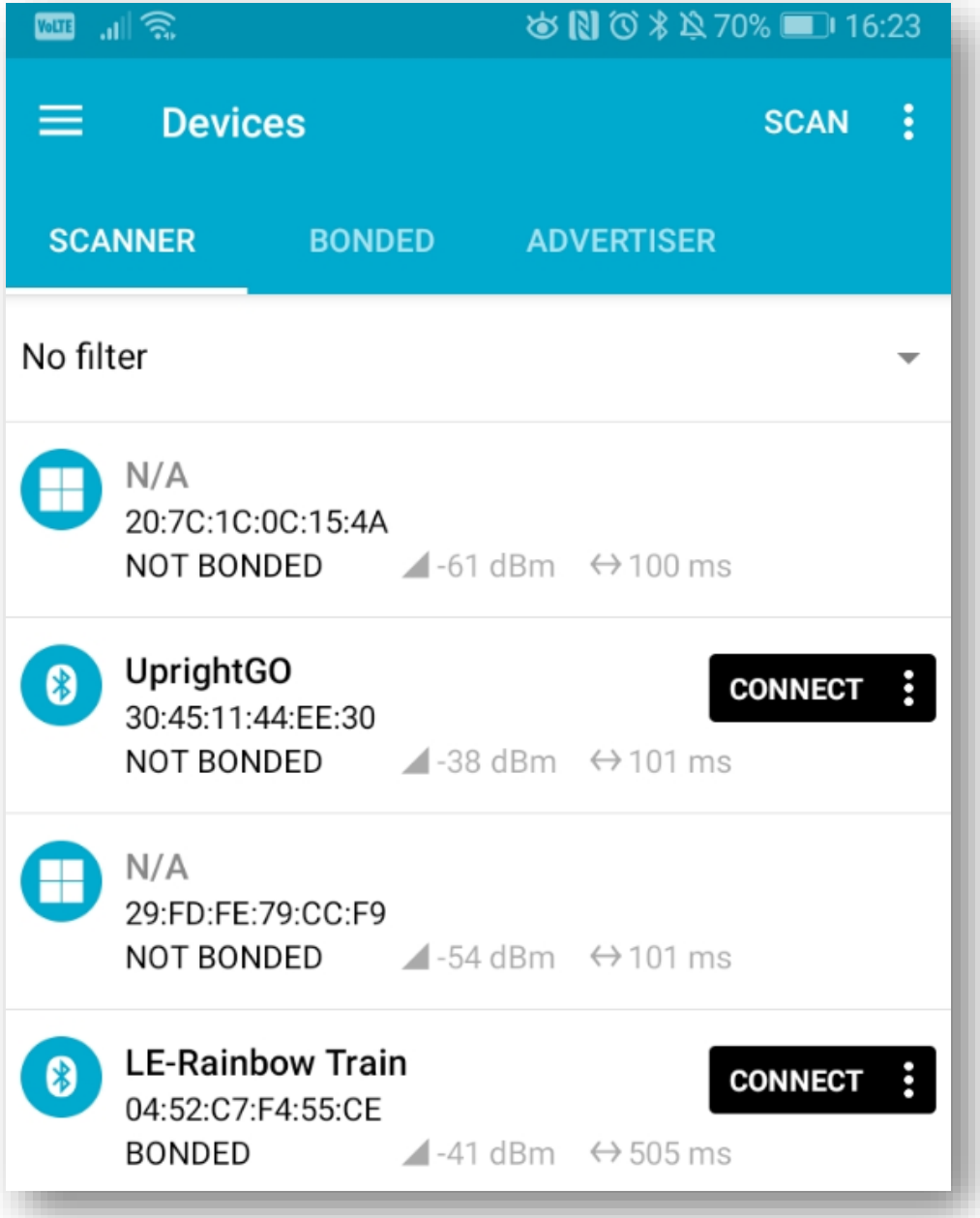
Android App

- nRF Connect for Mobile

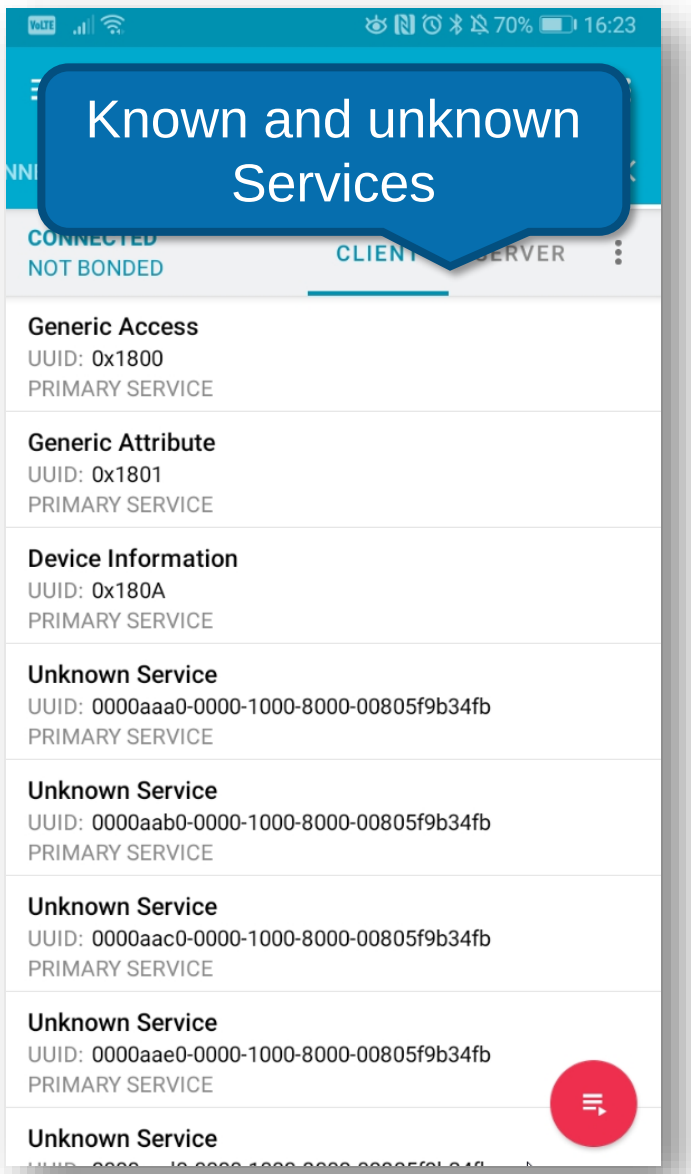


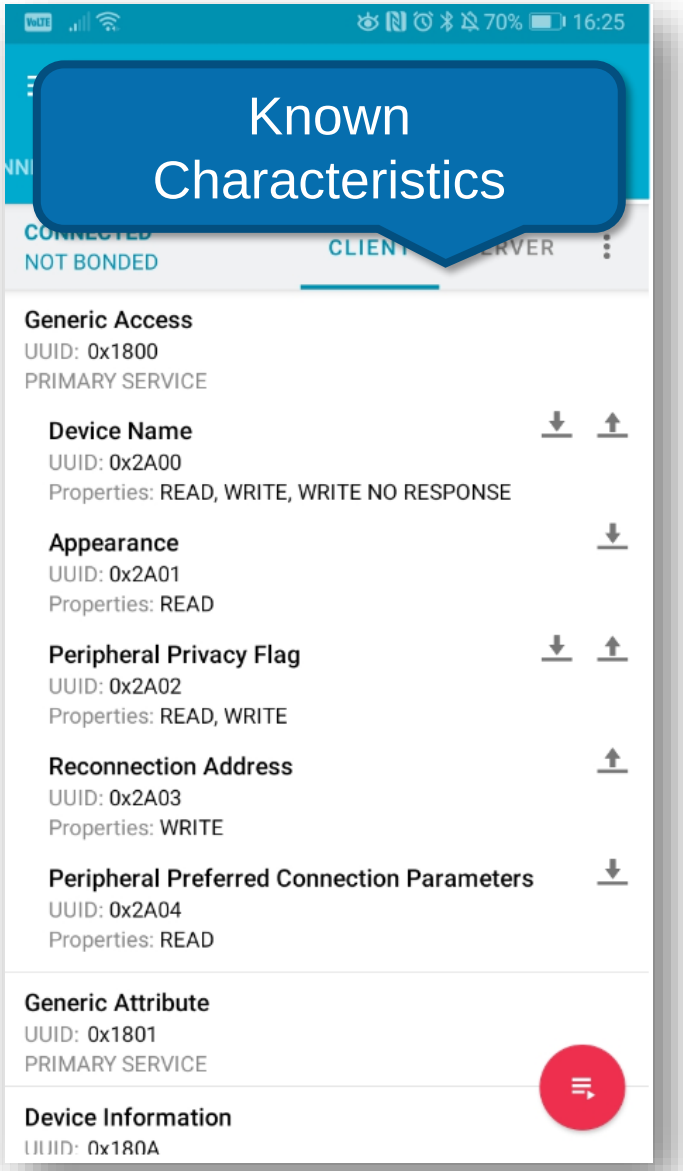
Download App

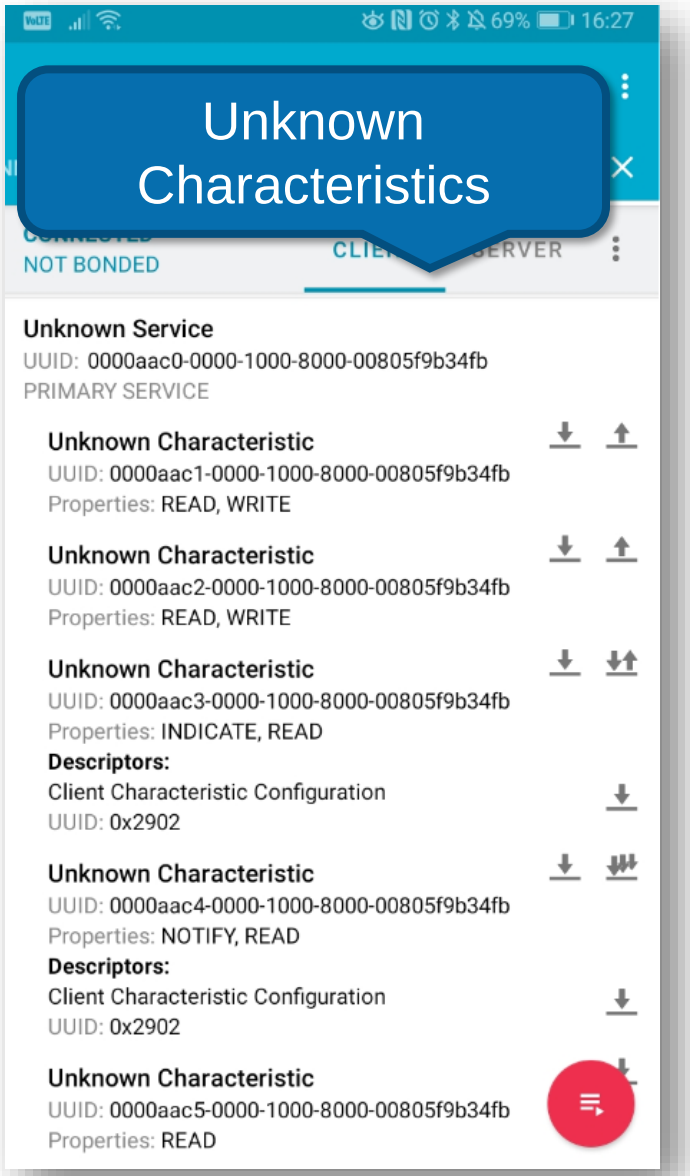
Scan for BLE devices



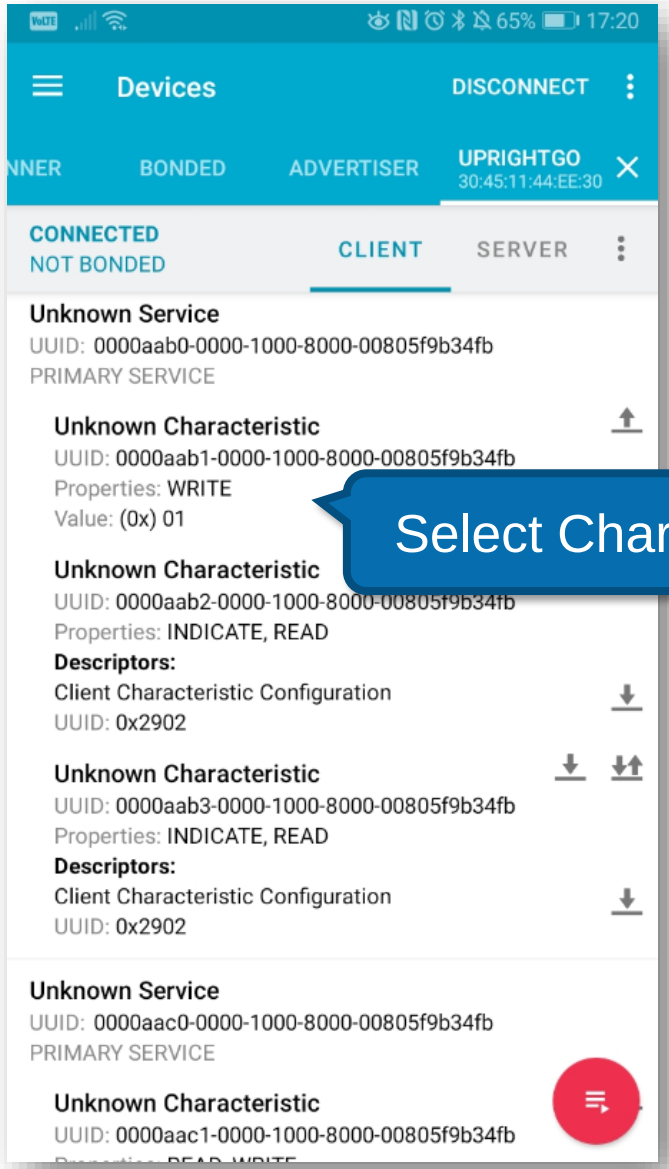
Android App


Known and unknown Services
NOT BONDED
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Device Information
UUID: 0x180A
PRIMARY SERVICE
Unknown Service
UUID: 0000aaa0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Service
UUID: 0000aab0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Service
UUID: 0000aac0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Service
UUID: 0000aae0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Service

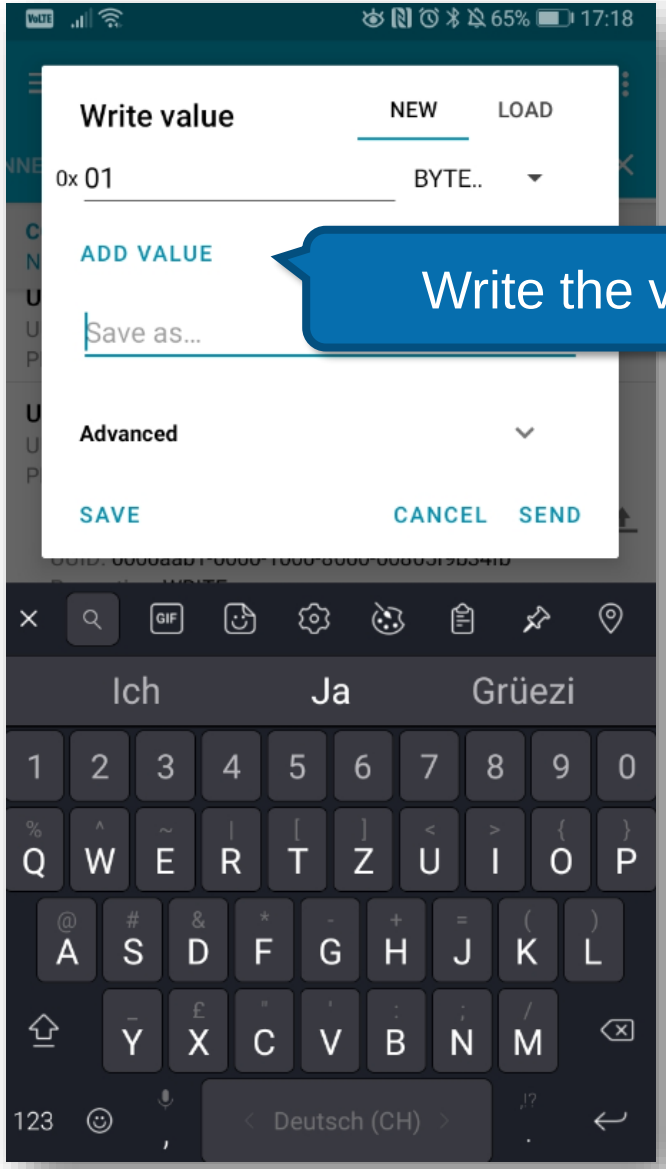

Known Characteristics
NOT BONDED
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Device Name
UUID: 0x2A00
Properties: READ, WRITE, WRITE NO RESPONSE
Appearance
UUID: 0x2A01
Properties: READ
Peripheral Privacy Flag
UUID: 0x2A02
Properties: READ, WRITE
Reconnection Address
UUID: 0x2A03
Properties: WRITE
Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Device Information


Unknown Characteristics
NOT BONDED
Unknown Service
UUID: 0000aac0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Characteristic
UUID: 0000aac1-0000-1000-8000-00805f9b34fb
Properties: READ, WRITE
Unknown Characteristic
UUID: 0000aac2-0000-1000-8000-00805f9b34fb
Properties: READ, WRITE
Unknown Characteristic
UUID: 0000aac3-0000-1000-8000-00805f9b34fb
Properties: INDICATE, READ
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Unknown Characteristic
UUID: 0000aac4-0000-1000-8000-00805f9b34fb
Properties: NOTIFY, READ
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Unknown Characteristic
UUID: 0000aac5-0000-1000-8000-00805f9b34fb
Properties: READ

Android App

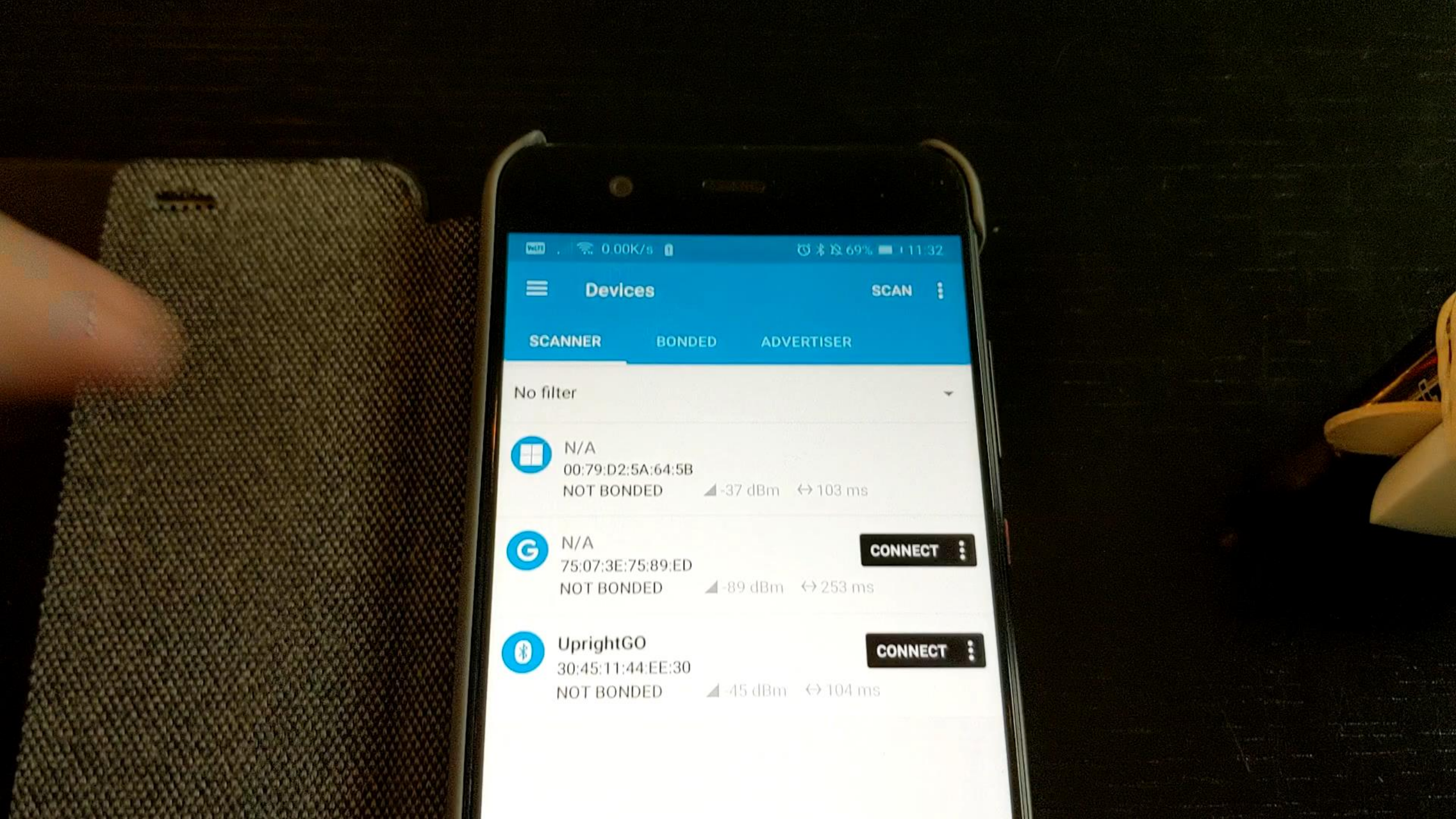


Select Characteristic



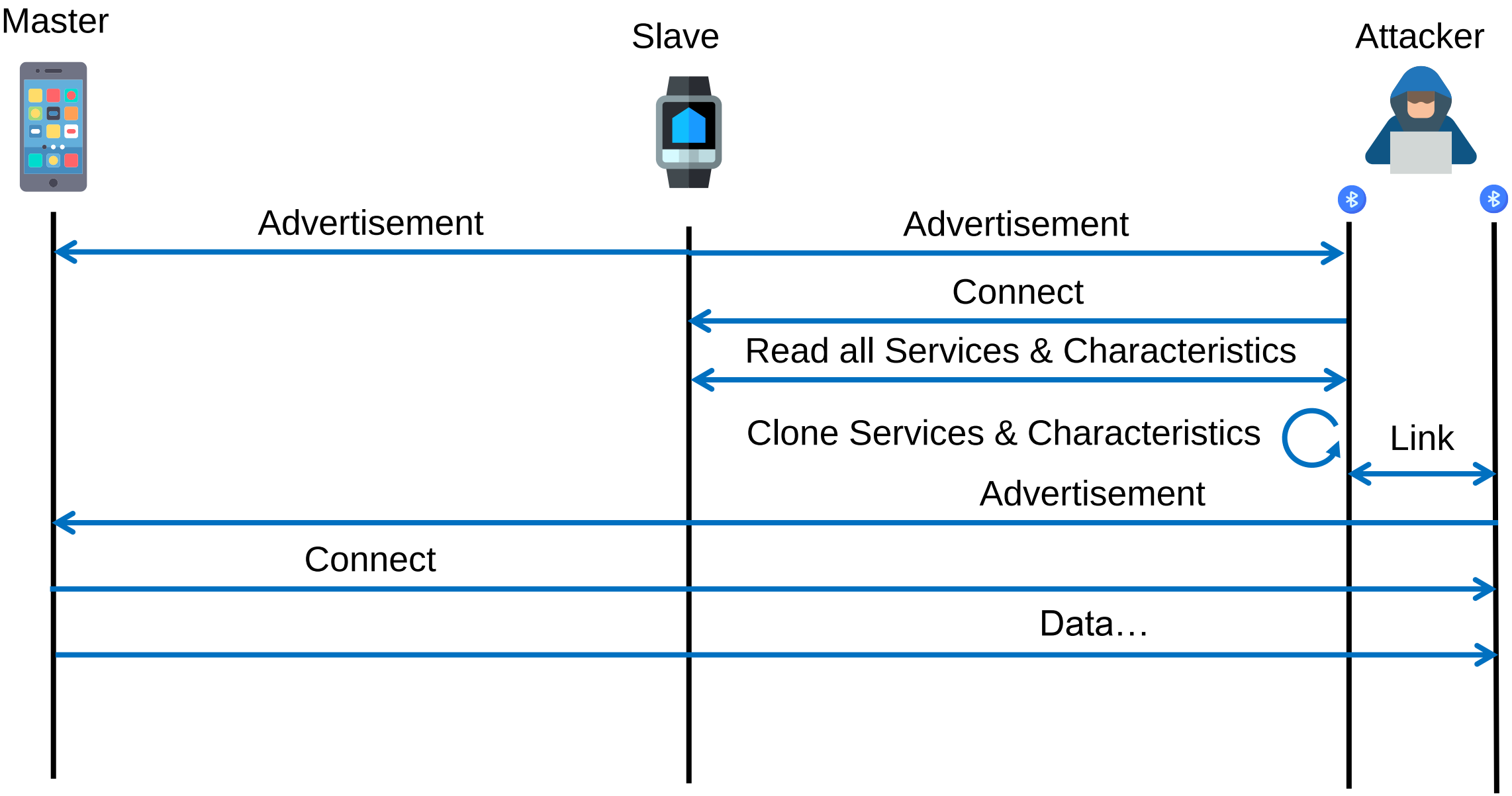
Write the value

Demo Time: Device Interaction

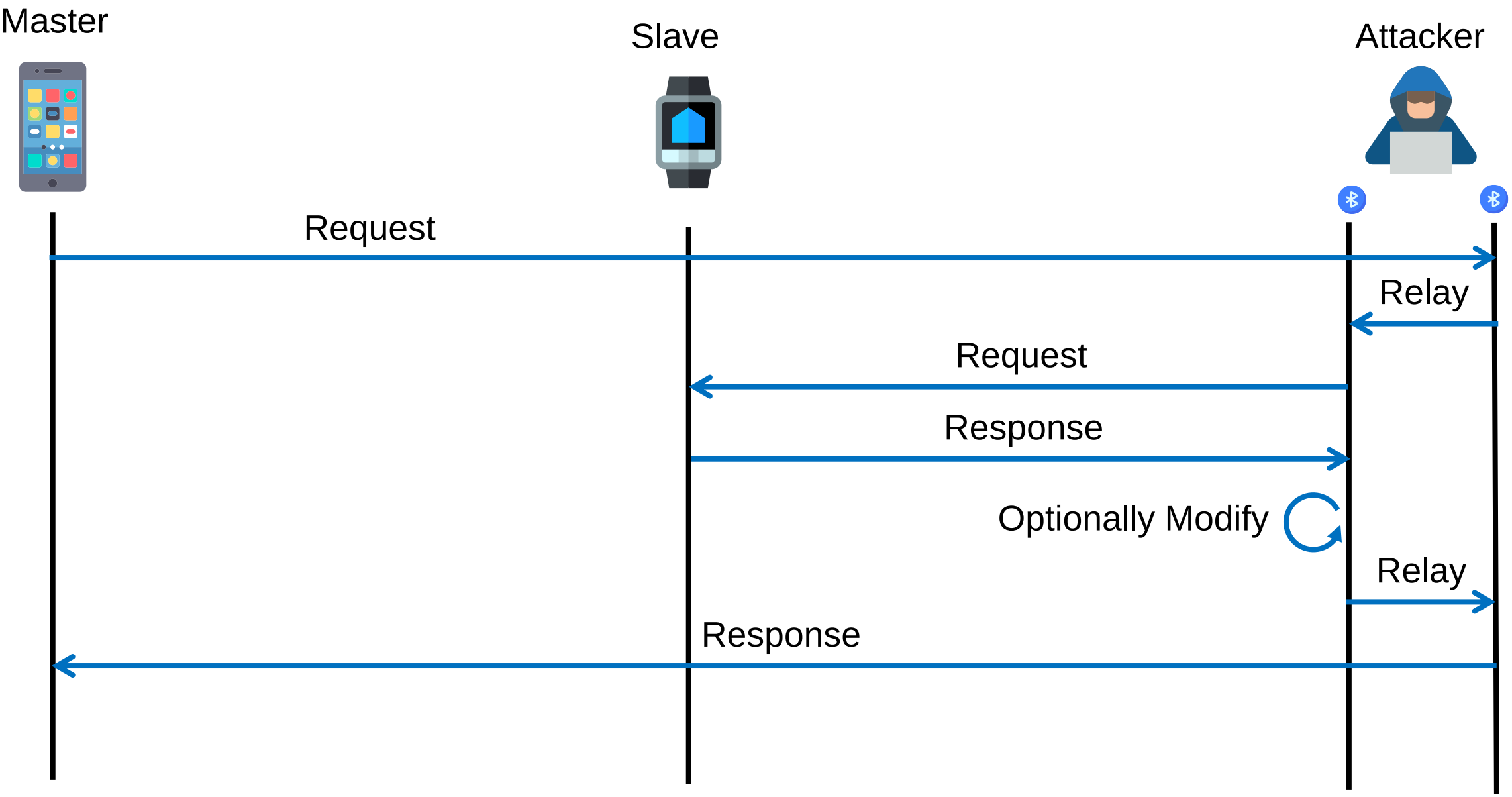

0.00K/s
69%
11:32
Devices
SCAN
SCANNER
BONDED
ADVERTISER
No filter
N/A
00:79:D2:5A:64:5B
NOT BONDED
-37 dBm
103 ms
N/A
CONNECT
75:07:3E:75:89:ED
NOT BONDED
-89 dBm
253 ms
UprightGO
CONNECT
30:45:11:44:EE:30
NOT BONDED
-45 dBm
104 ms

BLE Man-in-the-Middle

BLE Man-in-the-Middle



BLE Man-in-the-Middle



MITM Software

- GATTacker by Slawomir Jasek
  - Project Page: https://github.com/securing/gattacker
  - Console tools to perform the attacks
  - Writing hooks for manipulating the traffic
- BtleJuice by Econocom Digital Security
  - Project Page: https://github.com/DigitalSecurity/btlejuice
  - Webinterface to perform the attacks
- Both tools work in the same way:
  - 2 VMs: Master (central) and Slave (peripheral) with each one Bluetooth adapter
  - VM 1 (Master): Central connects to peripheral
  - VM 2: Websocket to VM 1 and clone/advertise the same GATT services
  - Sniff, intercept and modify, replay
- Downsides
  - Complex setup, they don't work properly, no pairing support

I invested a lot of work and time in getting the tools to work. It was not worth the time ☹

Feature Requirements for active MITM Protection

	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Display Only	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, initiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry: responder displays, initiator inputs Authenti-cated
Display YesNo	Just Works Unauthenti-cated	Just Works (For LE Legacy Pairing) Unauthenti-cated	Passkey Entry: responder displays, initiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry (For LE Legacy Numeric Comparison (For LE Secure Connections) Authenti-cated

Table 2.8: Mapping of IO capabilities to key generation method

This means, both devices need a keyboard AND/OR a display!

	Initiator				
Responder	**DisplayOnly**	**Display YesNo**	**Keyboard Only**	**NoInput NoOutput**	**Keyboard Display**
Keyboard Only	Passkey Entry: initiator displays, responder inputs Authenti-cated	Passkey Entry: initiator displays, responder inputs Authenti-cated	Passkey Entry: initiator and responder inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry: initiator displays, responder inputs Authenti-cated
NoInput NoOutput	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Just Works Unauthenti-cated	Just Works Unauthenti-cated
Keyboard Display	Entry: initiator displays, responder inputs Authenti-cated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenti-cated Numeric Comparison (For LE Secure Connections) Authenti-cated	Passkey Entry: responder displays, initiator inputs Authenti-cated	Just Works Unauthenti-cated	Passkey Entry (For LE Legacy Pairing): initiator displays, responder inputs Authenti-cated Numeric Comparison (For LE Secure Connections) Authenti-cated

Table 2.8: Mapping of IO capabilities to key generation method

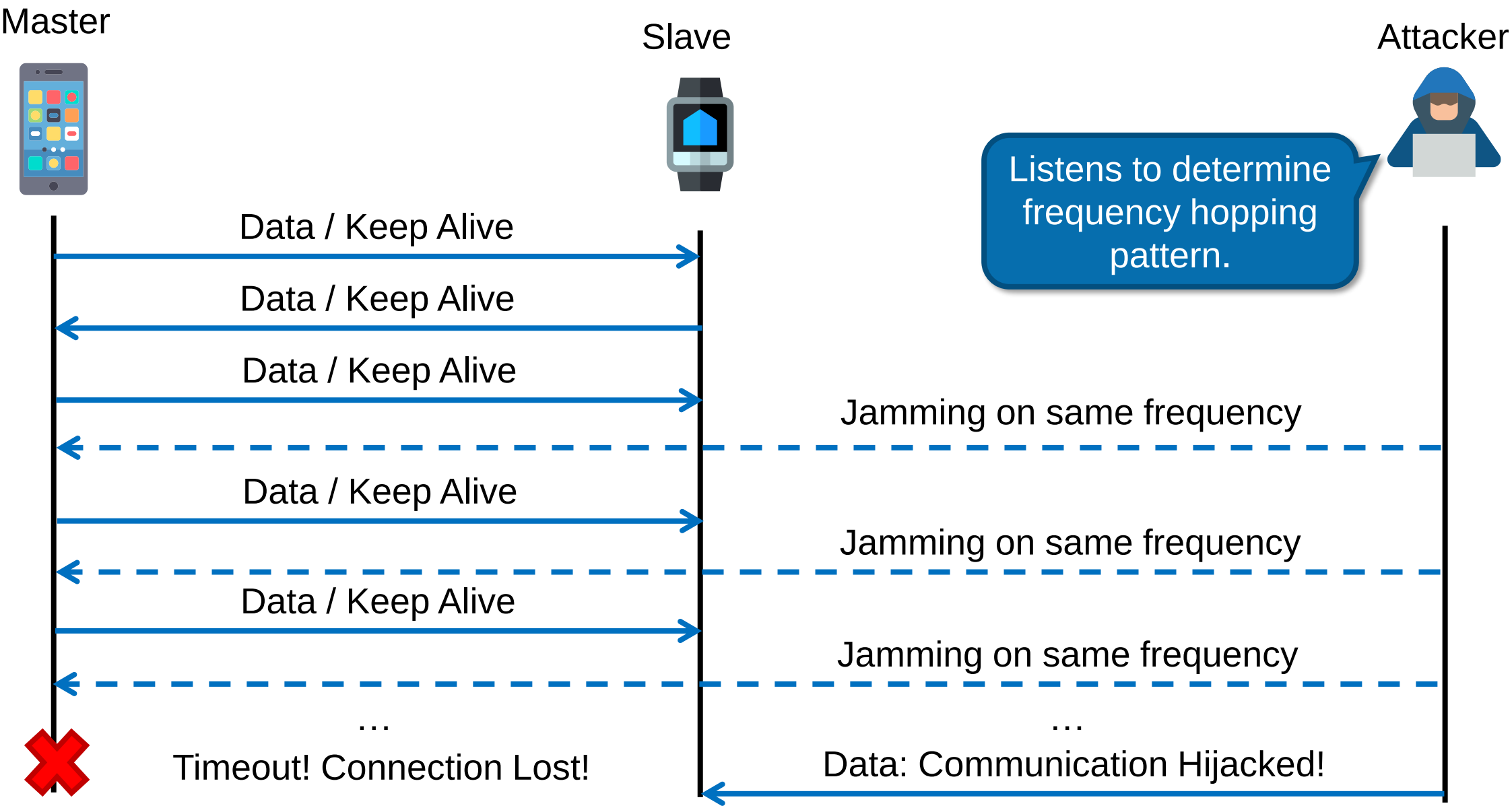
Which Pairing Methods are Secure?

- Use pairing methods which use strong key generation mechanisms and support authentication!

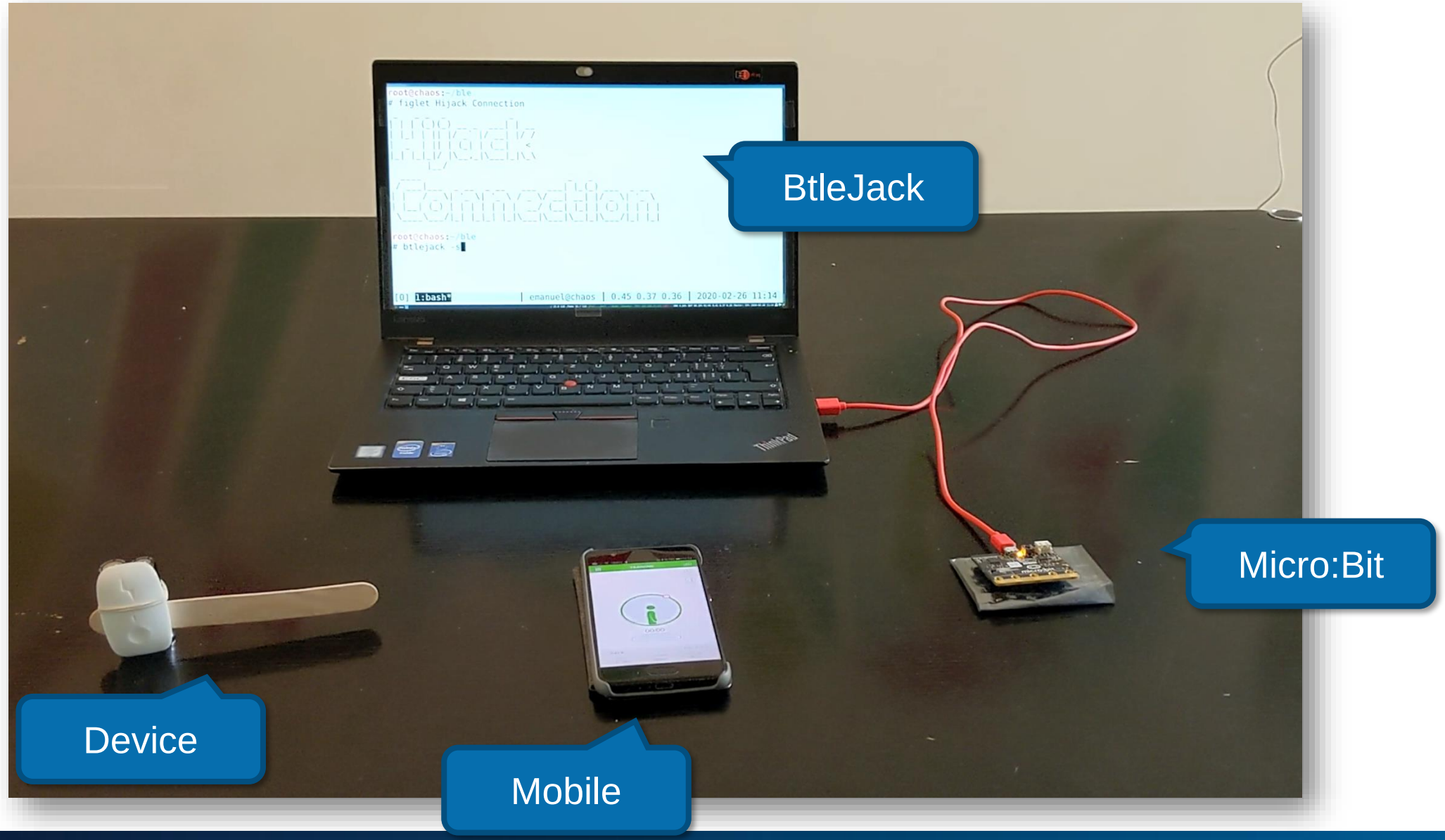
Security Type	Pairing Method	Passive Sniffing	Active MitM
No Pairing	-	FAIL	FAIL
LE Legacy Pairing	Just Works	FAIL	FAIL
LE Legacy Pairing	Passkey Entry	FAIL	PASS
LE Legacy Pairing	Out-of-Band	PASS	PASS
LE Secure Connection	Just Works	PASS	FAIL
LE Secure Connection	Passkey Entry	PASS	PASS
LE Secure Connection	Out-of-Band	PASS	PASS
LE Secure Connection	Numeric Comparison	PASS	PASS

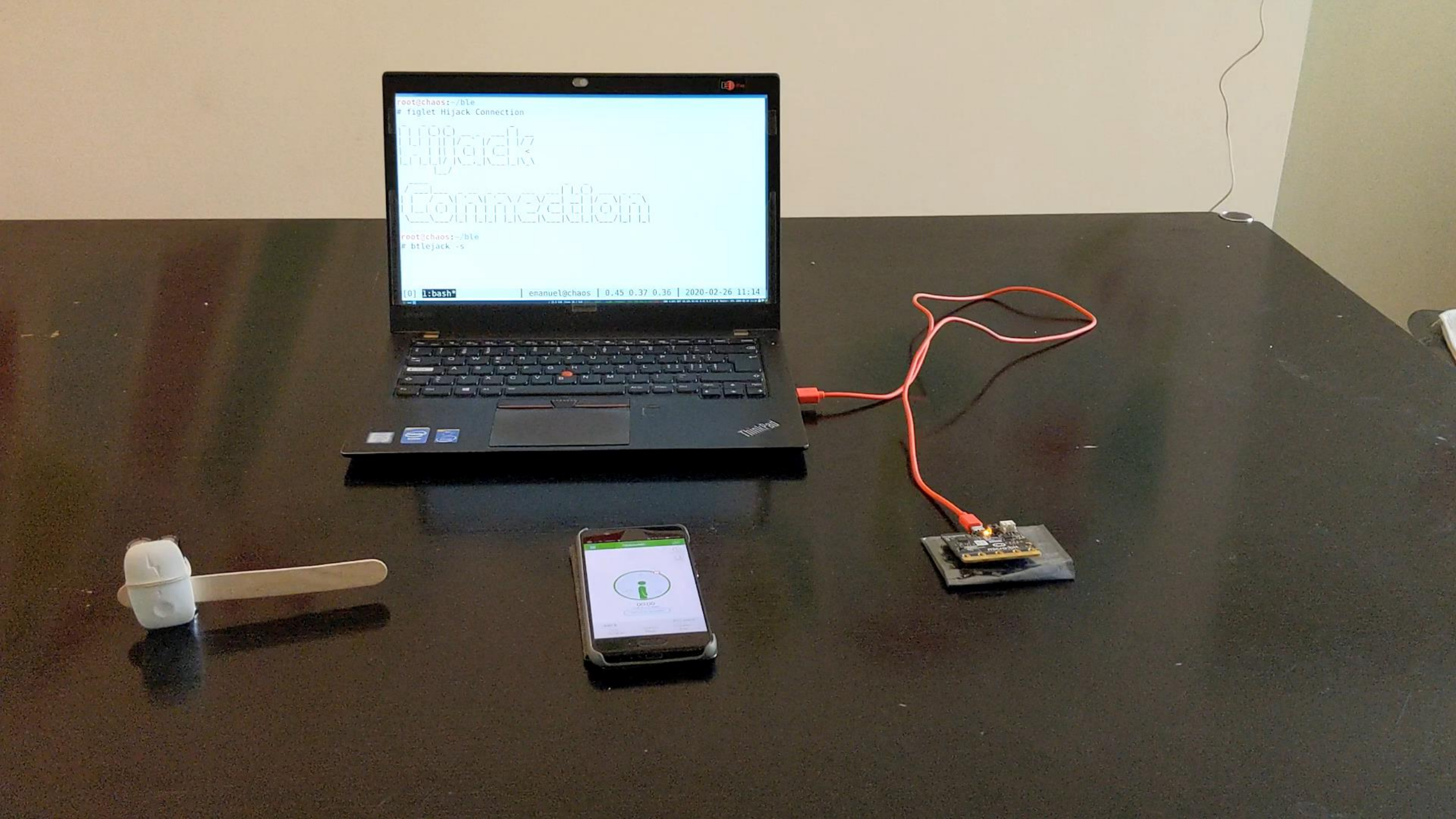
BLE Hijacking

BLE Hijacking



Demo Time: Hijacking

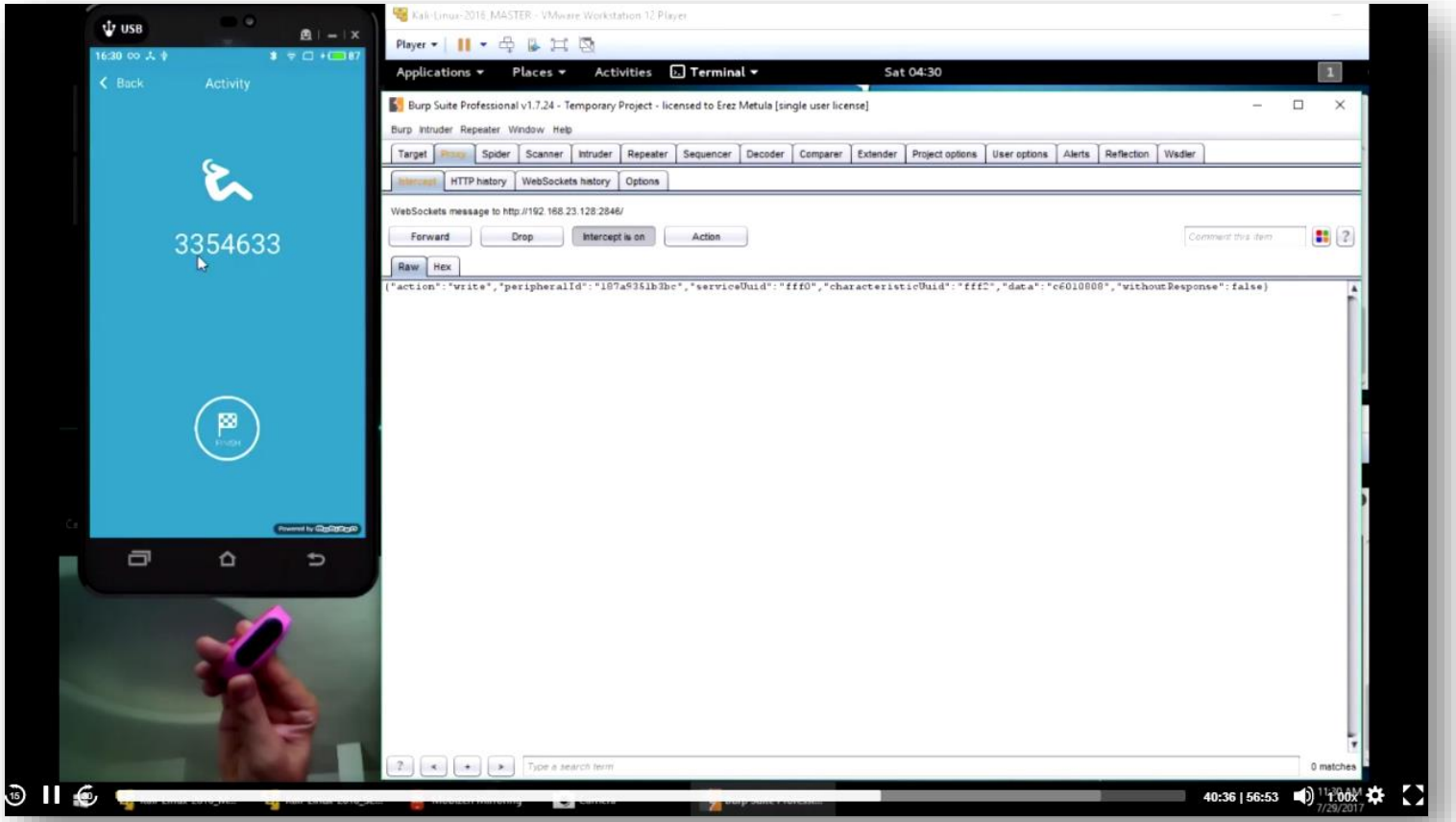



root@chaos:~/ble
figlet Hijack Connection
root@chaos:~/ble
btlejack -s
[0] 1:bash*
emanuel@chaos | 0.45 0.37 0.36 | 2020-02-26 11:14
ThinkPad

Example BLE Attacks

SHA2017 – Hack-a-ble

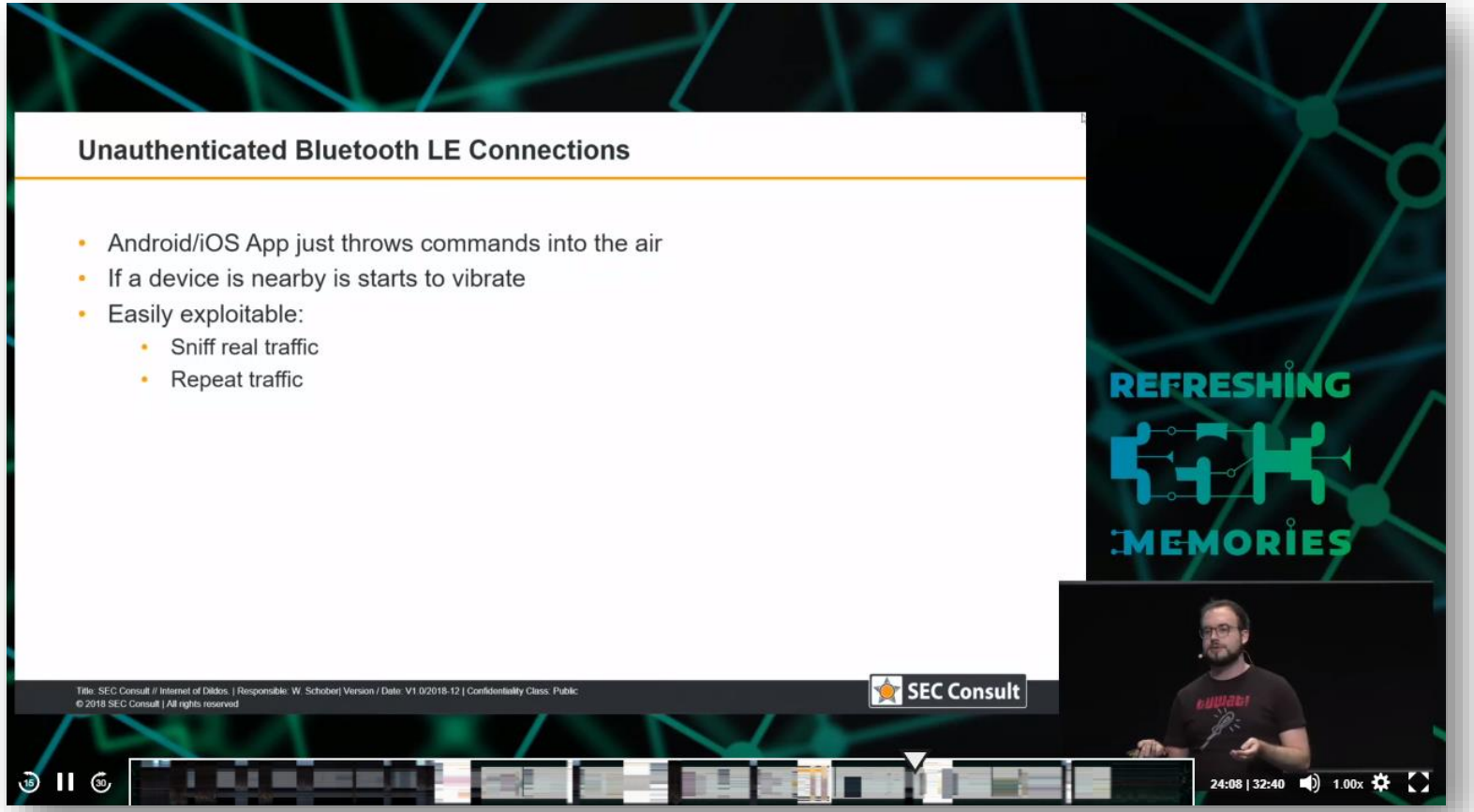
- General BLE security talk by Tal Melamed
- Example: Man-in-the Middle of a fitness watch



https://media.ccc.de/v/SHA2017-230-hack-a-ble

35c3 – Internet of Dongs

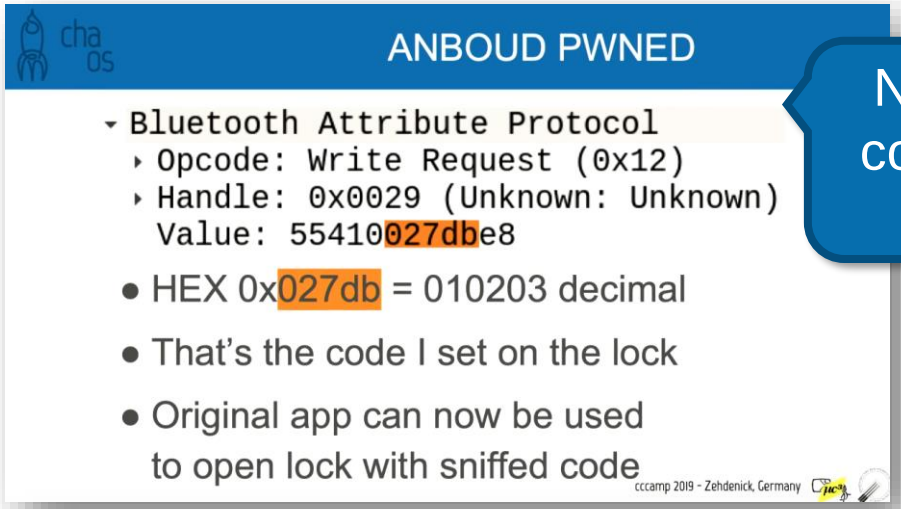
- Sex toy research that also covers BLE by Werner Schober
- No pairing at all (= no authentication): Let other's sex toys vibrate



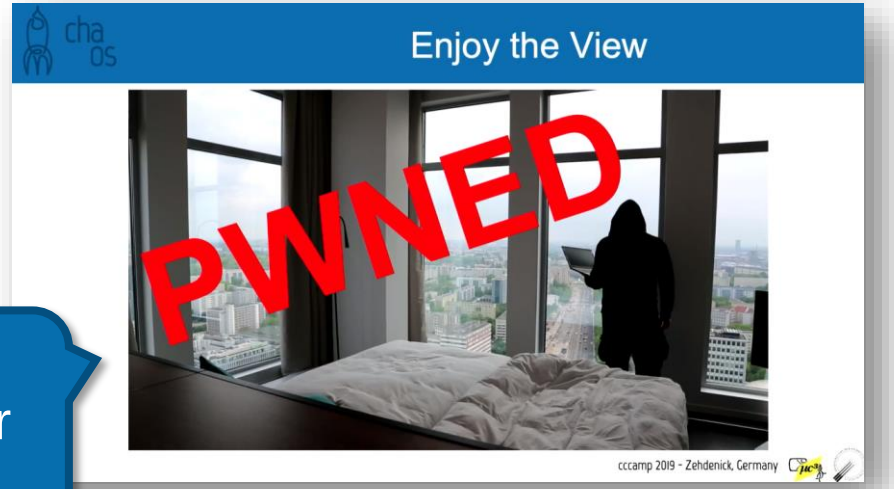
https://media.ccc.de/v/35c3-9523-internet_of_dongs

CCCamp2019 – Taking Bluetooth lockpicking to the next level

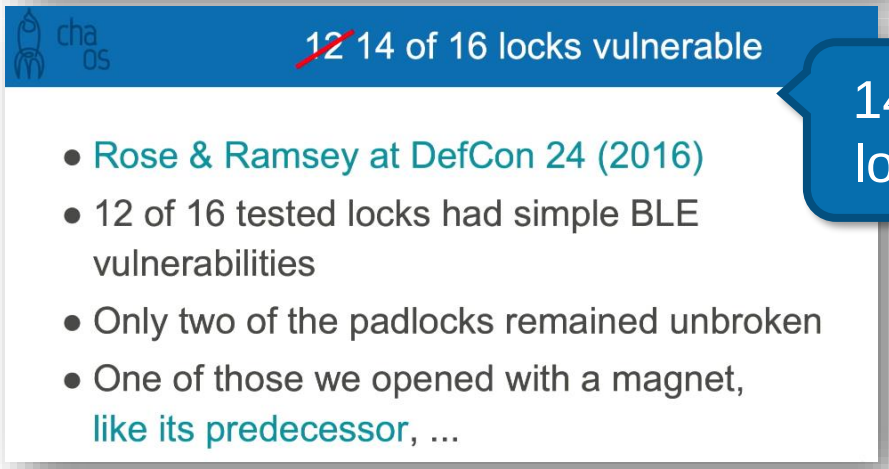
- BLE SmartLocks Lockpicking talk by Ray and mh



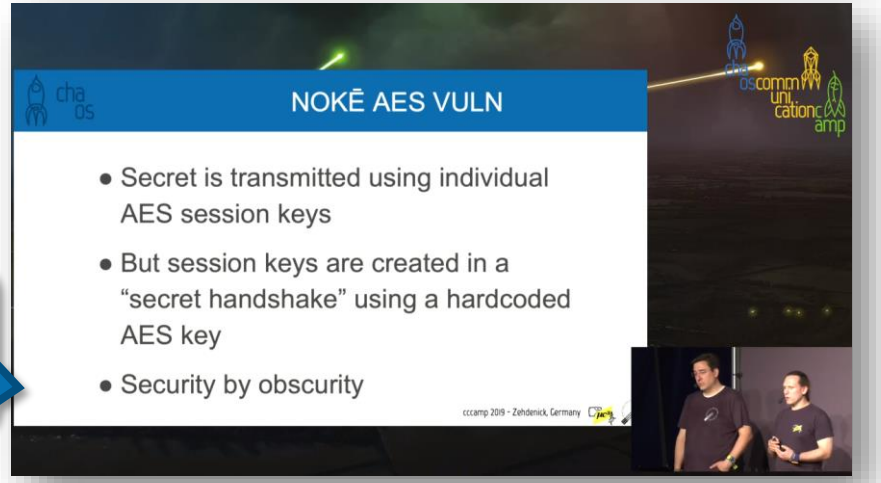
No pairing, lock code transmitted in cleartext



No pairing, code transmitted in clear in custom protocol



14/16 other analyzed locks had BLE vulns.

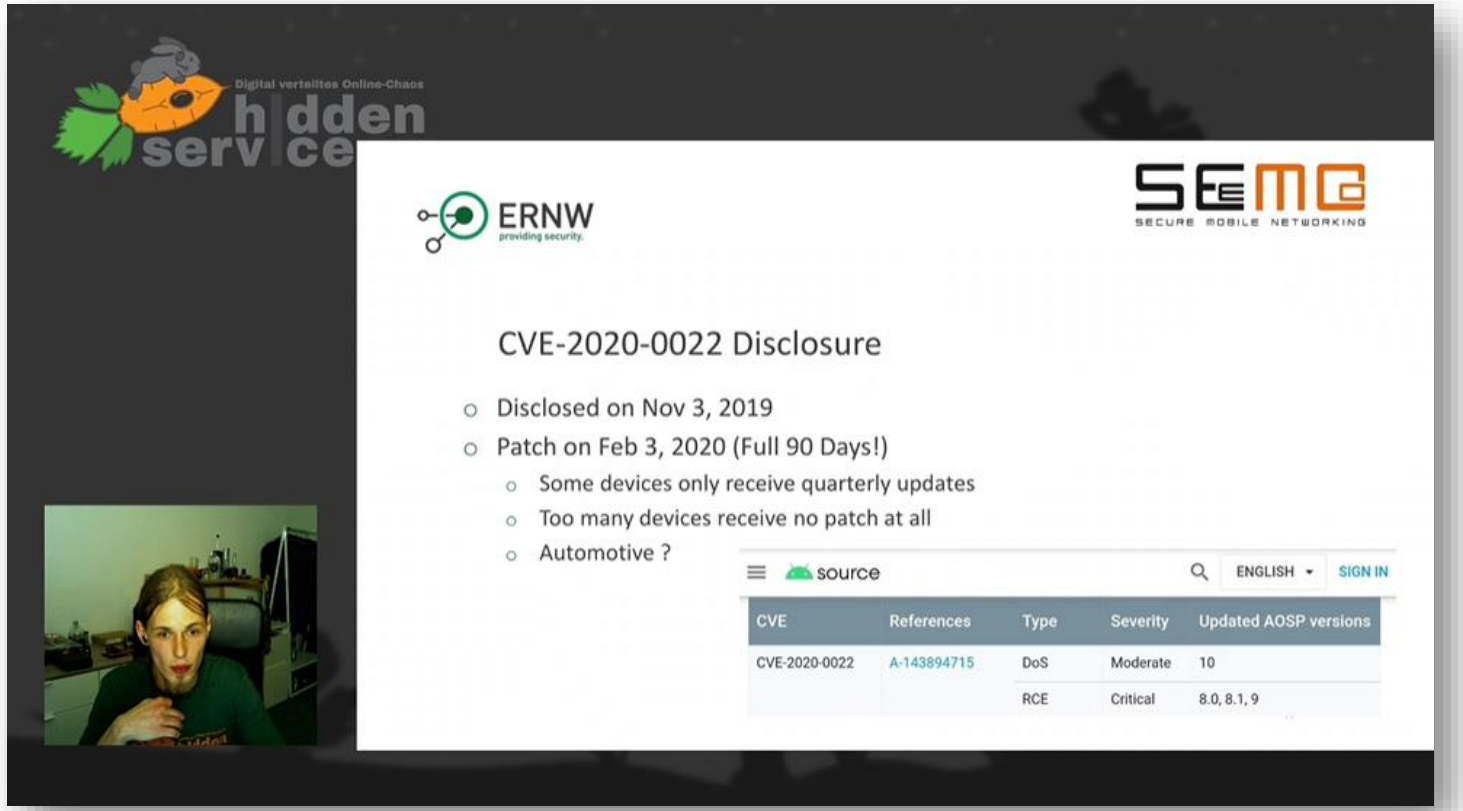


No pairing, hardcoded AES Key

https://media.ccc.de/v/Camp2019-10241-taking_bluetooth_lockpicking_to_the_next_level

Implementation Bugs

- There are several implementation bugs
- Example: BlueFrag vulnerability (CVE-2020-022) discovered by Jan Ruge
- RCE on all Android phones (version 8-9) when Bluetooth is just enabled



https://media.ccc.de/v/DiVOC-7-no_poc_no_fix_a_sad_story_about_bluetooth_security

Bluetooth Low Energy 5

Bluetooth Low Energy 5

- Version 5 released in 2016, Version 5.1 and 5.2 released in 2019
- Features: better speed, better range, improved coexistence
- In 2019: No BLE 5 products available in markets
  - Researcher has to build own BLE 5 devices in order to hack it

Core Specification Change History

Bluetooth

9 CHANGES FROM v4.2 TO v5.0

9.1 NEW FEATURES

Several new features are introduced in version 5.0. The major areas of improvement are:

- Slot Availability Mask (SAM)
- 2 Msym/s PHY for LE
- LE Long Range
- High Duty Cycle Non-Connectable Advertising
- LE Advertising Extensions
- LE Channel Selection Algorithm #2

Interesting
DEF CON 27 Talk

Physical Layers

- Two new physical layers
  - 2M LE Uncoded PHY: Better throughput up to 2 Mbps
  - LE Coded PHY: 4 times the range (125 kbps, up to 400m) or 2 times the range (500 kbps, up to 200m)
- Not supported by BtleJack at the moment, another chip would be needed

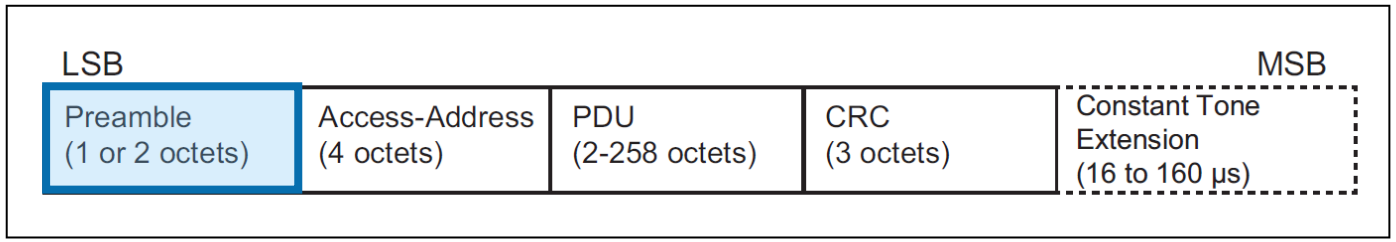


Figure 2.1: Link Layer packet format for the LE Uncoded PHYs

2 Octet Preamble:
2M LE Uncoded PHY

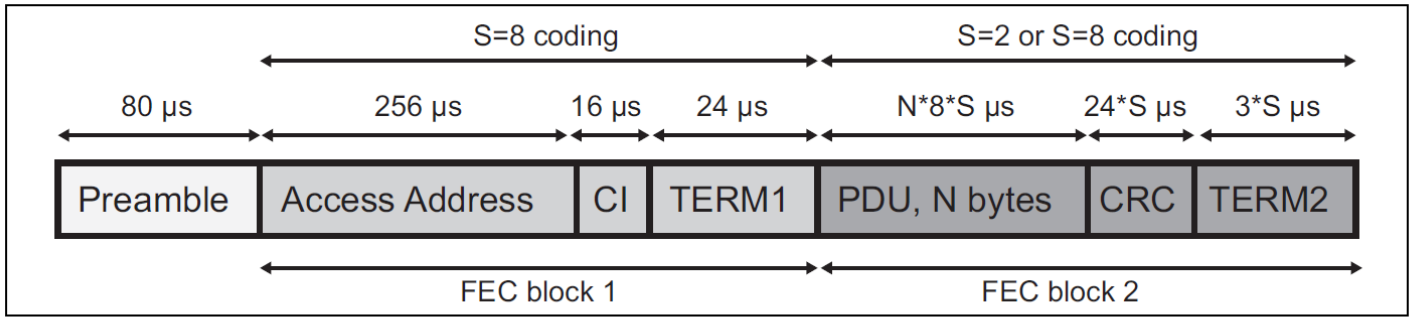
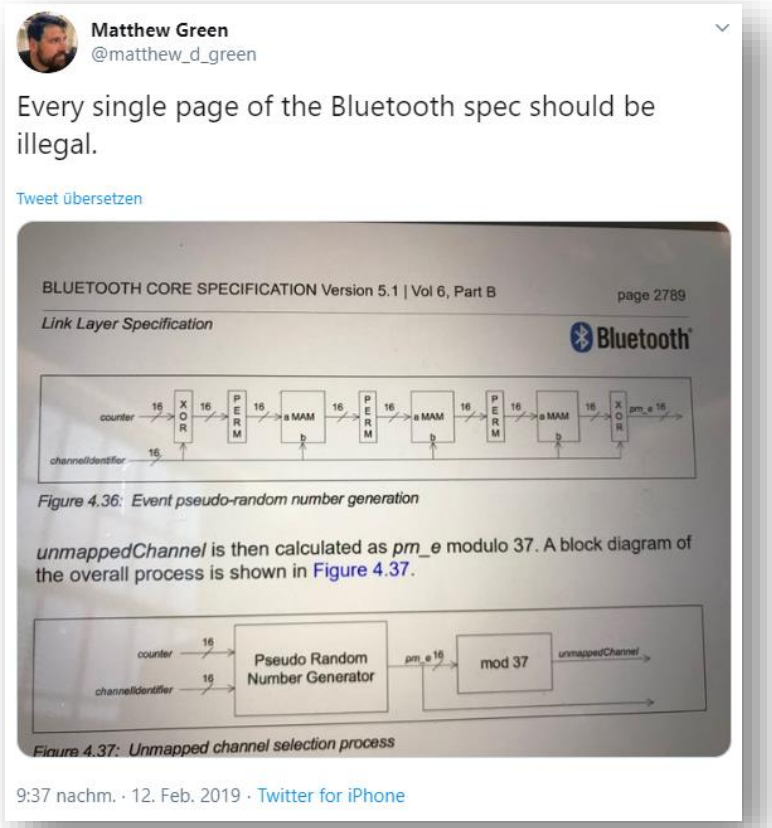


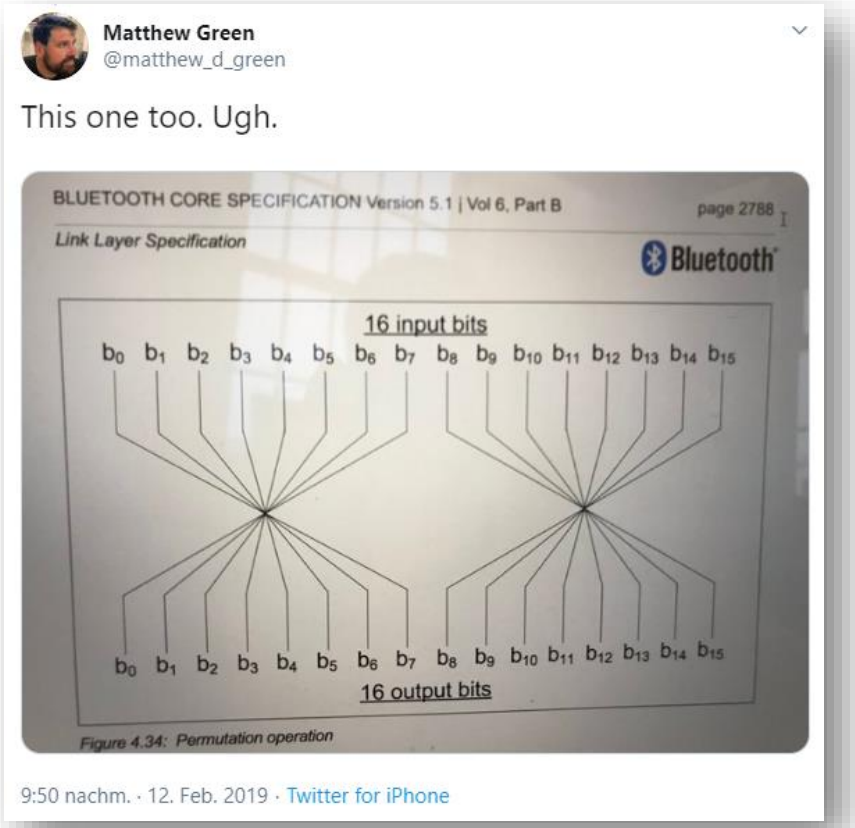
Figure 2.3: Link Layer packet format for the LE Coded PHY

LE Coded PHY

Channel Selection Algorithm

- New Hopping Scheme / Channel Selection Algorithm (CSA #2)
  - More random by using a Pseudorandom Number Generator (PRNG)
  - Devices specify in the advertisement packages if they support this (ChSel bit)
  - 65536-hop instead of 37-hop sequence





Easy, right?

Channel Selection Algorithm

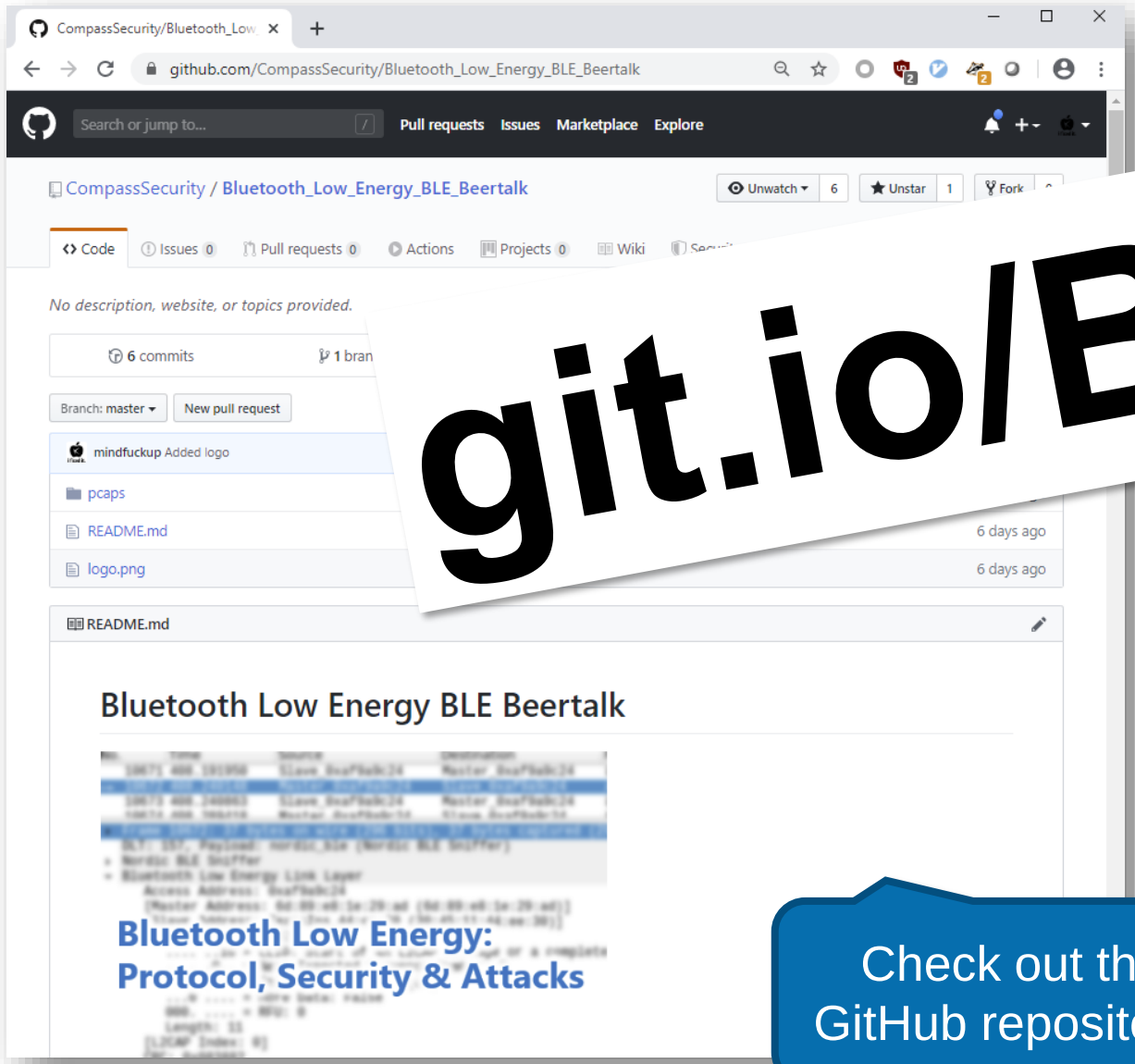
- Channel = PRNG(Channel Identifier, Counter) mod 37
- Channel Identifier (16 bit)
  - Can be calculated from the Access Address (split in 2 and XOR)
- Counter (16 bit)
  - Periodically incremented by 1
- The counter can be guessed by measuring time difference between consecutive channels and some math™
- Knowing both, it’s possible to follow the connection
- Used to improve coexistence, not security!
- Implemented in BtleJack version 2.0

BLE 5 Attacks

- No sniffing devices for the new physical layers at the moment
- Sniffing new connections is possible
- Sniffing existing connections is possible
- Jamming existing connections is possible
- Hijacking existing connections is theoretically possible
  - Not implemented in BtleJack at the moment because the attack is time-sensitive

References

Resources @ https://github.com/CompassSecurity



git.io/BLE

Check out the GitHub repository!

- Slides (full version!)
- Links to demo videos
- Links to this Beer-Talk video
- Links to software / hardware
- Example PCAPs
- Links to further resources


COMPASS
SECURITY