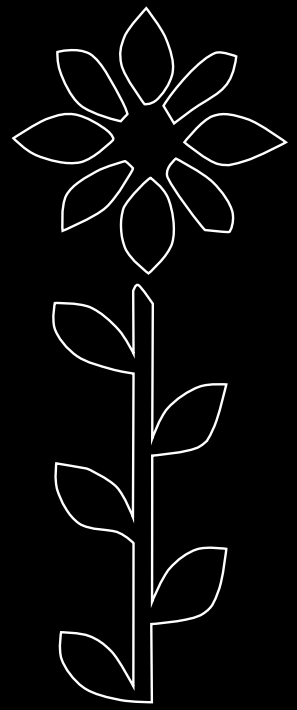
Doping your Fitbit

Firmware Modifications Faking you Fitter



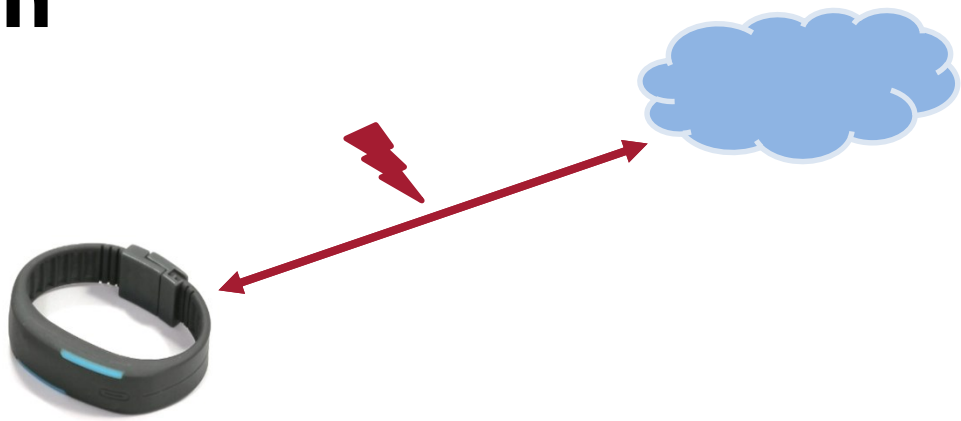
Jiska Classen & Daniel Wegemer

Technische Universität Darmstadt
Secure Mobile Networking Lab - SEEMOO
Department of Computer Science

Motivation

Most fitness trackers…

- Do not encrypt **local connections**.
- Apps require data upload **to the cloud**.



Why **Fitbit**?

- Market leaders: Apple, Xiaomi and Fitbit (~70 million devices)
- Interesting ecosystem, including end-to-end encryption ☺
  → Lessons learned apply to many IoT systems.
- Their security model requires sharing your data with them ☹

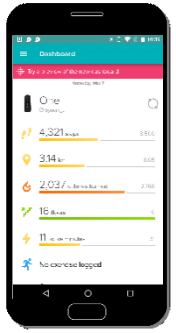
system overview

Communication Paradigm

Tracker

BLE

App



HTTPS

Server

Device-specific symmetric key

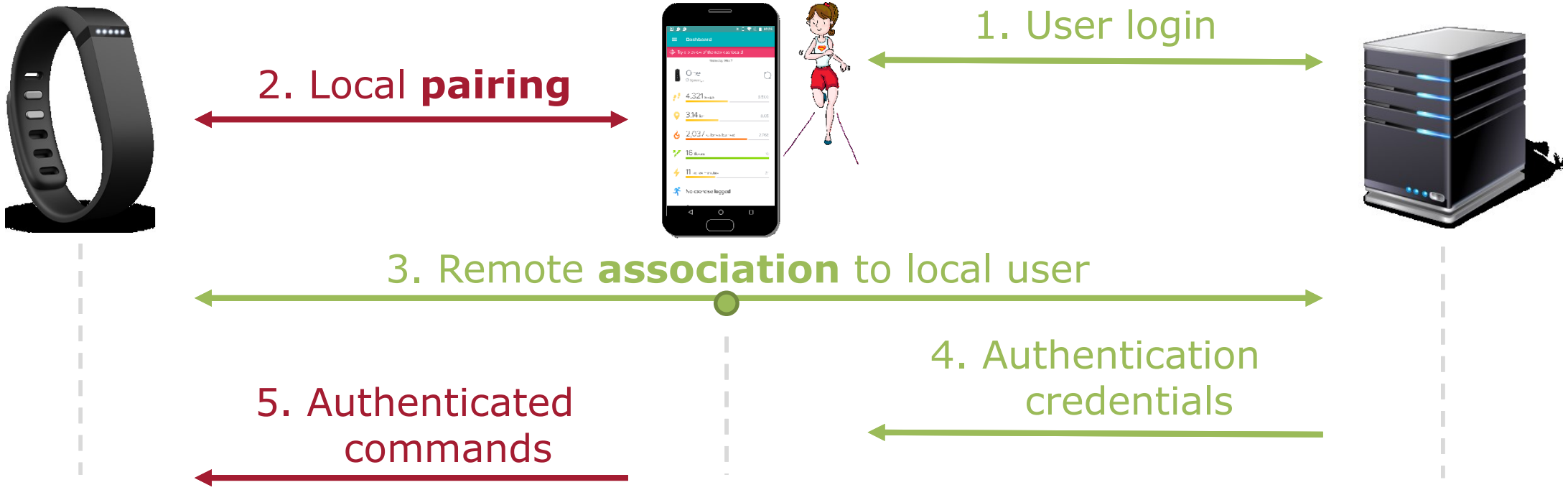
End-to-end encryption

Recent trackers only …

Memory read attack in firmware before October 2017

Encryption Flaws

Association & Authentication



- User **associates local tracker** with remote server account
  - Requires entering a code displayed on tracker or physical tapping
- App receives **authentication credentials** and stores them locally
- App can use authentication credentials for **authenticated local commands**

Remote Association Replay

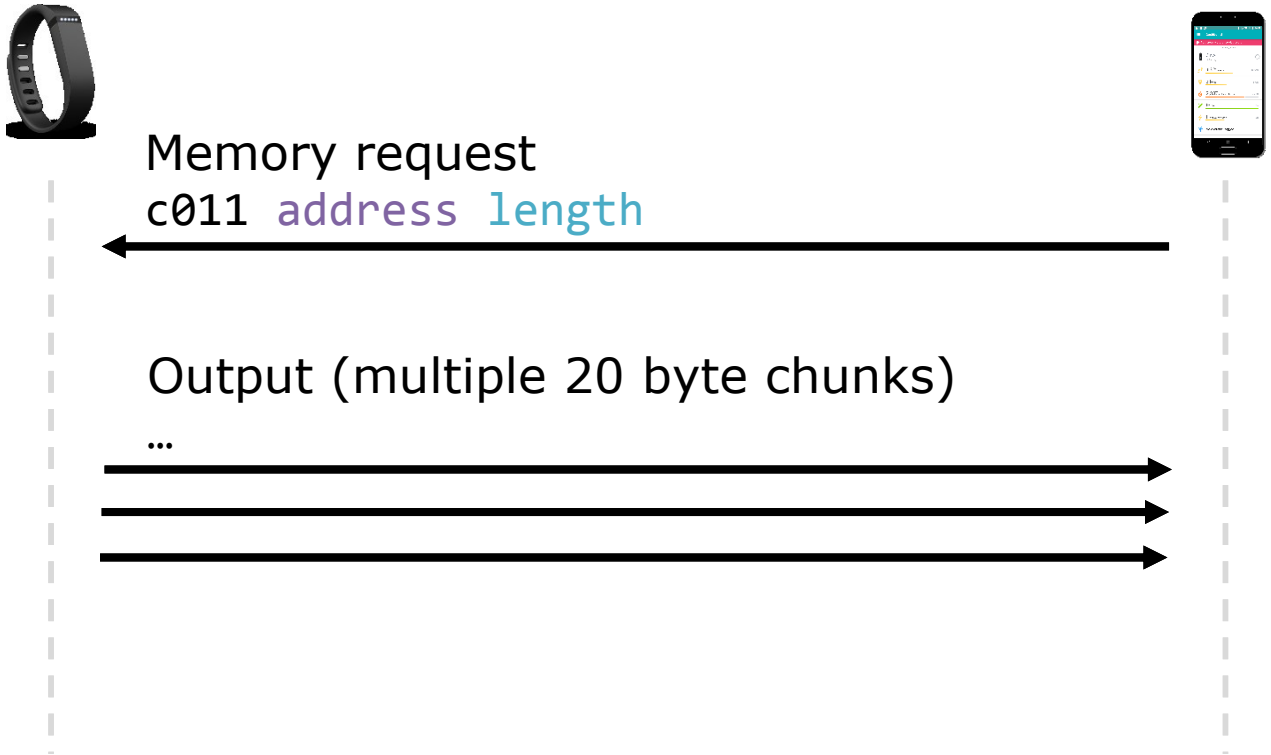
Associating a tracker should require physical presence!

- **PIN** displayed on tracker is entered into the app, **server-side comparison**.
- Tapping-only trackers send **local confirmation of tapping**.

- No confirmation of freshness, **replay possible**.
- Plaintext associations only require knowledge of **serial number**, which is **printed** on the original packing.
- **Authentication credentials** depend on persistent device key, they stay **valid forever**.

Authenticated Memory Readout

- Present in old Charge and Charge HR firmware, discovered by binary diff of firmware update: Read memory, including configurations.
- Update 6.44 and 7.88 (October 2017): **Fix for One & Flex**



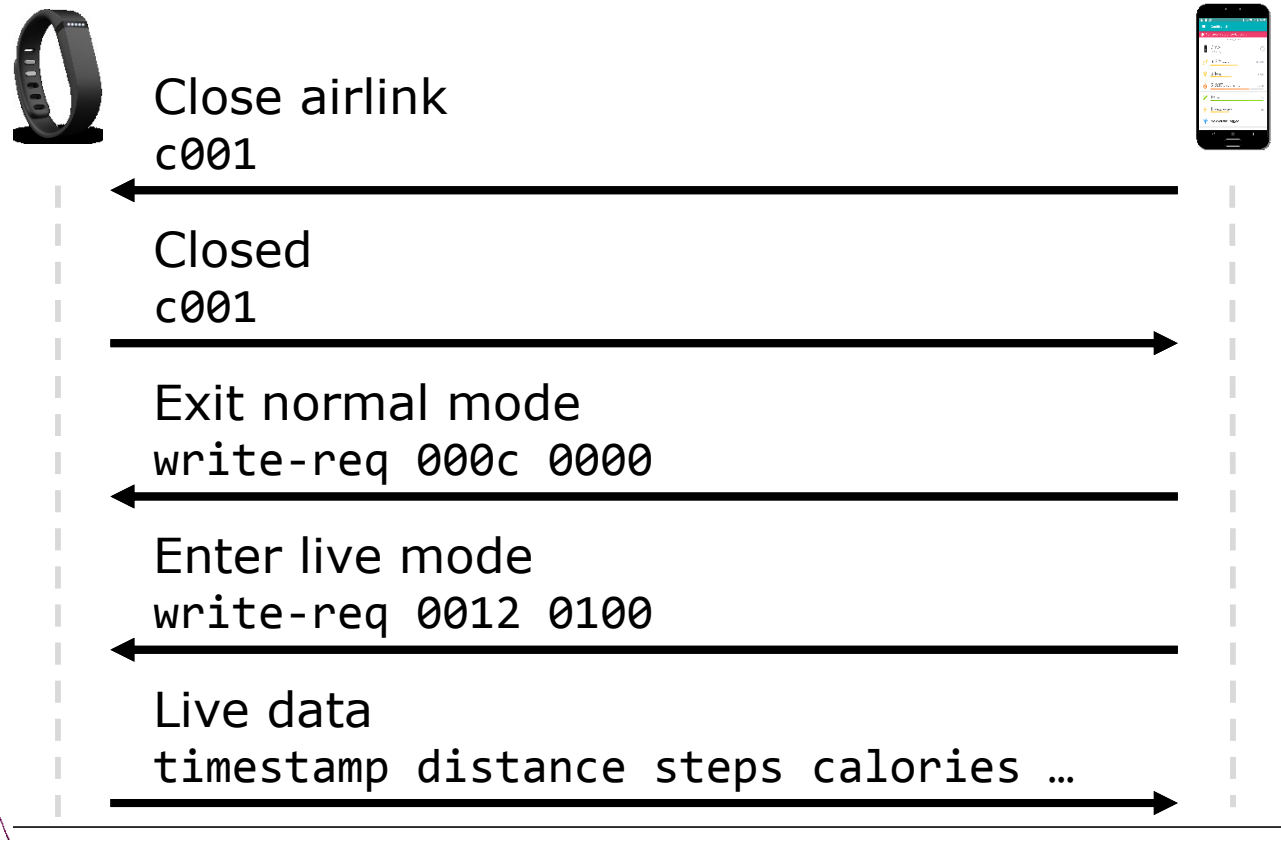
Reading encryption key enables:

- Server independence
- Encrypted fake packets

M. Schellevis, B. Jacobs, C. Meijer, J. de Ruiter: **Getting access to your own Fitbit data.** *2016.*

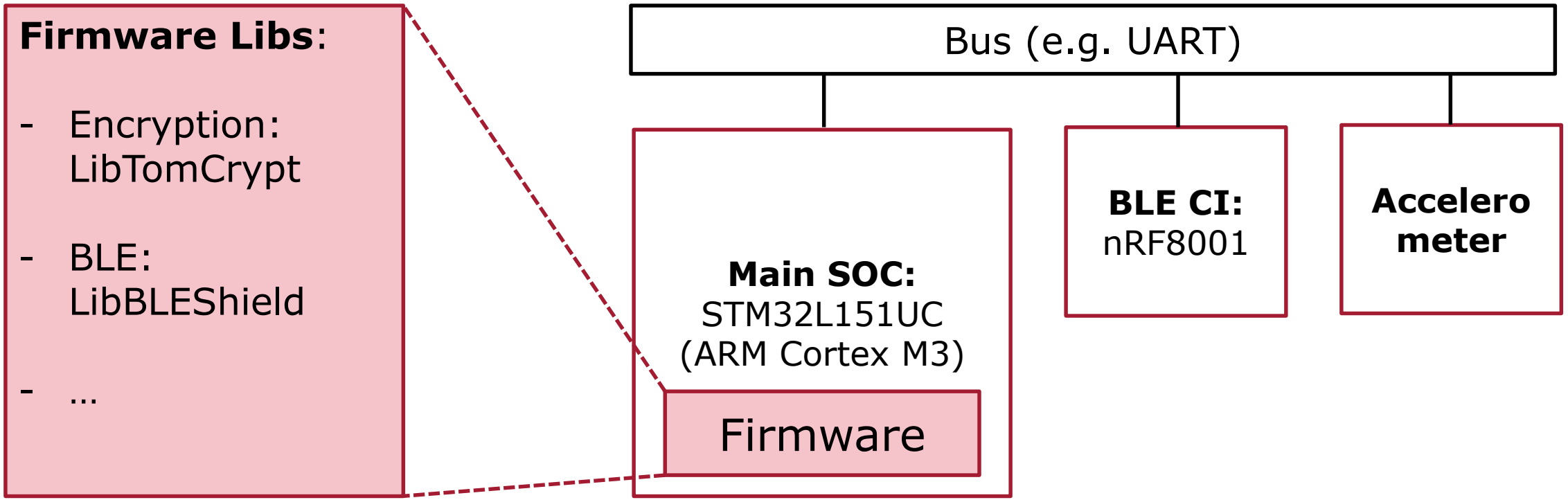
Authenticated Live Mode

- Local **plaintext connection** to the app, showing current activity summary.
- Update for all trackers, Alta … Surge (October 2017): **Optionally disable live mode**, but we even saw live mode in Ionic smartwatch logs…



Hardware Access

Fitbit Flex Hardware & Software



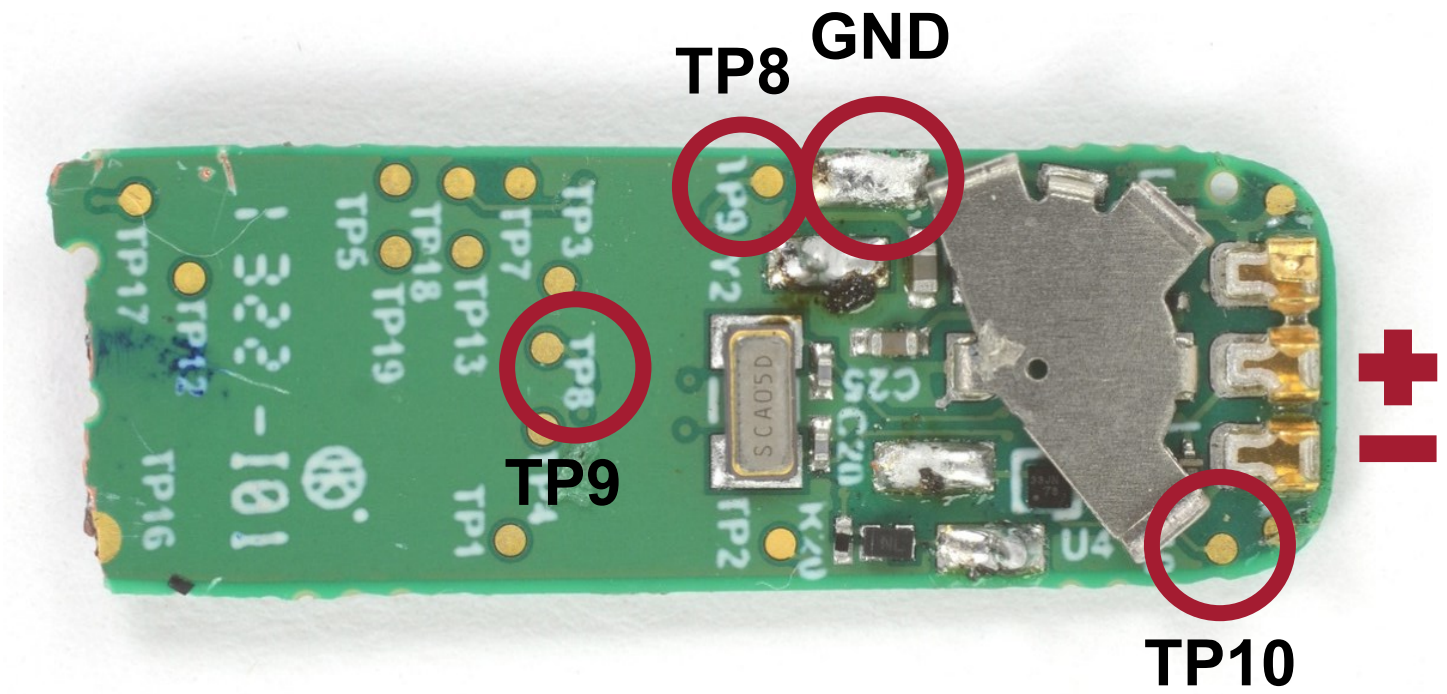
Hardware Access

Testing Points to connect to debugger:

- TP8 SWDIO
- TP9 SWCLK
- TP10 NRST
- GND (from battery)

Goals:

- Dump Firmware
- Modify stored data



Memory Layout

Flash

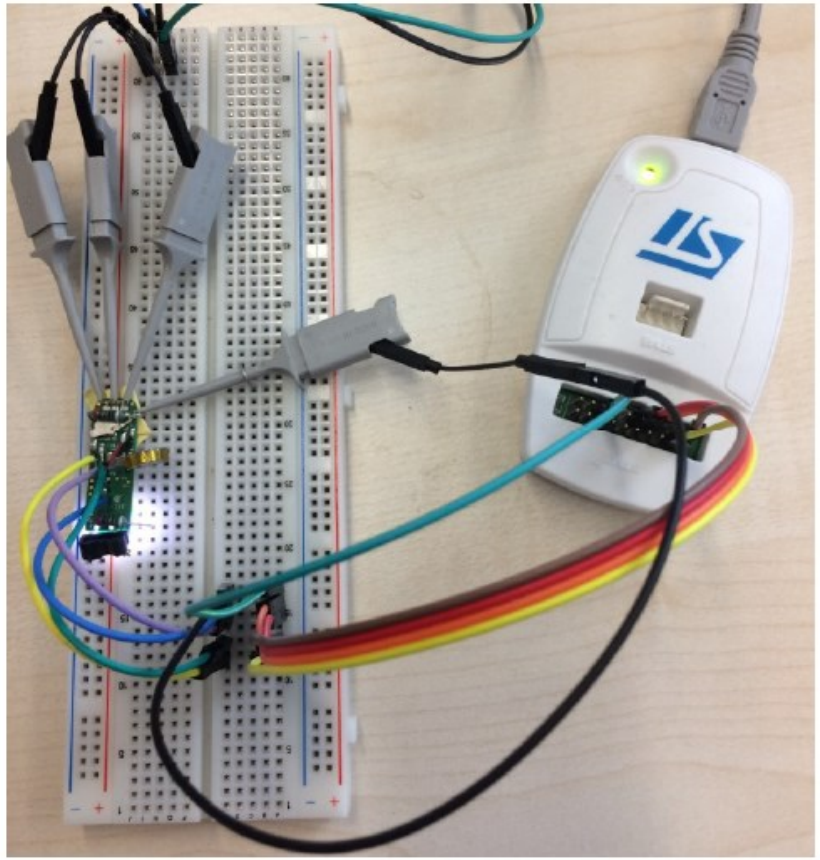
- Firmware code

EEPROM

- Information that should survive empty battery

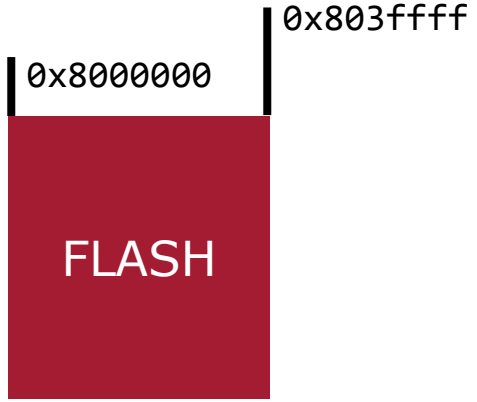
SRAM

- Firmware variables



Flash Contents

- **BSL** ~ 500 functions
- **APP** ~ 1000 functions (including BSL duplicates)
- Both BSL and APP run **independently**



- **Serial number**
- Encryption **key**
- Encryption switch
- Fitness data

Enabling GDB Access

Debugger Access

- Debugging is only enabled **during reset**
- Firmware initialization **disables GPIO ports** necessary for debugging
- Lets reset them!

How? Nexmon!

- Nexmon is a **binary patching framework**
- We adapt Nexmon for the Fitbit firmware
- Goal:
  - **Modify** firmware
  - Enable **dynamic debugging** (GDB)

Wireless Firmware Flashing

Update Process

Fitness record synchronization includes FW version

New FW available

Tacker

FW request

BSL

`https://.../firmware.json`

Request: Microdump

Response: BSL, APP

Validate

Write to flash

Reboot to BSL

APP

...

Update Format (Plaintext)

Header	Total Length
30 02 00 00 00 00 01 00 00 00 (**encryption options**, nonce)	40 00 00 00

Tracker	Chunk	Memory Address	Length	Length
06 (One) 07 (**Flex**) 12 (Charge HR)	01 (BSL) 02 (**APP**) 03 (reboot BSL) 04 (reboot APP)	F0 9F 00 08	10 00 00 00	10 00
		End of BSL/APP: 14 bytes zero Reboot: 14 bytes zero		

	Chunk CRC	BSL/APP Data
00 00	12 34	SLIP-encoding as in other dumps, size constraint ~ 65kB

Multiple chunks: APP firmware contains 3 data chunks, 1 empty chunk, 1 reboot chunk

Trailer	Length
56 78 00 00 00 00 00 00 (2 Byte **CRC** + padding / XTEA-EAX tag)	64 00 00

Additional Firmware Checks

Additional **checks** to be passed:

- **Address range** must stay within BSL or APP
- Additional **bit flip and CRC** within firmware

```
firmware[0x204] = 0x00
firmware[0x200:0x201] = crc(firmware[0:0x200] + firmware[0x208:])
```

- Failed firmware updates result to firmware version 0.00 in dumps…

Firmware & Dump Encryption

Older trackers use **XTEA in EAX mode** (One, Flex, Charge):

- **2 byte nonce** in beginning of each dump
- **128 bit encryption key**, extractable from EEPROM via memory readout attack
- **8 byte authentication MAC** in the end of each dump before length field

→ Firmware is based on **LibTomCrypt** (C)

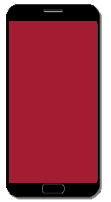
All functions are also available in **spongycastle** (Java).

Newer trackers use **AES** in EAX mode.

Based on own reverse-engineering and insights of M. Schellevis

Steps to Flash Modified Firmware

1. Get symmetric key
2. Get plaintext firmware binary



https://github.com/seemoo-lab/fitness-app

3. Transfer the firmware to your PC
4. Modifiy firmware using nexmon

nexmon

https://github.com/seemoo-lab/fitness-firmware

5. Format and encrypt firmware
6. Flash firmware to tracker

Affected Models & Versions

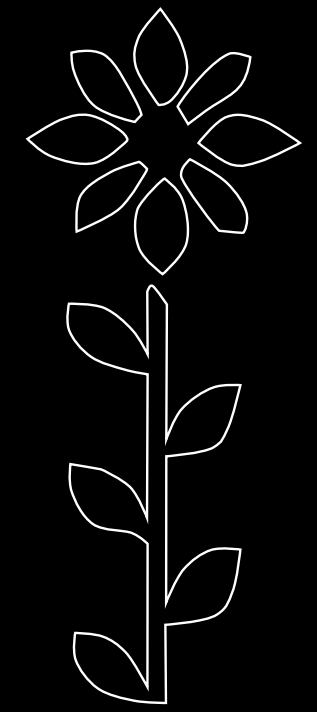
Encrypted wireless **firmware modifications** (requires memory readout):

Tracker	Firmware Version
One	5.60 (before October 2017)
Flex	7.81 (before October 2017)
Charge HR	18.102 (older…)

Live mode:

Security fix adds an option to disable live mode, introduced in October 2017 for **all tracker models**.

summary



Summary

1. Go out and flash your neighbors‘ devices
2. Keep control of your own data
3. Run any code on your Fitbit