BreakMi: Reversing, Exploiting and Fixing Xiaomi Fitness Tracking Ecosystem

Hardwear.io USA 2023

Marco Casagrande and Daniele Antonioli (EURECOM, FR)

EURECOM
Sophia Antipolis

hardwear.io

Daniele Antonioli

Assistant Professor at EURECOM (FR)

Research interests:

- Wireless Communication (Bluetooth, Wi-Fi, ...)
- Embedded (IoT, cars, ...)
- Mobile (Android, iOS, ...)
- Cyber-Physical Systems (ICS)

We are hiring PhDs, and Postdocs

Email: antonioli.daniele@gmail.com

Website: https://francozappa.github.io

2nd time speaker at HWIO, Bluetooth 0-days talk in 2020

Marco Casagrande

PhD student at EURECOM (FR)

Research Topics:

- Bluetooth / Bluetooth Low Energy
- Internet-of-Things
- Android

Email: marco.casagrande@eurecom.fr

Talk Outline

- Intro on proprietary fitness tracking ecosystems
- Reverse engineering (RE) methodology
- Xiaomi FTE vulns and attacks
- BreakMi OS toolkit and (live) demos
- Fitbit FTE vulns and attacks
- Countermeasures and responsible disclosure

Acknowledgements

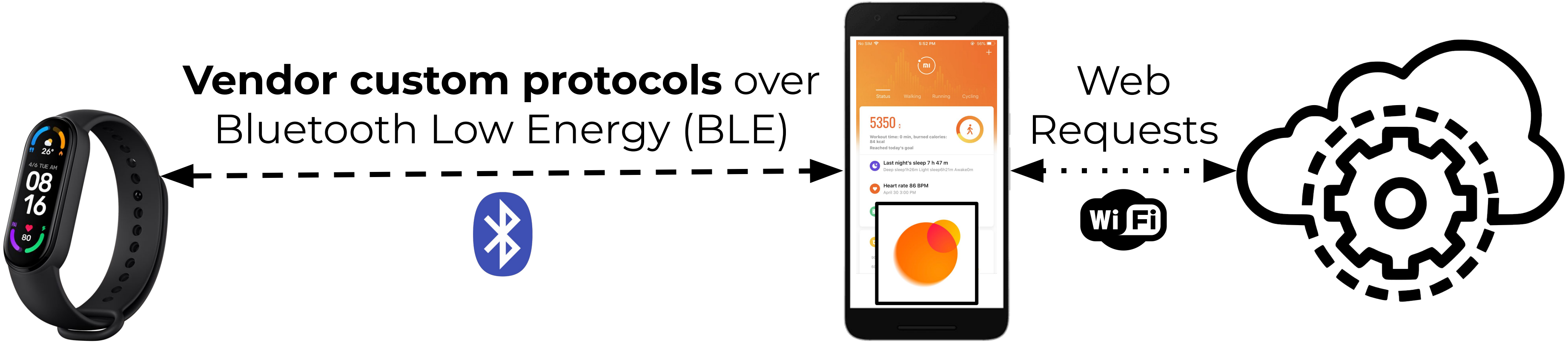
Eleonora Losiouk
Assistant Professor at University of Padova (IT)

Mauro Conti
Professor at University of Padova (IT)

Mathias Payer
Associate Professor at EPFL (CH)

INTRODUCTION

Fitness Tracking Ecosystem



Vendor custom protocols over Bluetooth Low Energy (BLE)

Web Requests

Fitness Tracker

Companion App

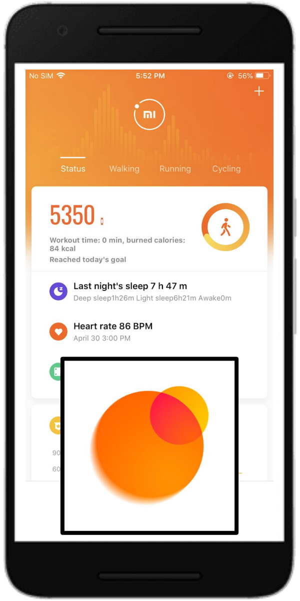
Vendor Backend

Fitness Tracker (FT)

- Wearable IoT device with sensors
  - Monitors **sensitive** data
  - E.g., steps and heart rate
  - Controls smartphone lock screen
  - Displays SMSes and alerts
  - BLE connection to smartphone app

FT Companion App

- Interact with the FT
  - Connect
  - Read sensor values
- Gateway to the backend
  - Send sensor and user values



5350

Workout time: 0 min, burned calories: 84 kcal

Reached today's goal

Last night's sleep 7 h 47 m

Deep sleep1h26m Light sleep6h21m Awake0m

Heart rate 86 BPM

April 30 3:00 PM

FT Backend

- Internet-accessible infrastructure
  - Registered users
  - Registered (paired) devices per user
  - Backups
  - FT firmware

Bluetooth Low Energy (BLE)

- De-facto standard protocol for IoT devices
  - E.g., trackers, watches, ...
- Device discovery
  - *Scanner* (App)
  - *Advertiser* (FT)
- Connection establishment
  - *Central* aka Initiator (App)
  - *Peripheral* aka Responder (FT)
  - Client-server data model (GATT)

BLE Scanning and Advertising

- App (scanner) scans for advertisers
- FT (advertiser) periodically **broadcast** presence
- **Advertising** packets
  - Contain data to **connect** to the advertiser
  - E.g., BLE MAC address, device name, list of service UUIDs, manufacturer’s data

BLE Generic Attribute Profile (GATT)

- GATT defines client-server communication
  - **Hierarchy** format of **services** and **characteristics**
  - Each one identified by UUID
- Service = **feature** granted by GATT server
  - E.g., Heart Rate Service
  - **Collection** of characteristics

BLE Generic Attribute Profile (GATT)

- Characteristic = single **data point**
  - E.g., Heart Rate Measurement Characteristic
  - Defined by Attribute Profile (ATT)
- ATT defines how data is represented/interacted
  - Characteristic value
  - Characteristic **read/write/notify** permissions

BLE Link-Layer Security

- **Pairing**
  - Agree on a long-term pairing key
  - Usually happens only once
- **Session establishment**
  - Derive a session key from the pairing key
  - Encrypt the communication using the session key
- Vendors can
  - Enable/disable BLE link-layer security
  - Provide application layer security on top

FT Ecosystem Security (1)

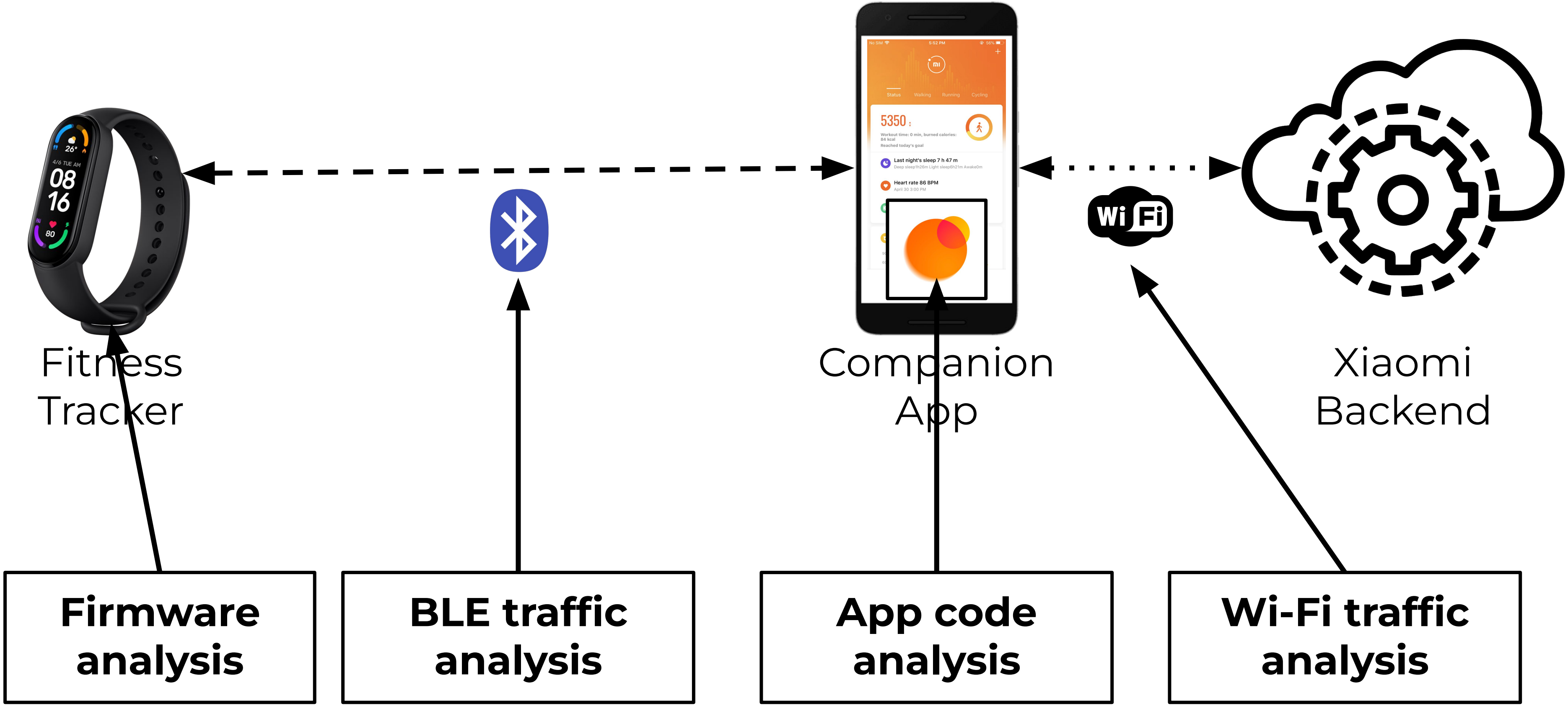
- Security risks
  - E.g., **tamper** with BLE packets
  - E.g., **data loss** due to factory reset
- Privacy risks
  - E.g., leaking sensitive **health data** (e.g., heart rate)
  - E.g., reading **2FA** messages

FT Ecosystem Security (2)

- **Proprietary** protocols spoken over BLE (or Wi-Fi)
  - Unknown custom security mechanisms
  - No public documentation
  - No test environment or tools available
- Need to **reverse-engineer** Xiaomi protocols to assess their security

RE METHODOLOGY

RE Targets



Fitness Tracker

Companion App

Xiaomi Backend

Firmware analysis

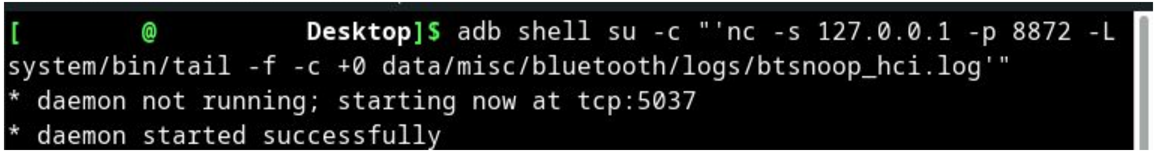
BLE traffic analysis

App code analysis

Wi-Fi traffic analysis

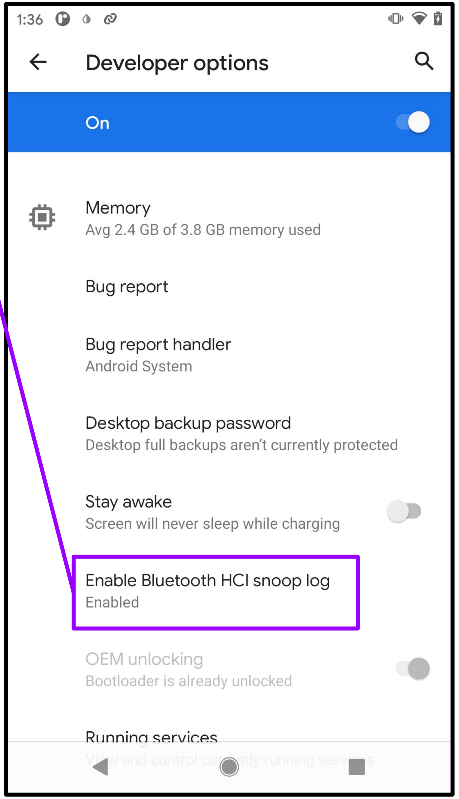
BLE Traffic Analysis

- Enable Android **BT HCI snoop log**
  - Capture file with BLE traffic
  - Enable Wireshark live capture



```
[[redacted]@[redacted] Desktop]$ adb shell su -c "'nc -s 127.0.0.1 -p 8872 -L
system/bin/tail -f -c +0 data/misc/bluetooth/logs/btsnoop_hci.log'"
* daemon not running; starting now at tcp:5037
* daemon started successfully
```

- Or use `adb bugreport my_report`



1:36

Developer options

On

Memory
Avg 2.4 GB of 3.8 GB memory used

Bug report

Bug report handler
Android System

Desktop backup password
Desktop full backups aren't currently protected

Stay awake
Screen will never sleep while charging

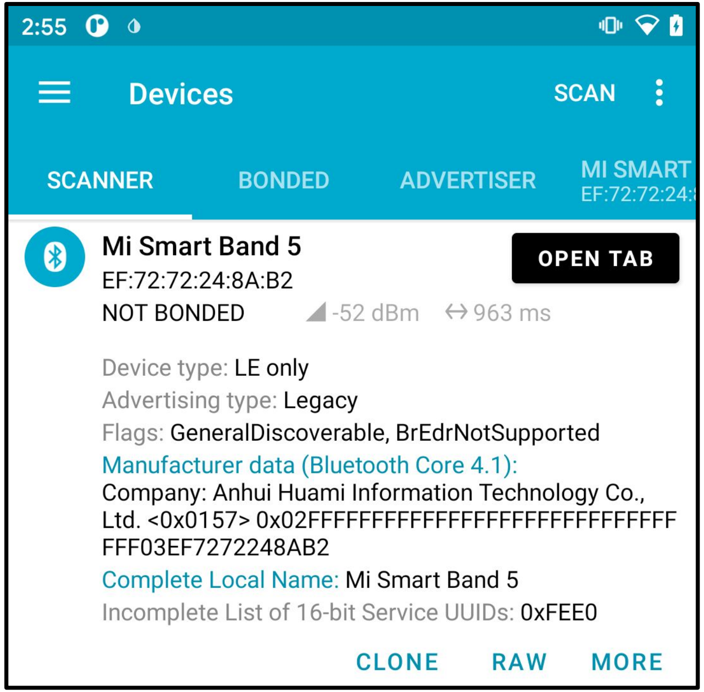
Enable Bluetooth HCI snoop log
Enabled

OEM unlocking
Bootloader is already unlocked

Running services
View and control currently running services

BLE Traffic Analysis - Advertising

- FTs periodically advertises if not connected
- Random **BLE MAC address**
  - Changes upon factory reset
  - App looks address to check If already paired or not

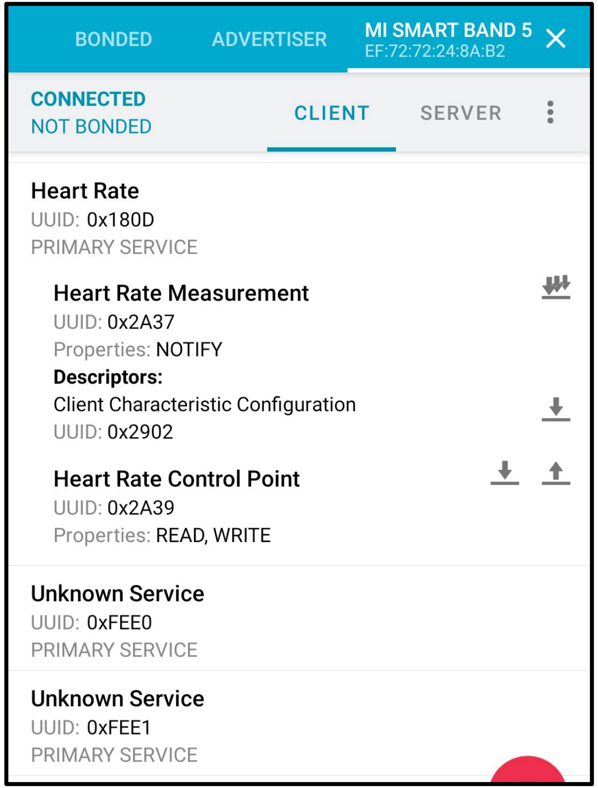


Mi Smart Band 5
EF:72:72:24:8A:B2
NOT BONDED -52 dBm ↔ 963 ms

Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Manufacturer data (Bluetooth Core 4.1):
Company: Anhui Huami Information Technology Co., Ltd. <0x0157> 0x02FFFFFFFFFFFFFFFFFFFFFFFFFFFFF03EF7272248AB2
Complete Local Name: Mi Smart Band 5
Incomplete List of 16-bit Service UUIDs: 0xFEE0

BLE Traffic Analysis - GATT

- Xiaomi GATT custom services
  - E.g., `0xFEE0`, `0xFEE1`
- **Heart Rate** and **Steps**
  - Protected by Xiaomi auth
  - `GATT READ NOT PERMIT`



CONNECTED
NOT BONDED

Heart Rate
UUID: 0x180D
PRIMARY SERVICE

Heart Rate Measurement
UUID: 0x2A37
Properties: NOTIFY
Descriptors:
Client Characteristic Configuration
UUID: 0x2902

Heart Rate Control Point
UUID: 0x2A39
Properties: READ, WRITE

Unknown Service
UUID: 0xFEE0
PRIMARY SERVICE

Unknown Service
UUID: 0xFEE1
PRIMARY SERVICE

BLE Traffic Analysis - Custom Packets

- **Binary** data payload inside BLE packets
- Custom **opcodes**
  - `Pairing Init`: `0100`
  - `Pairing Complete`: `100101`
  - `Pairing Key`: `0100||Key`
  - `User Confirmation`: `108301`
  - `Auth Chal`: `100201||Chal` or `108201||Chal`
- Protocol **dissectors** to automate detection

Firmware Analysis

- Retrieving FT **firmware** is not trivial
    - Debug port or intercept BLE firmware update
- Static code analysis with Ghidra/IDA
    - Lengthy, **stripped** binaries, manual work
- Challenging to **debug** dynamically

App Code Analysis

- Extracting app.**apk** from Android app is trivial
  - `adb shell pm path com.example.someapp`
  - `adb pull path/to/apk path/to/destination`
- Static code analysis with decompilers
  - Outputs accurate Java decompiled code
- Dynamic analysis is also **possible**
  - Dynamic binary instrumentation

App Static Analysis (1)

- App features and capabilities
  - Permissions (normal, dangerous)
  - Components (activities, services, receivers, providers)
  - Resource files and strings.xml
  - Networking (IPs, URLs, domains)

App Static Analysis (2)

- Code decompilation
  - Crypto/security **API calls**
  - E.g., `Cipher`, `MessageDigest`, `Random`
  - Logic of Xiaomi **custom classes**
  - E.g., `HMBaseProfile`, `HMWebBindInfo`, `HMDeviceWebAPI`
  - Presence of **obfuscation**

App Dynamic Binary Instrumentation

- Dump and hook code at runtime
  - Classes, methods, system calls, ...
- Monitor functions parameters and return values
  - **Compare** BLE traffic data with input/output values
  - Also **inject** values and logic inside such functions
- Print **stack traces**
  - E.g., going backwards to find which Xiaomi custom class invoked AES-ECB

Wi-Fi Traffic Analysis

- Intercept web traffic with Xiaomi backend
  - Deploy HTTPS **proxy**
  - Man-in-the-middle the traffic to read it
- Multiple Xiaomi **endpoints**
  - `account.xiaomi.com/oauth2/authorize`
  - `account.huami.com/v2/client/login`
  - `api-mifit-de2.huami.com/v1/device/binds.json`

XIAOMI FT ECOSYSTEM SECURITY EVALUATION

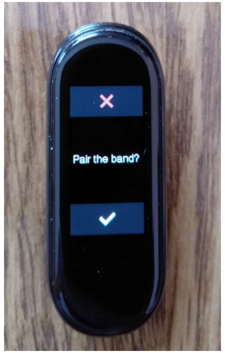
Xiaomi FTs

Mi
Band 2

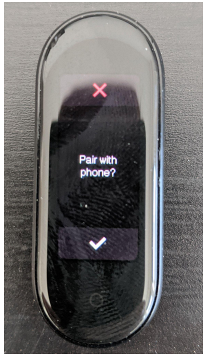
Mi
Band 3

Photo
N/A

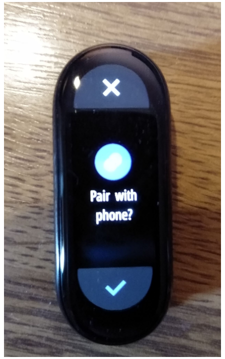
Amazfit
Cor 2



Mi
Band 4

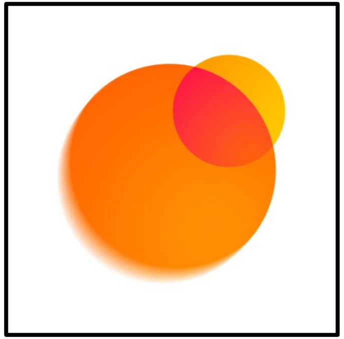


Mi
Band 5



Mi
Band 6

Xiaomi Companion Apps



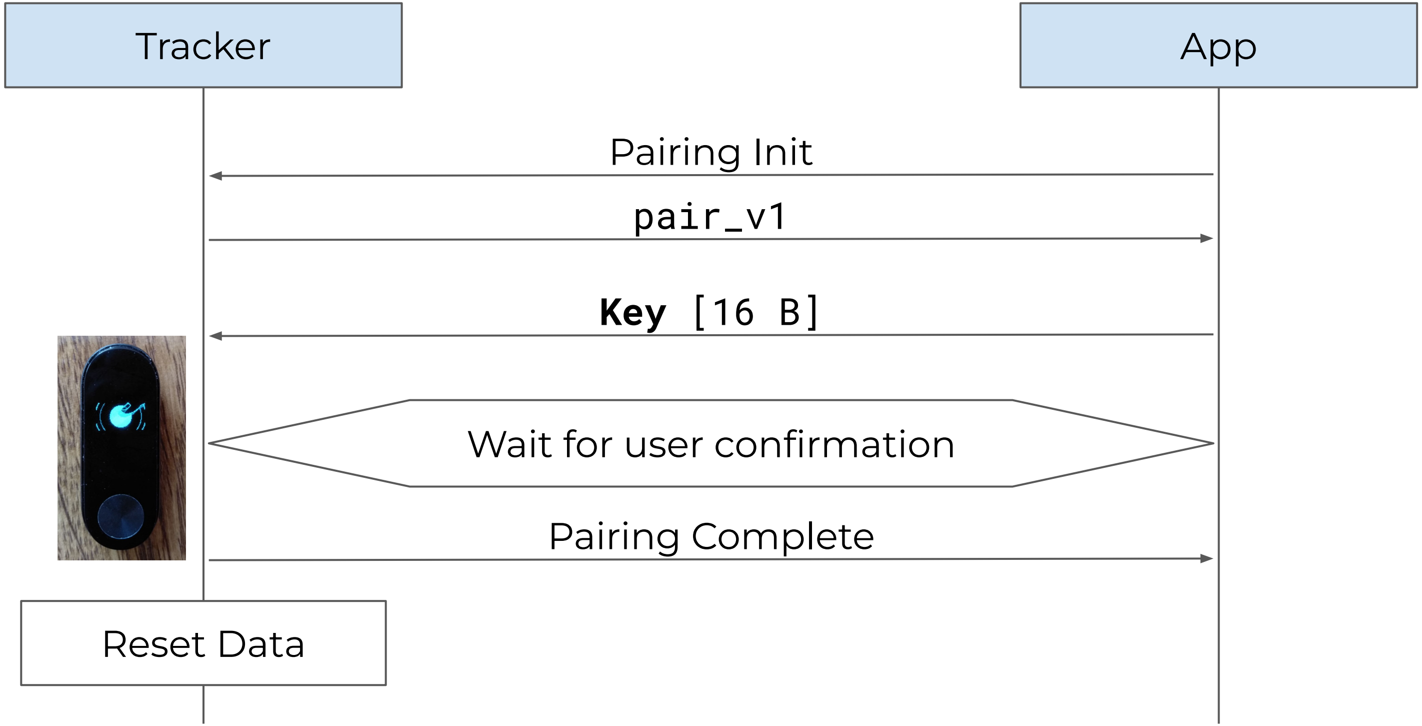
Zepp Life
(formerly Mi Fit)

Zepp
(formerly Amazfit)

Xiaomi Security Protocols

- BLE link-layer security?
  - **Disabled** by Xiaomi, despite device support
  - No link-layer confidentiality, integrity, and authenticity
- Xiaomi application layer security?
  - Custom binary protocols (Pairing, ...)
  - We found critical vulnerabilities (BLA)
  - And exploited them (BLA)
- Now we present them in detail

Xiaomi Pairing v1



Tracker

App

Pairing Init

`pair_v1`

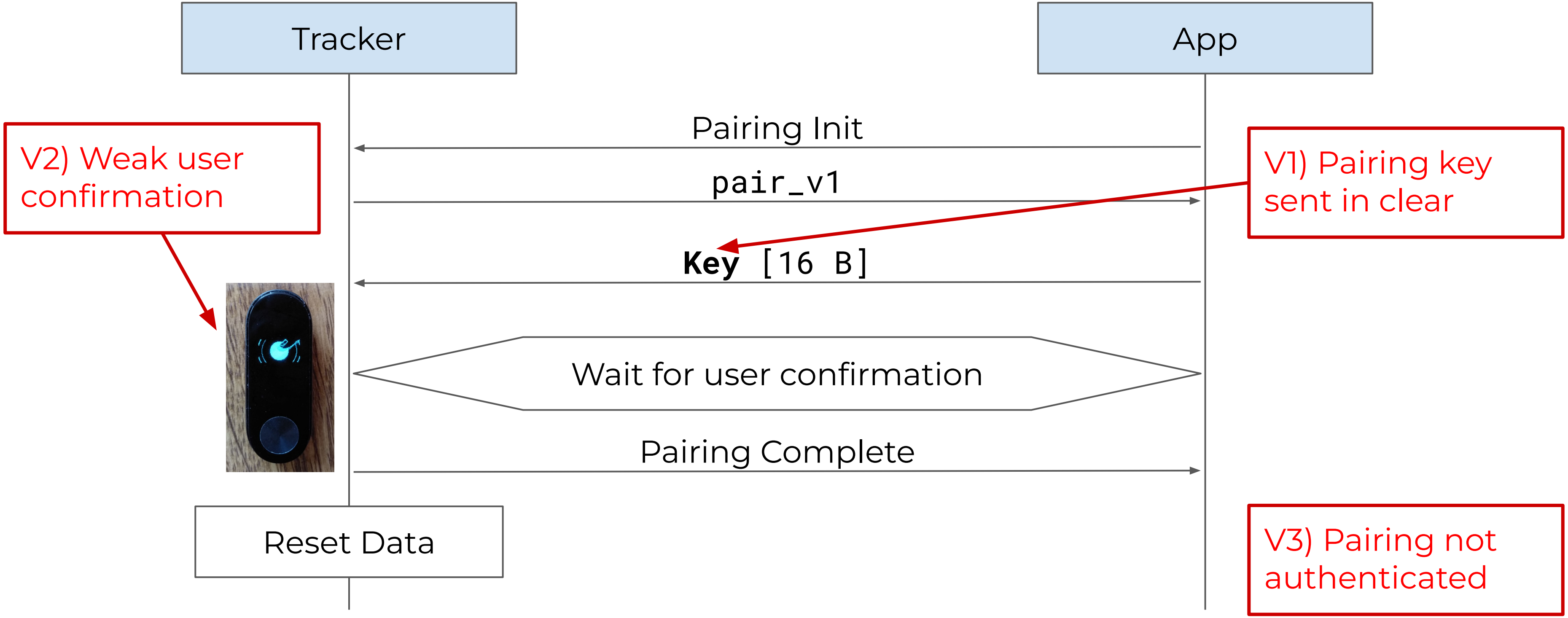
Key [16 B]

Wait for user confirmation

Pairing Complete

Reset Data

Xiaomi Pairing v1



Tracker

App

Pairing Init

pair_v1

Key [16 B]

Wait for user confirmation

Pairing Complete

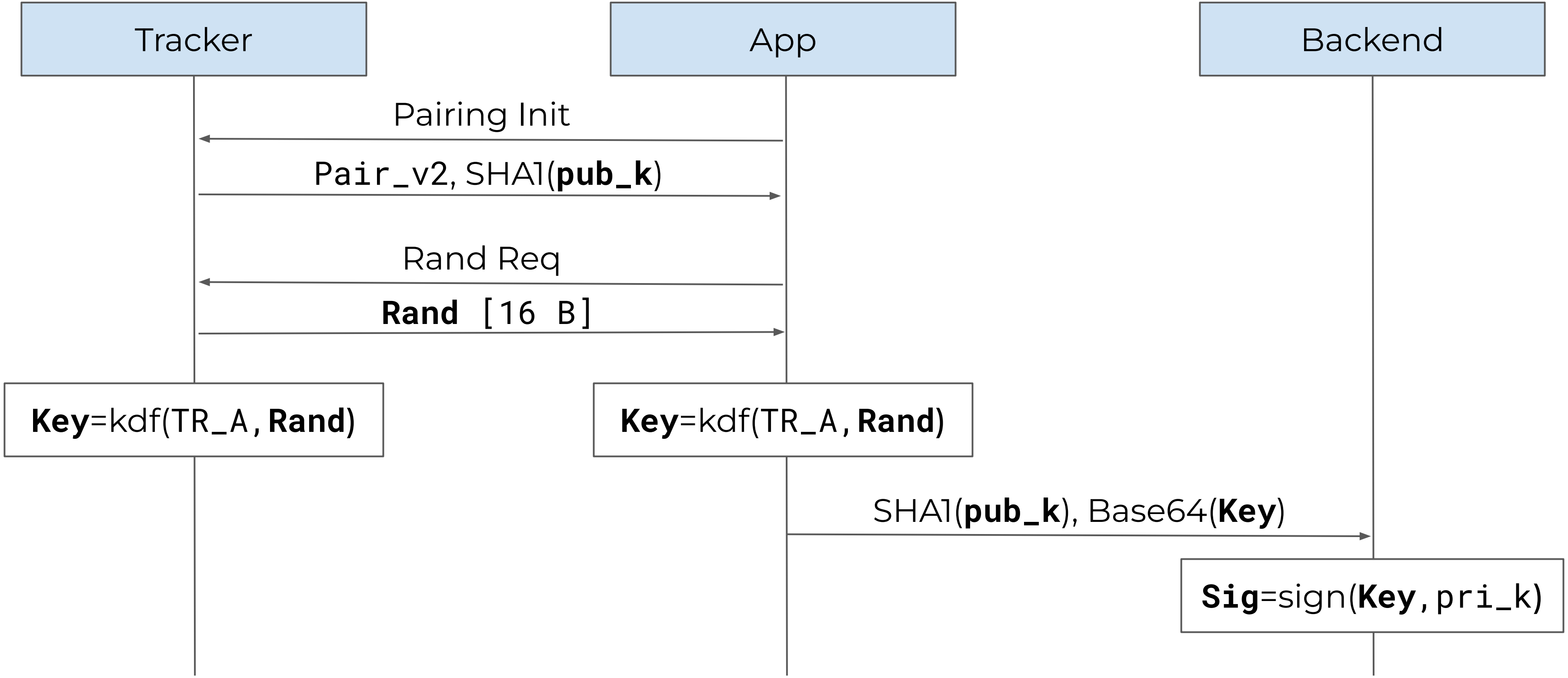
Reset Data

V1) Pairing key sent in clear

V2) Weak user confirmation

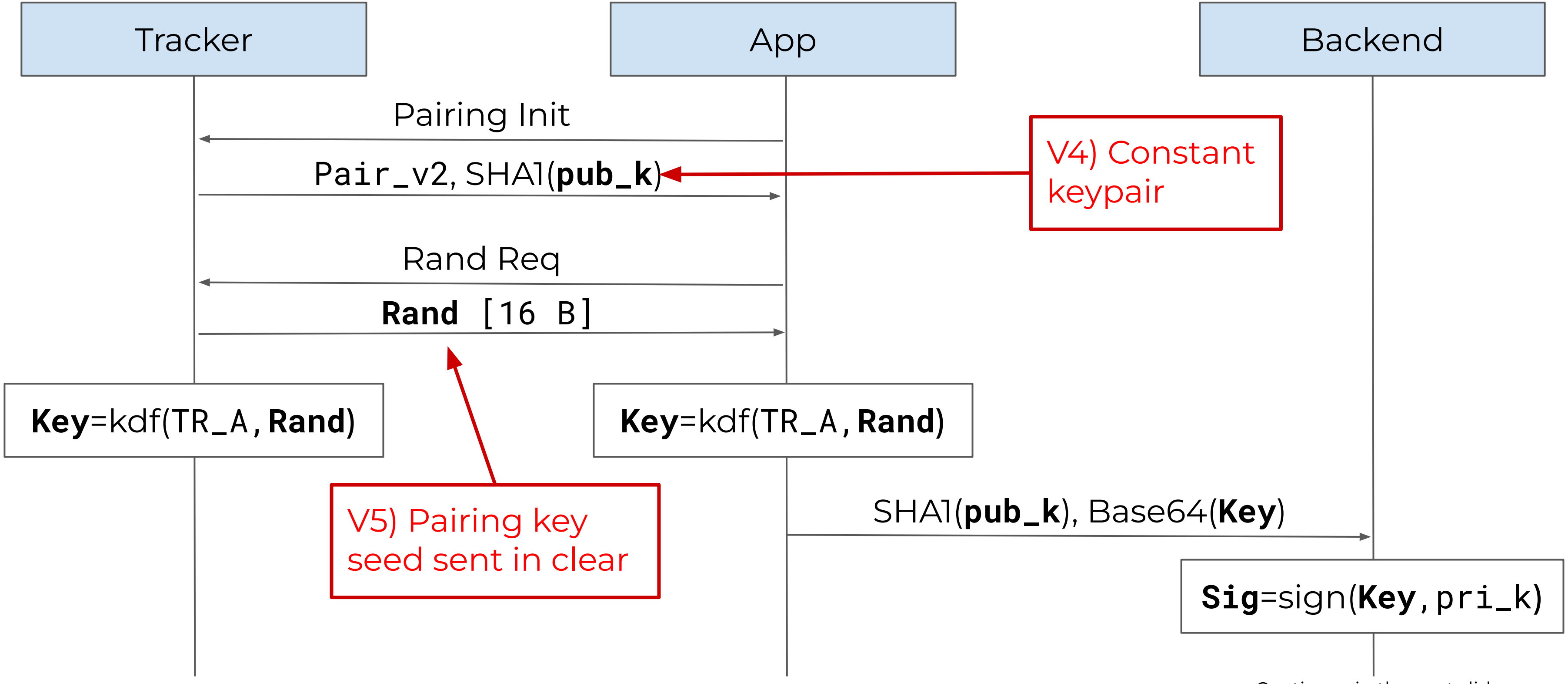
V3) Pairing not authenticated

Xiaomi Pairing v2


Tracker
App
Backend
Pairing Init
Pair_v2, SHA1(**pub_k**)
Rand Req
Rand [16 B]
Key=kdf(TR_A, **Rand**)
Key=kdf(TR_A, **Rand**)
SHA1(**pub_k**), Base64(**Key**)
Sig=sign(**Key**, pri_k)

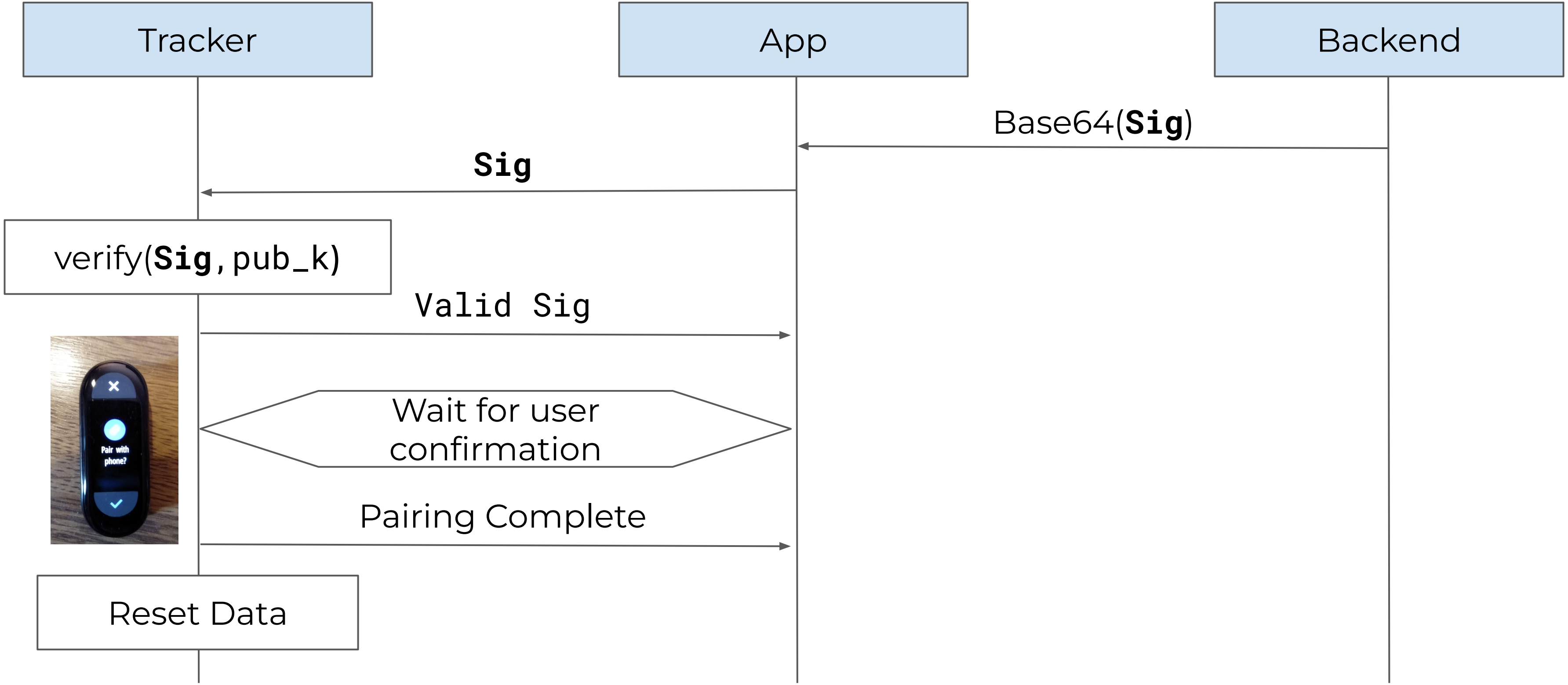
Continues in the next slide

Xiaomi Pairing v2


Tracker
App
Backend
Pairing Init
Pair_v2, SHA1(**pub_k**)
Rand Req
Rand [16 B]
Key=kdf(TR_A, **Rand**)
Key=kdf(TR_A, **Rand**)
SHA1(**pub_k**), Base64(**Key**)
Sig=sign(**Key**, pri_k)
V4) Constant keypair
V5) Pairing key seed sent in clear

Continues in the next slide

Xiaomi Pairing v2 (cont)



Tracker

App

Backend

Base64(**Sig**)

Sig

verify(**Sig**, pub_k)

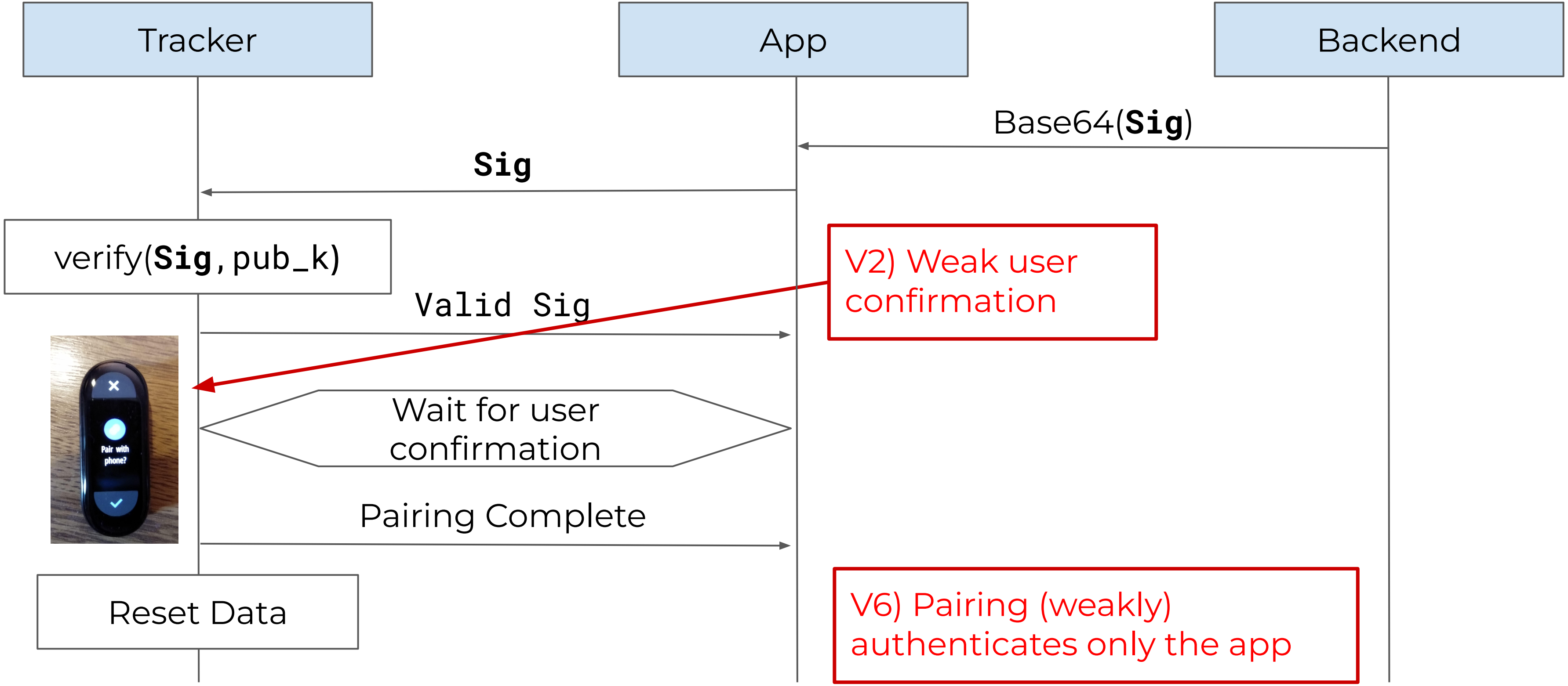
Valid Sig

Wait for user confirmation

Pairing Complete

Reset Data

Xiaomi Pairing v2 (cont)



Tracker

App

Backend

Base64(**Sig**)

Sig

verify(**Sig**,pub_k)

Valid Sig

V2) Weak user confirmation

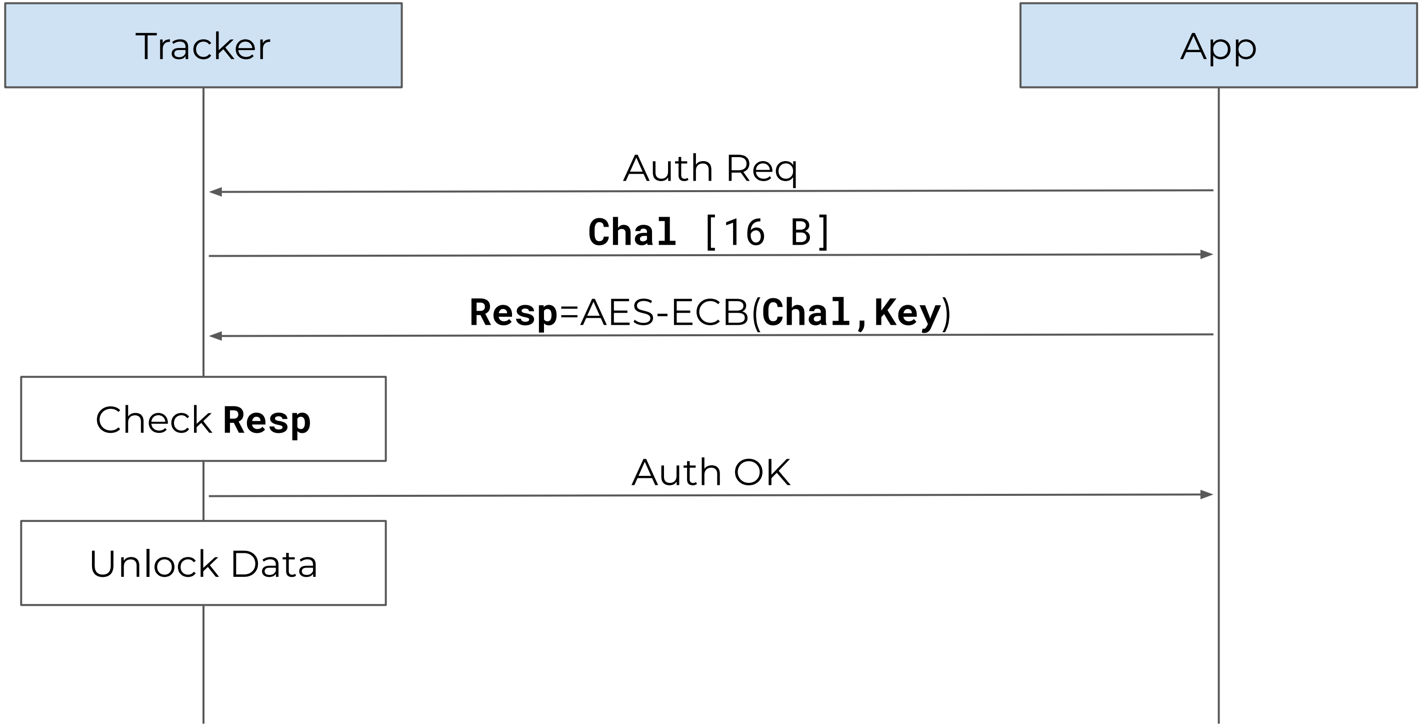
Wait for user confirmation

Pairing Complete

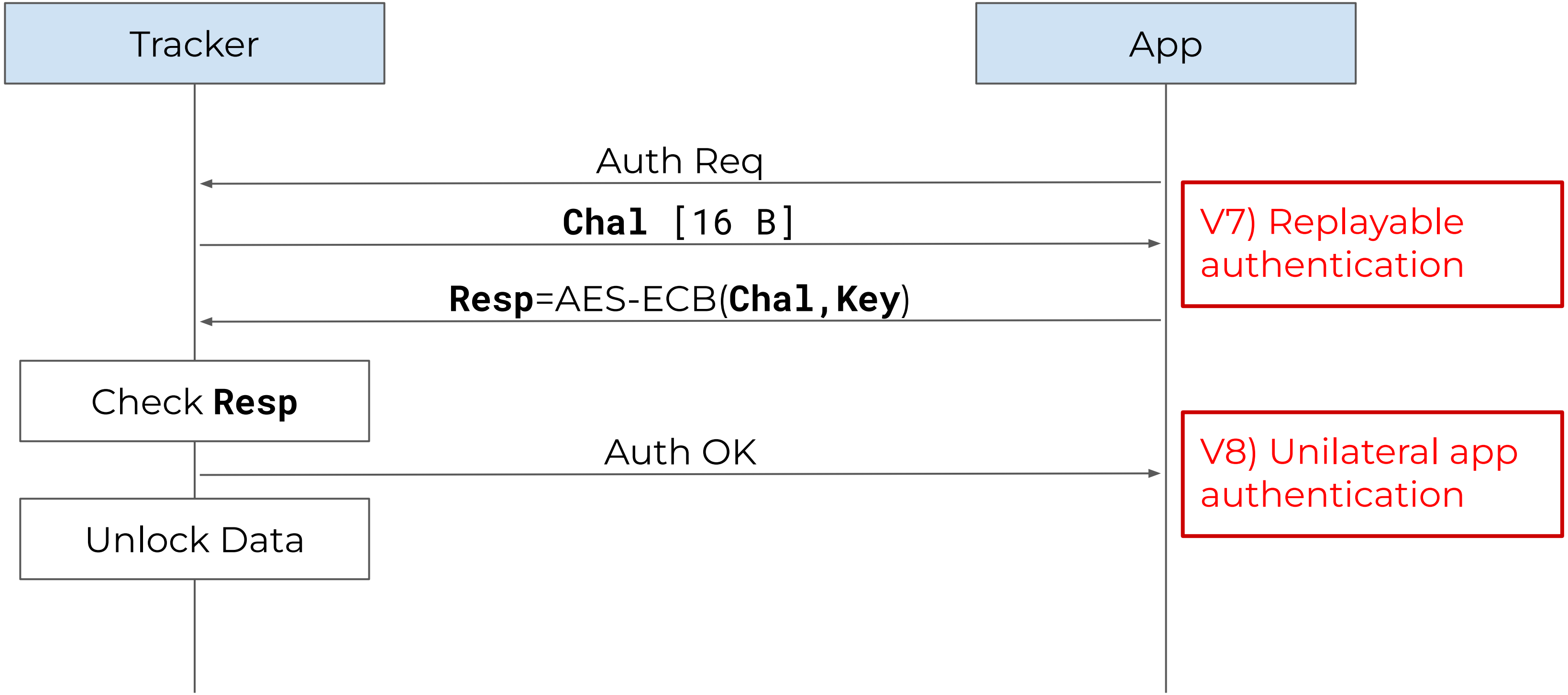
Reset Data

V6) Pairing (weakly) authenticates only the app

Xiaomi Authentication



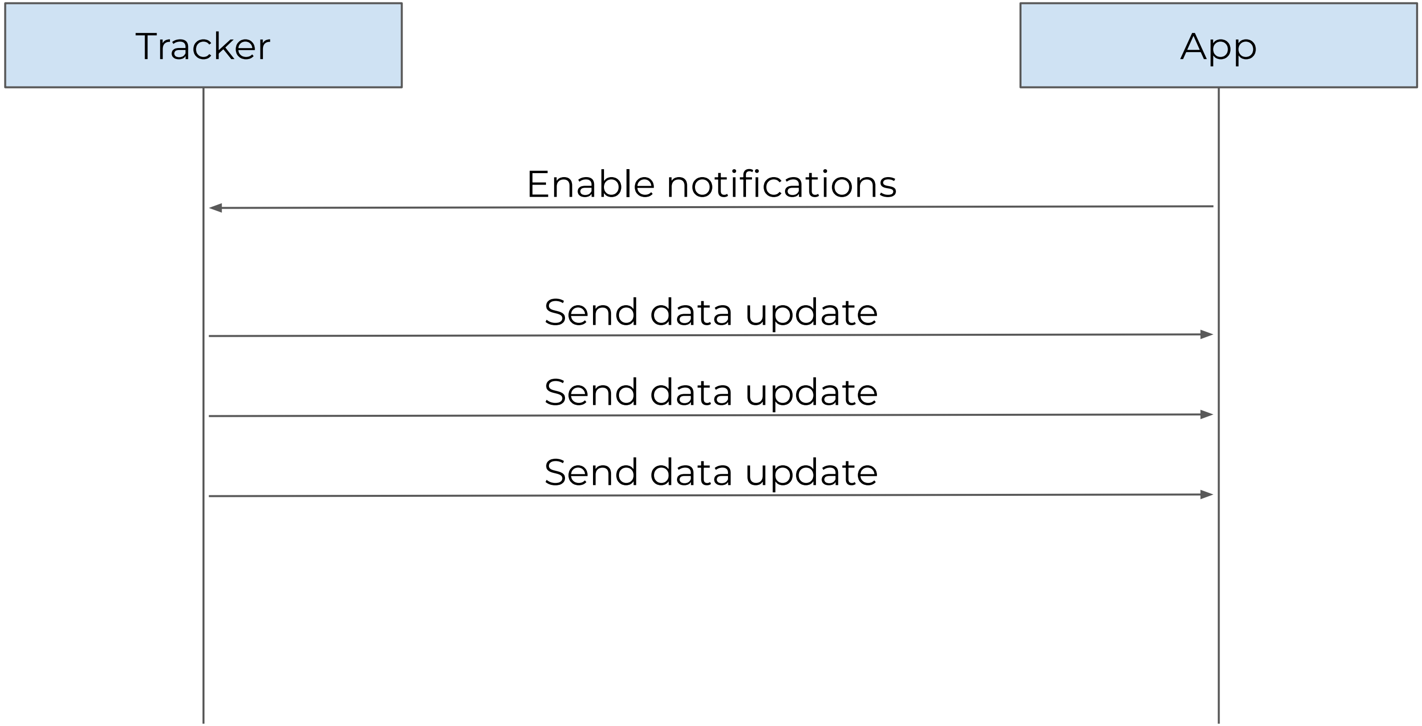
Xiaomi Authentication



V7) Replayable authentication

V8) Unilateral app authentication

Xiaomi Communication



Tracker

App

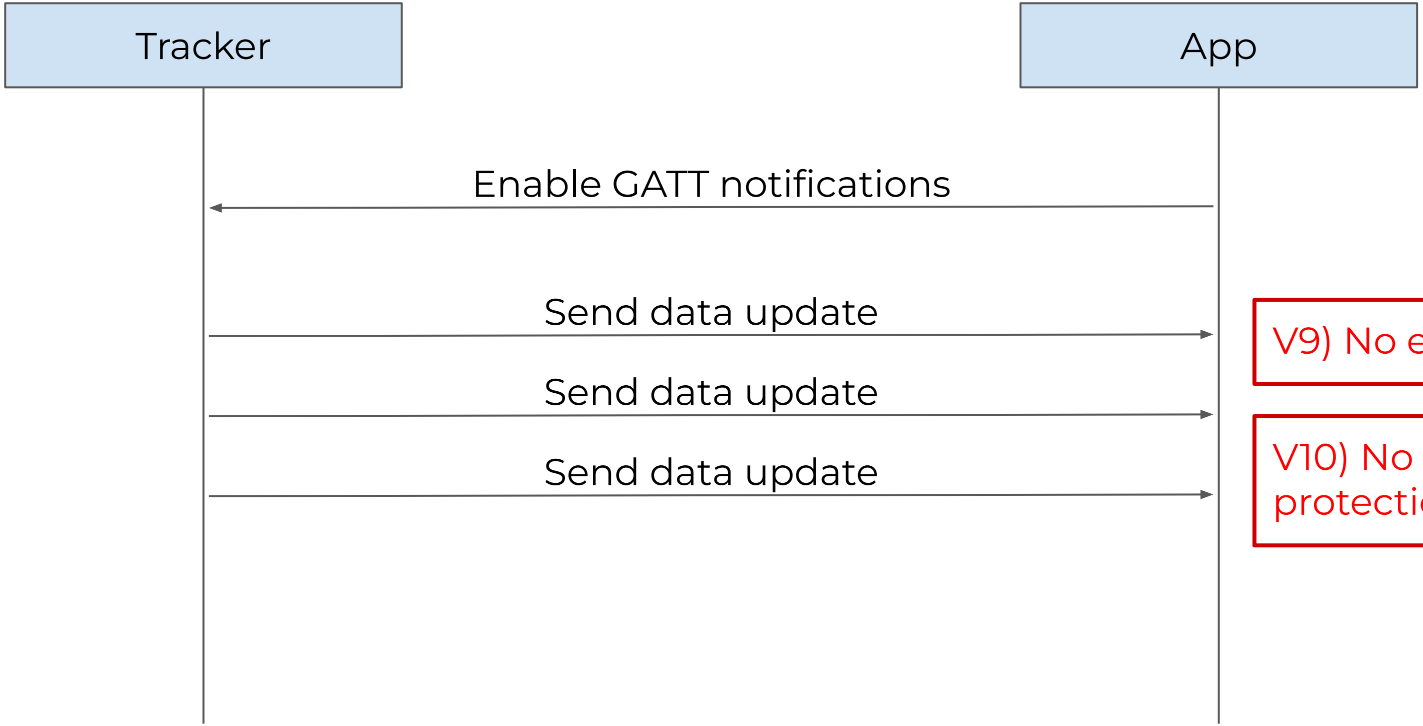
Enable notifications

Send data update

Send data update

Send data update

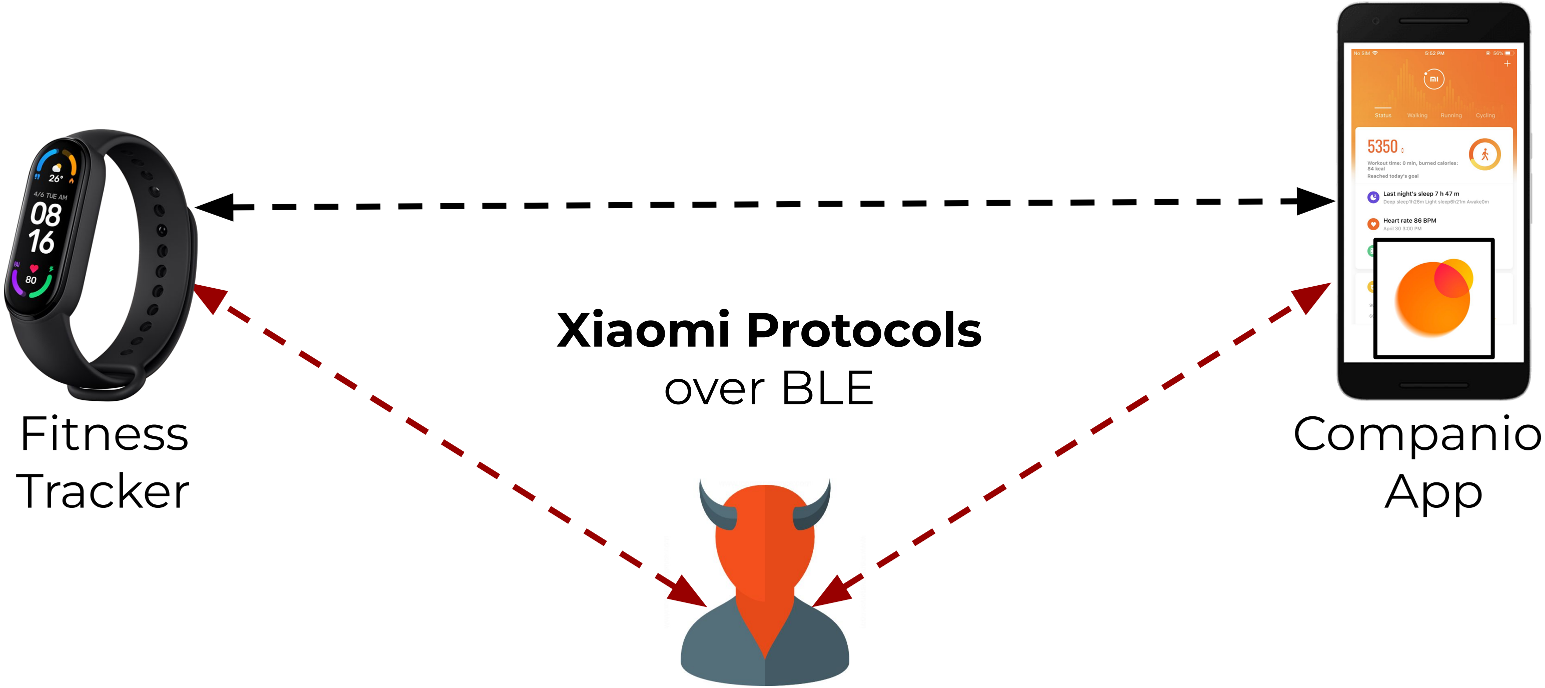
Xiaomi Communication



V9) No encryption

V10) No integrity protection

Proximity Attacker and Attacks



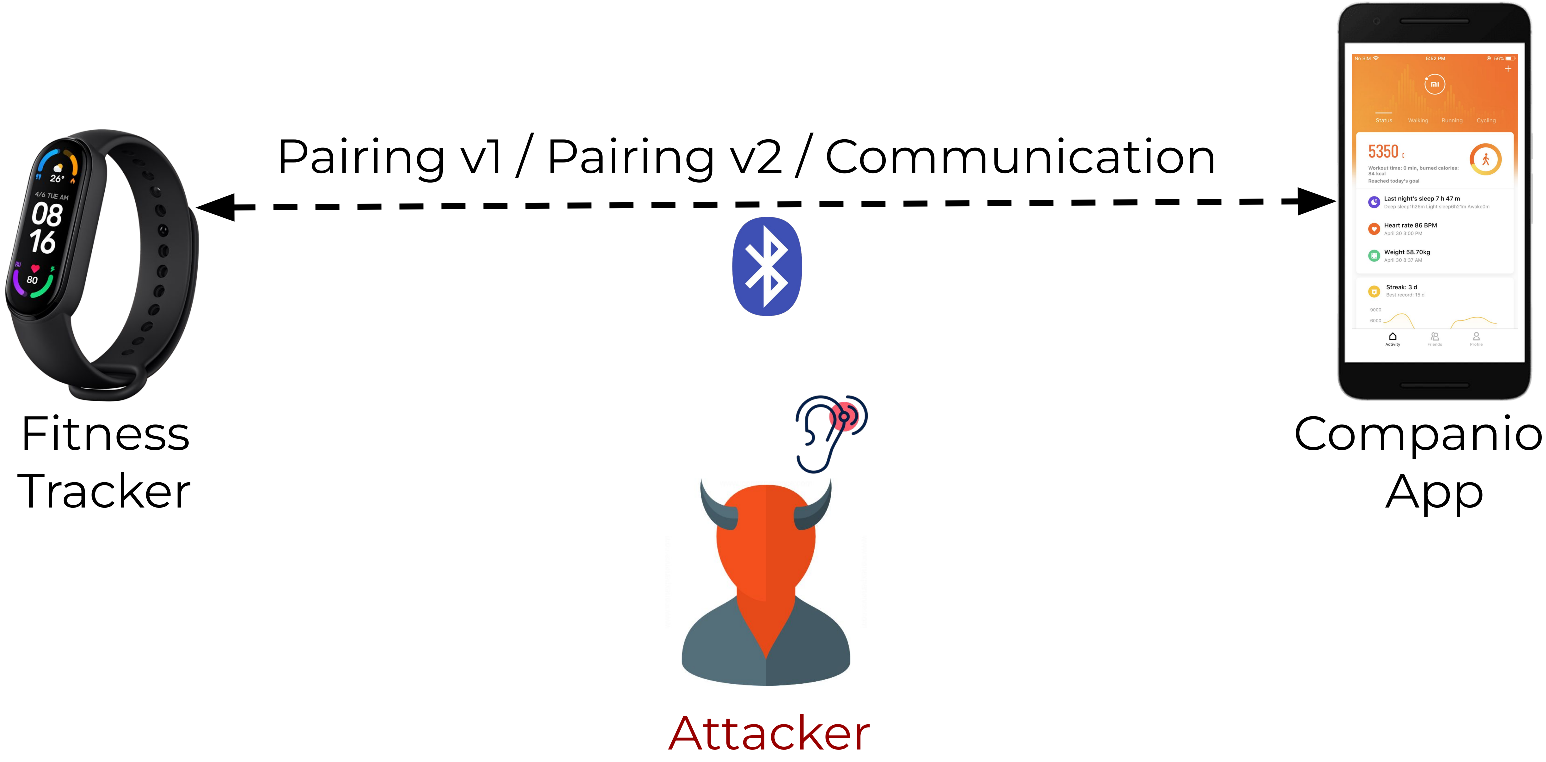
Fitness Tracker

Xiaomi Protocols
over BLE

Companion App

Proximity Attacker

Proximity Eavesdropping



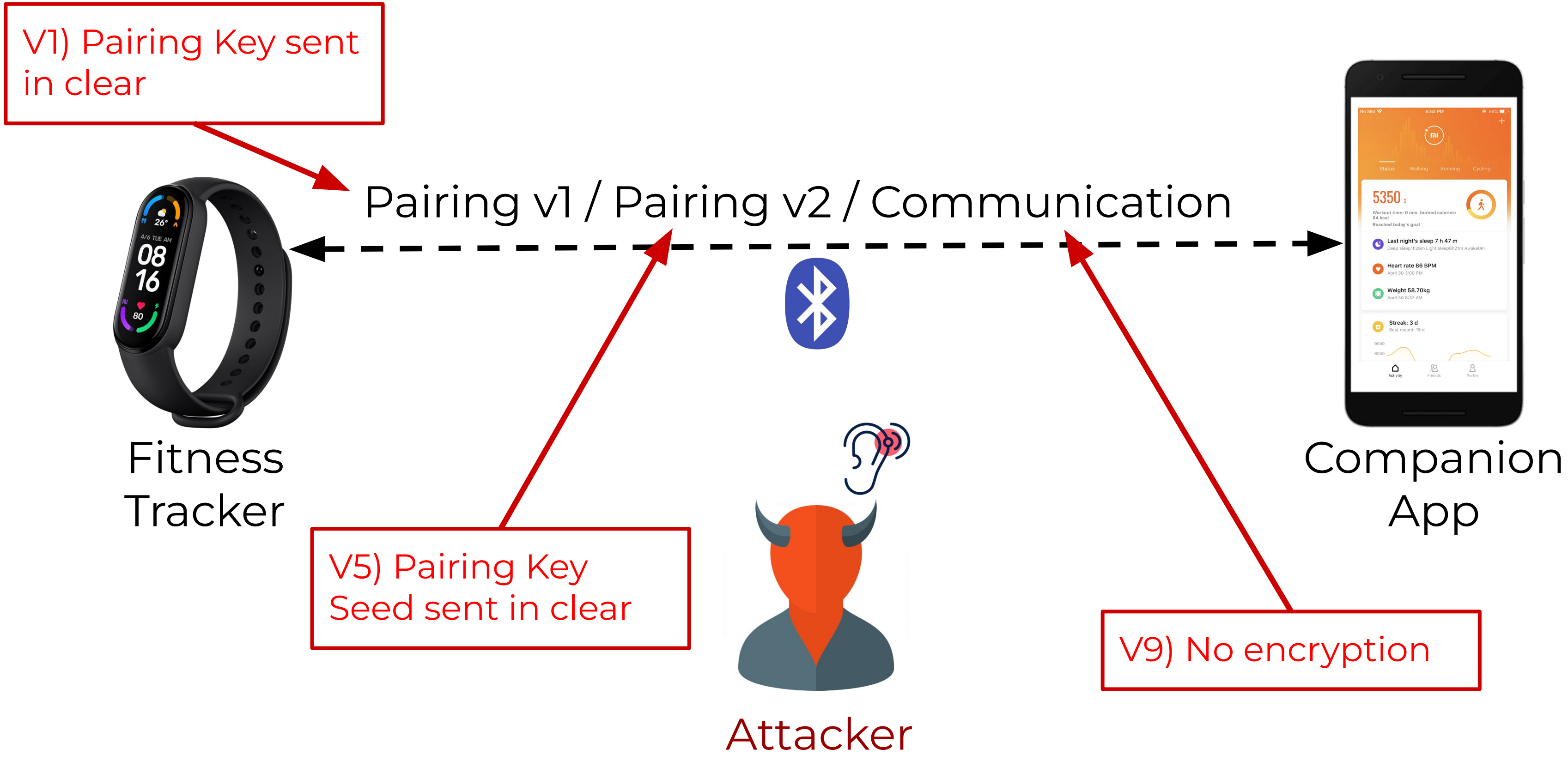
Pairing v1 / Pairing v2 / Communication

Fitness Tracker

Companion App

Attacker

Proximity Eavesdropping



V1) Pairing Key sent in clear

Pairing v1 / Pairing v2 / Communication

Fitness Tracker

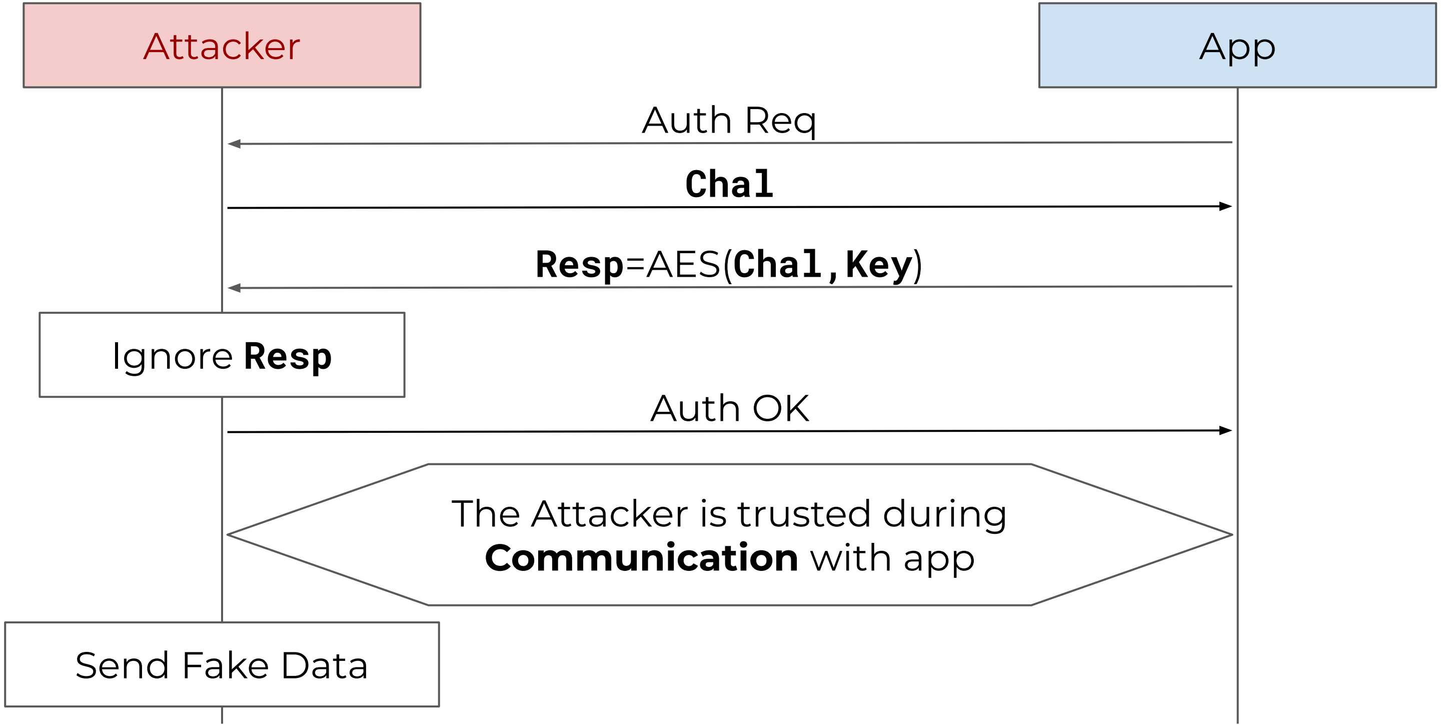
Companion App

V5) Pairing Key Seed sent in clear

V9) No encryption

Attacker

Proximity Tracker Impersonation



Attacker

App

Auth Req

Chal

Resp=AES(**Chal**,**Key**)

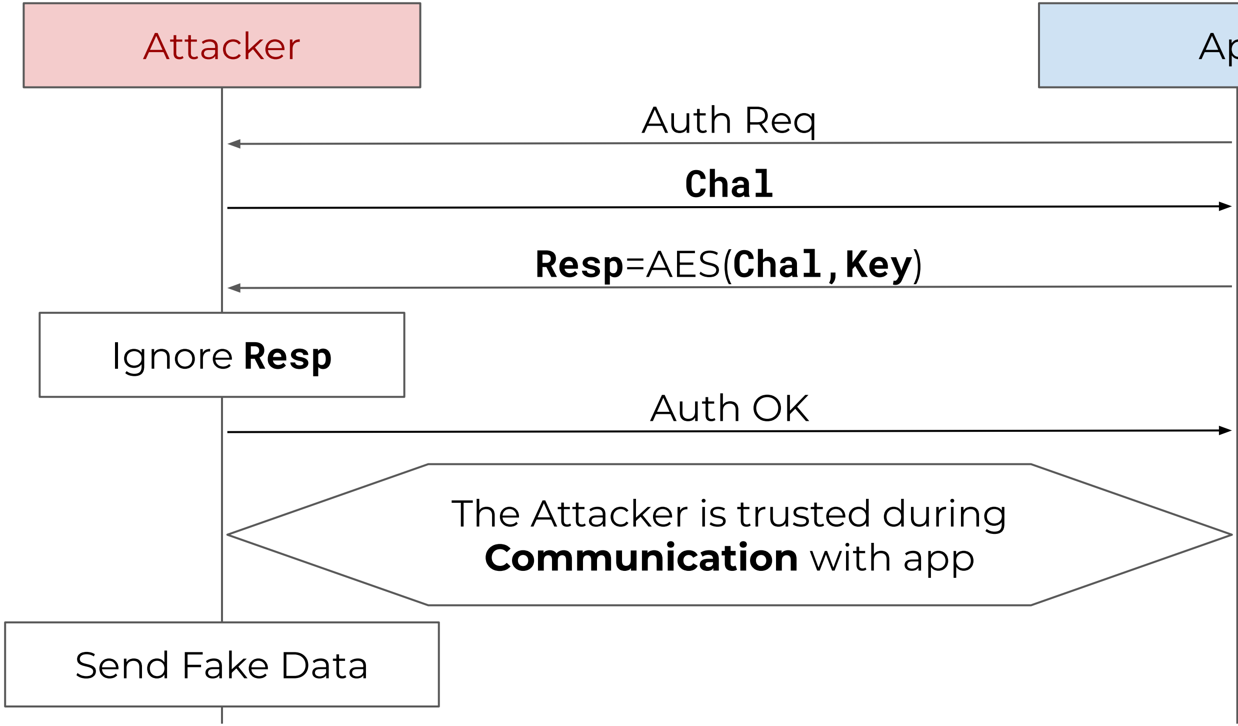
Ignore **Resp**

Auth OK

The Attacker is trusted during **Communication** with app

Send Fake Data

Proximity Tracker Impersonation

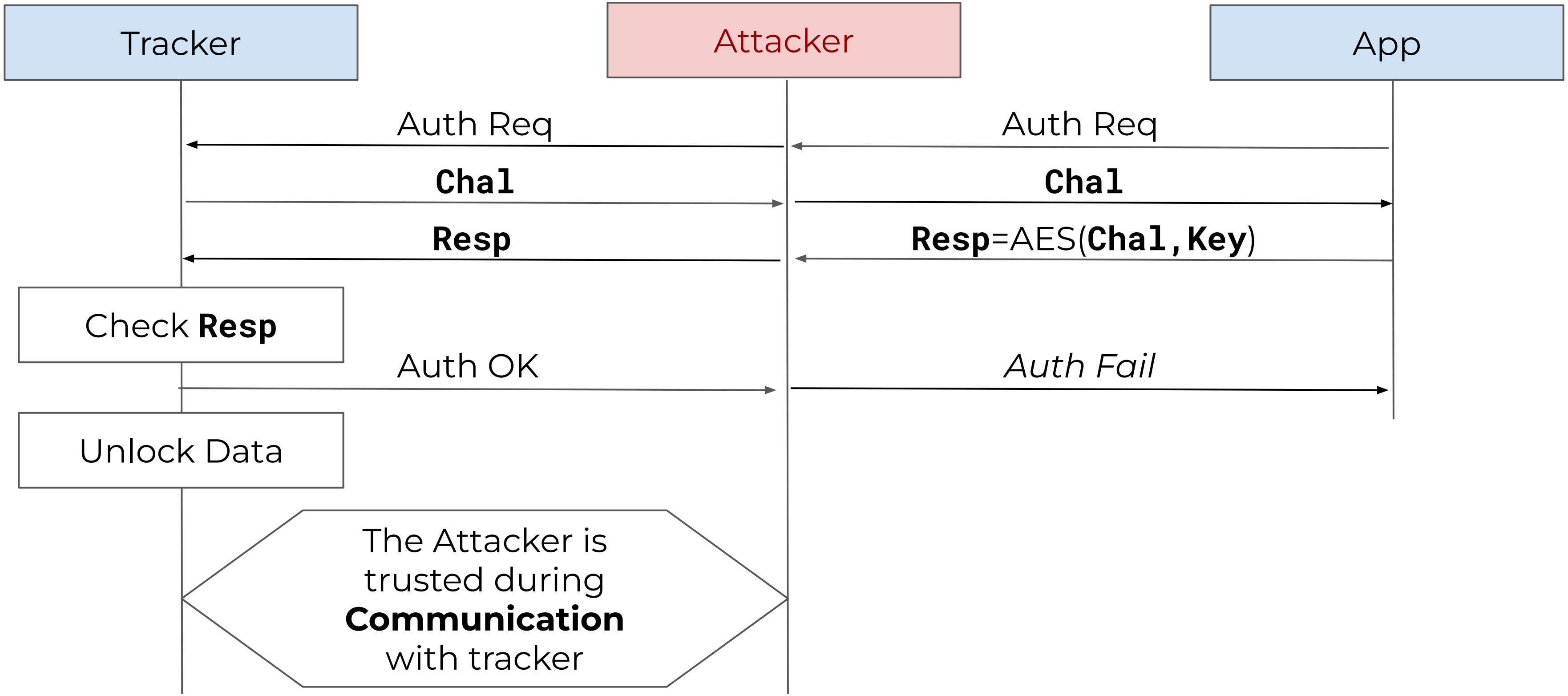

Attacker
App
Auth Req
Chal
Resp=AES(**Chal,Key**)
Ignore **Resp**
Auth OK
The Attacker is trusted during **Communication** with app
Send Fake Data

V8) Unilateral app authentication

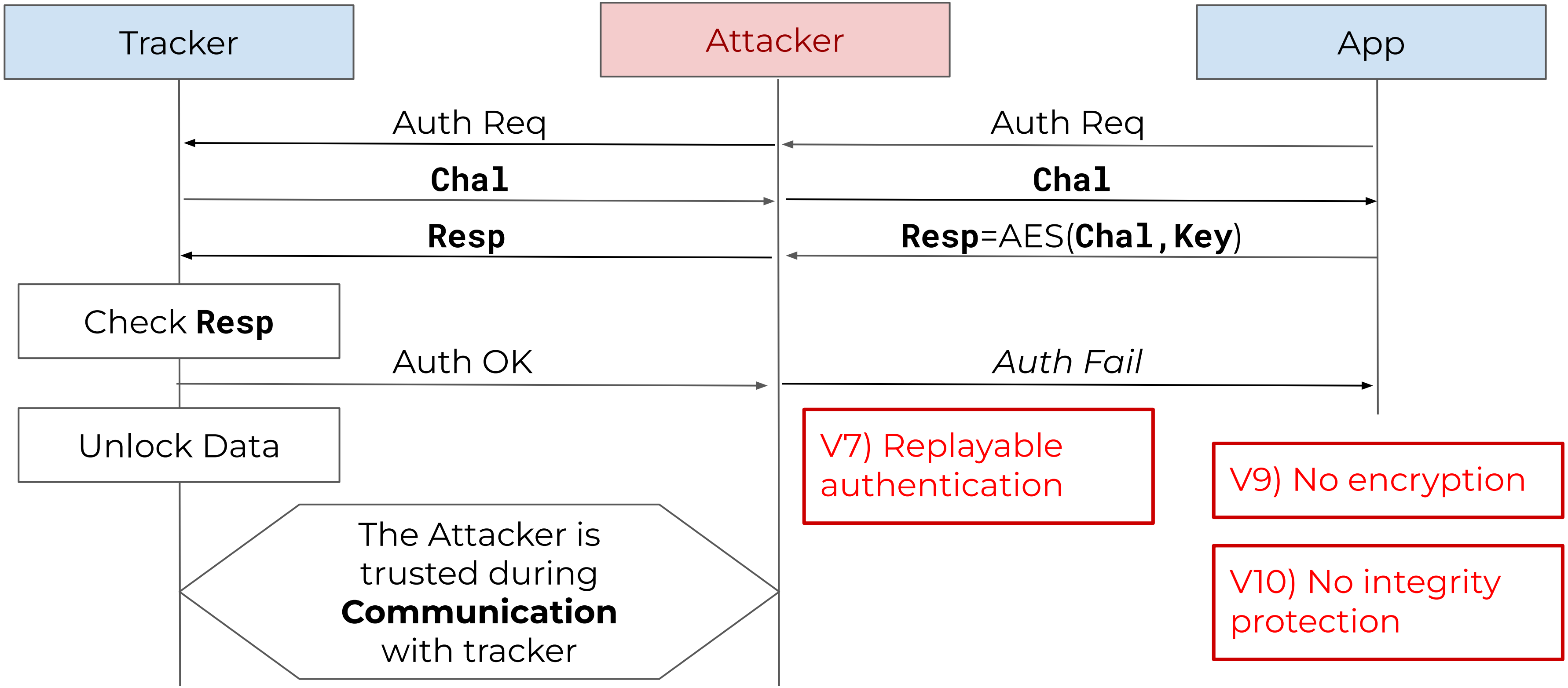
V9) No encryption

V10) No integrity protection

Proximity App Impersonation



Proximity App Impersonation

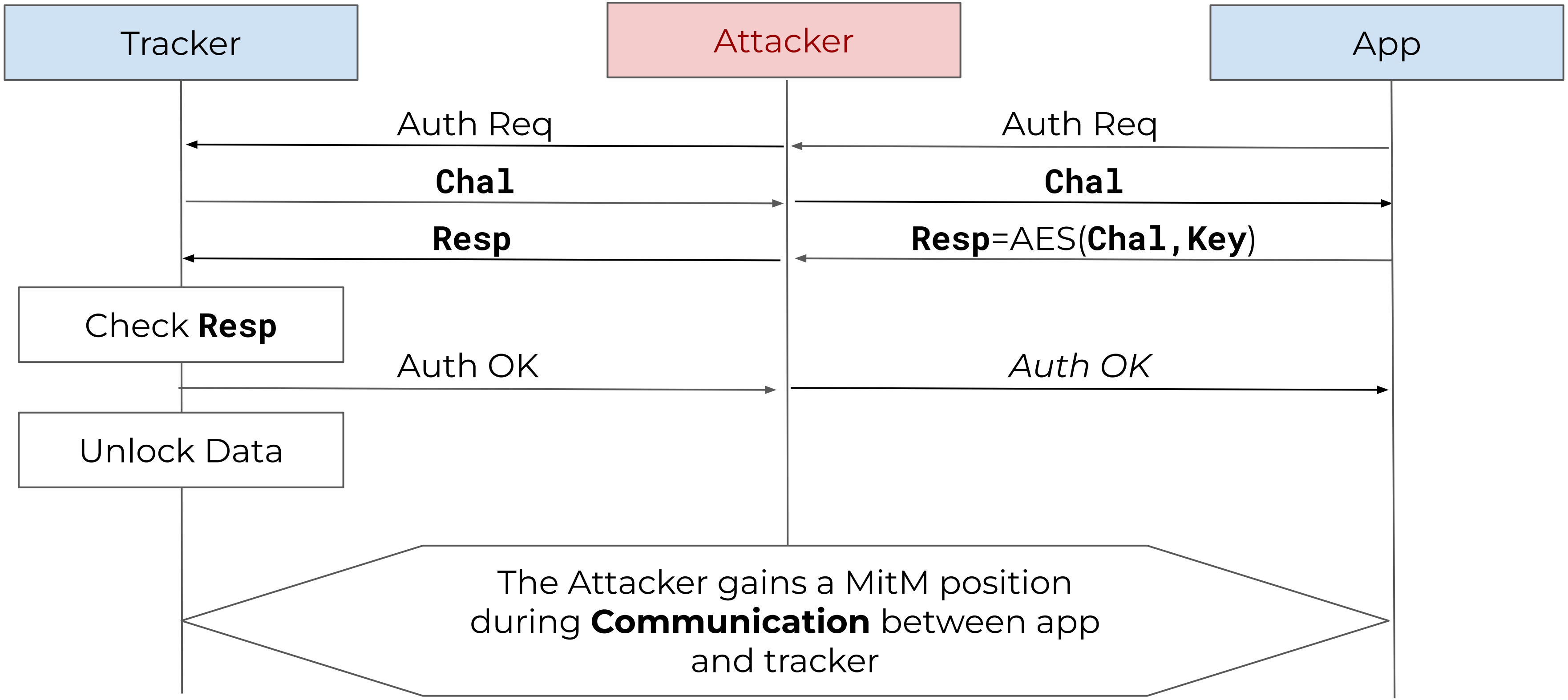


V7) Replayable authentication

V9) No encryption

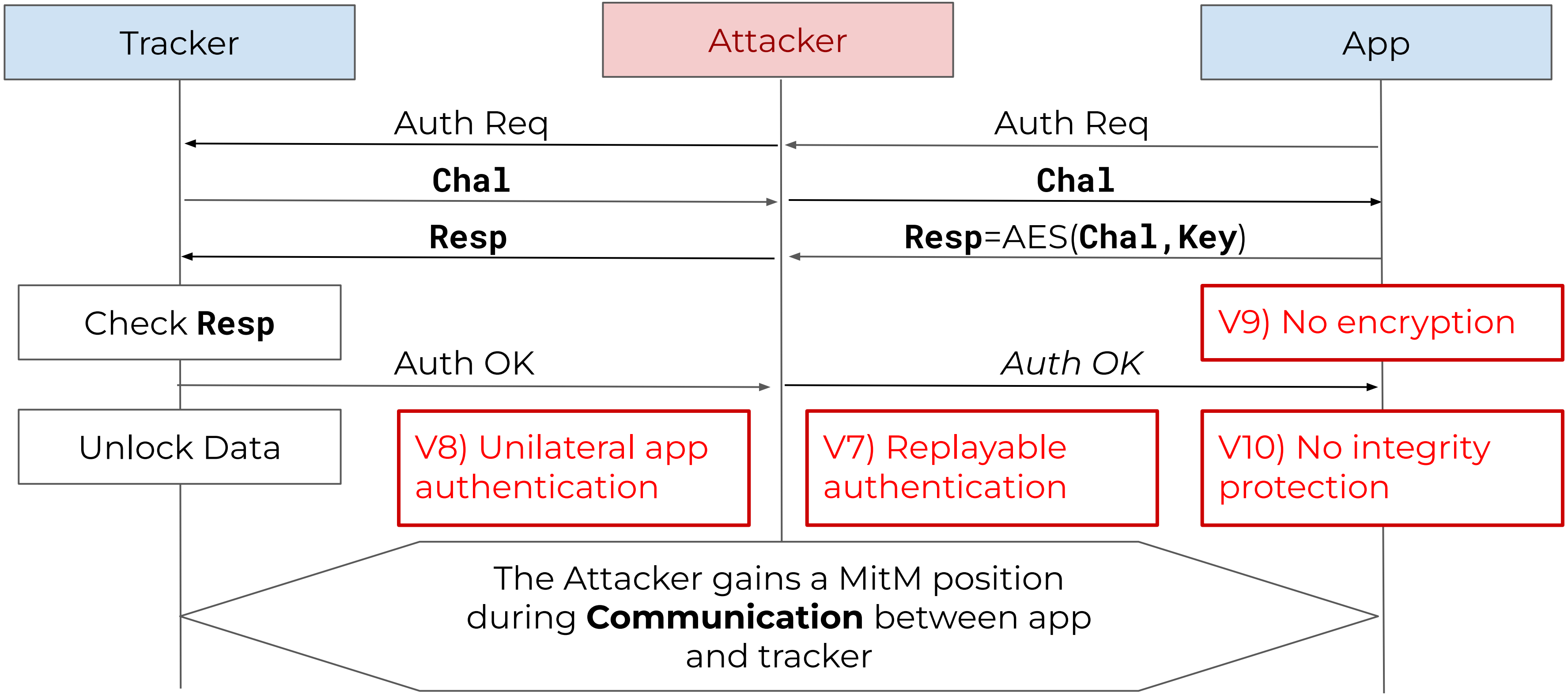
V10) No integrity protection

Proximity Man-in-the-Middle



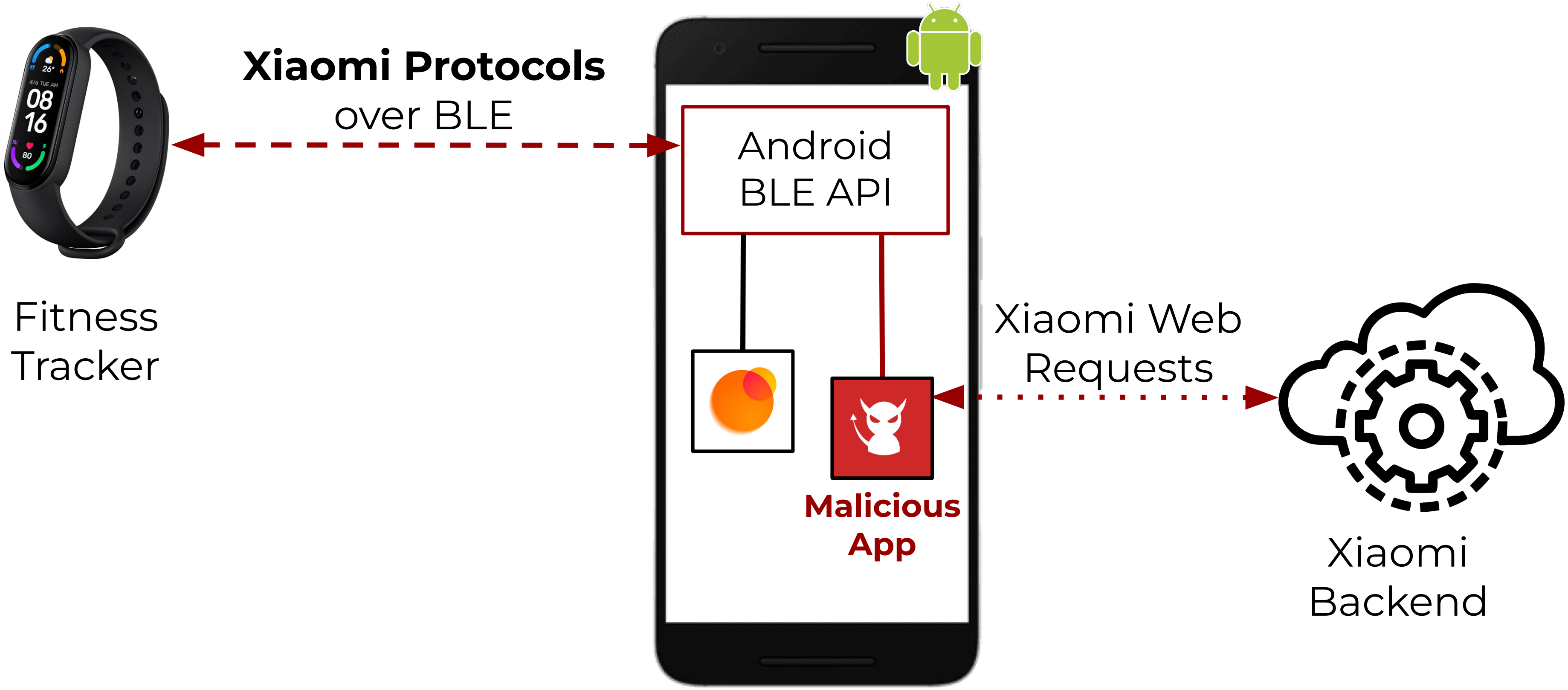
The Attacker gains a MitM position during **Communication** between app and tracker

Proximity Man-in-the-Middle

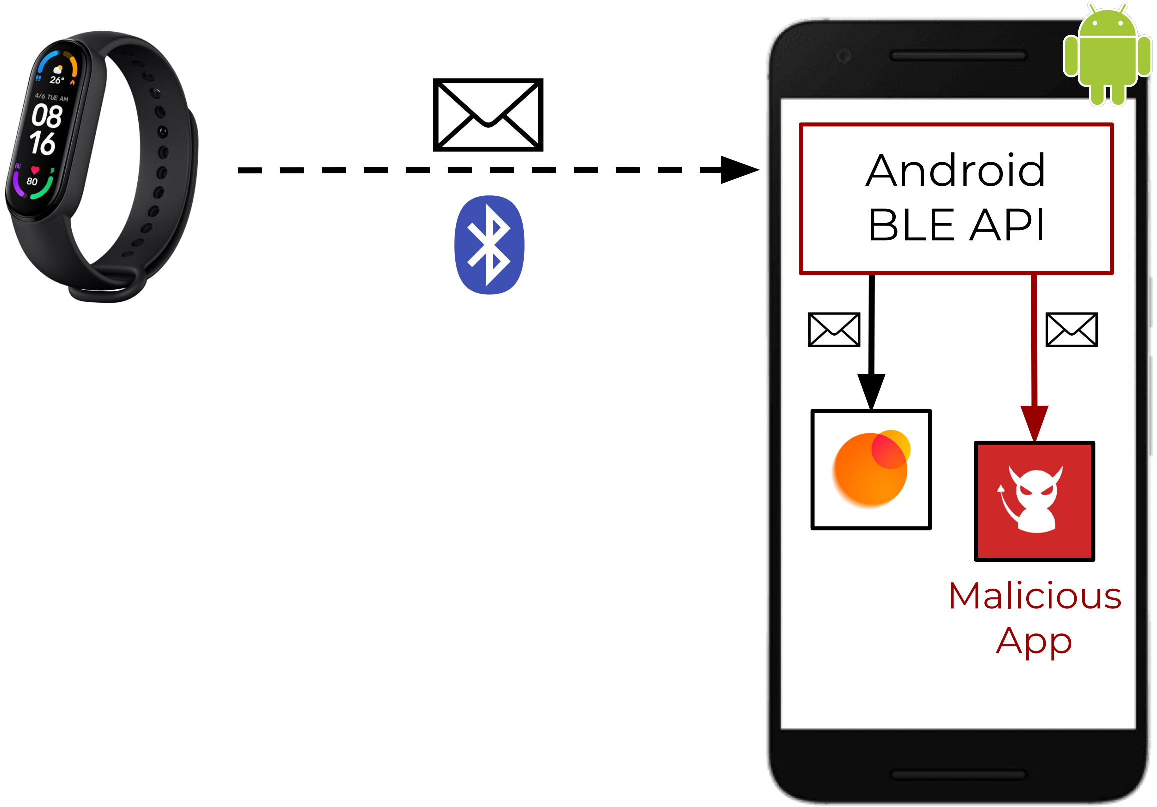


Tracker
Attacker
App
Auth Req
Auth Req
Chal
Chal
Resp
Resp=AES(Chal,Key)
Check Resp
V9) No encryption
Auth OK
Auth OK
Unlock Data
V8) Unilateral app authentication
V7) Replayable authentication
V10) No integrity protection
The Attacker gains a MitM position during **Communication** between app and tracker

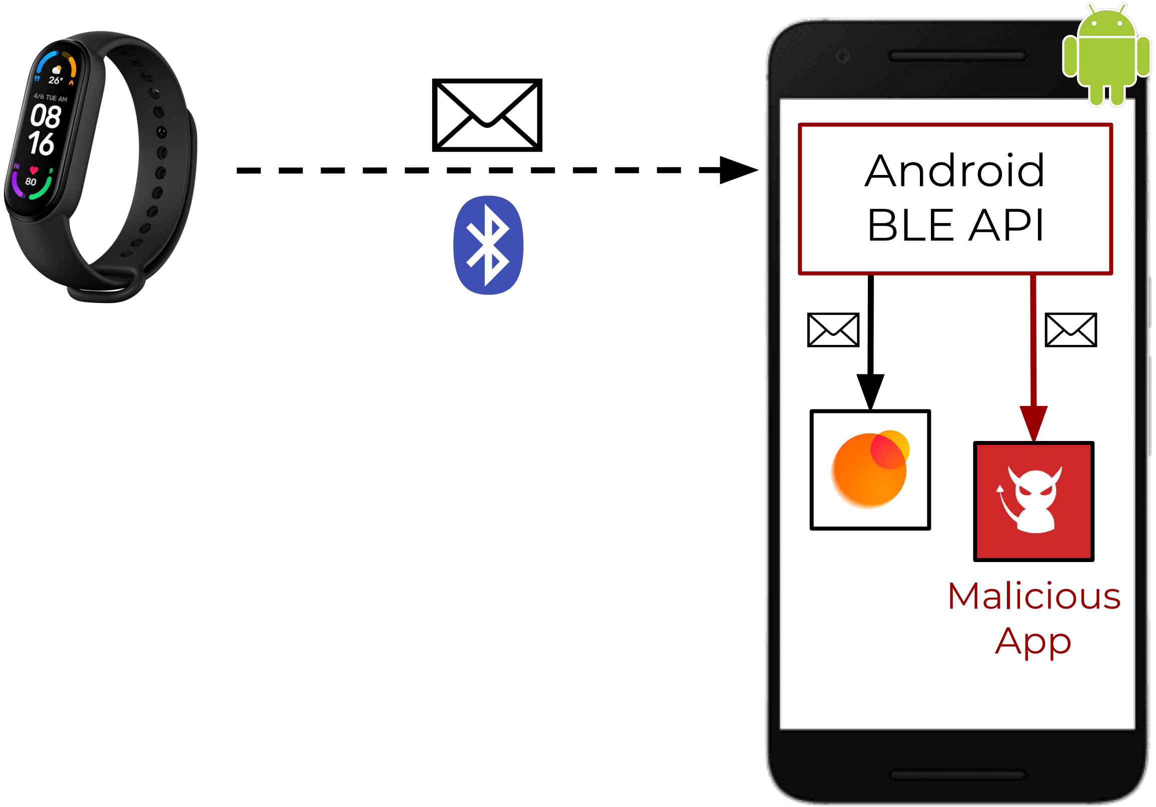
Remote Attacker and Attacks



Remote Eavesdropping



Remote Eavesdropping

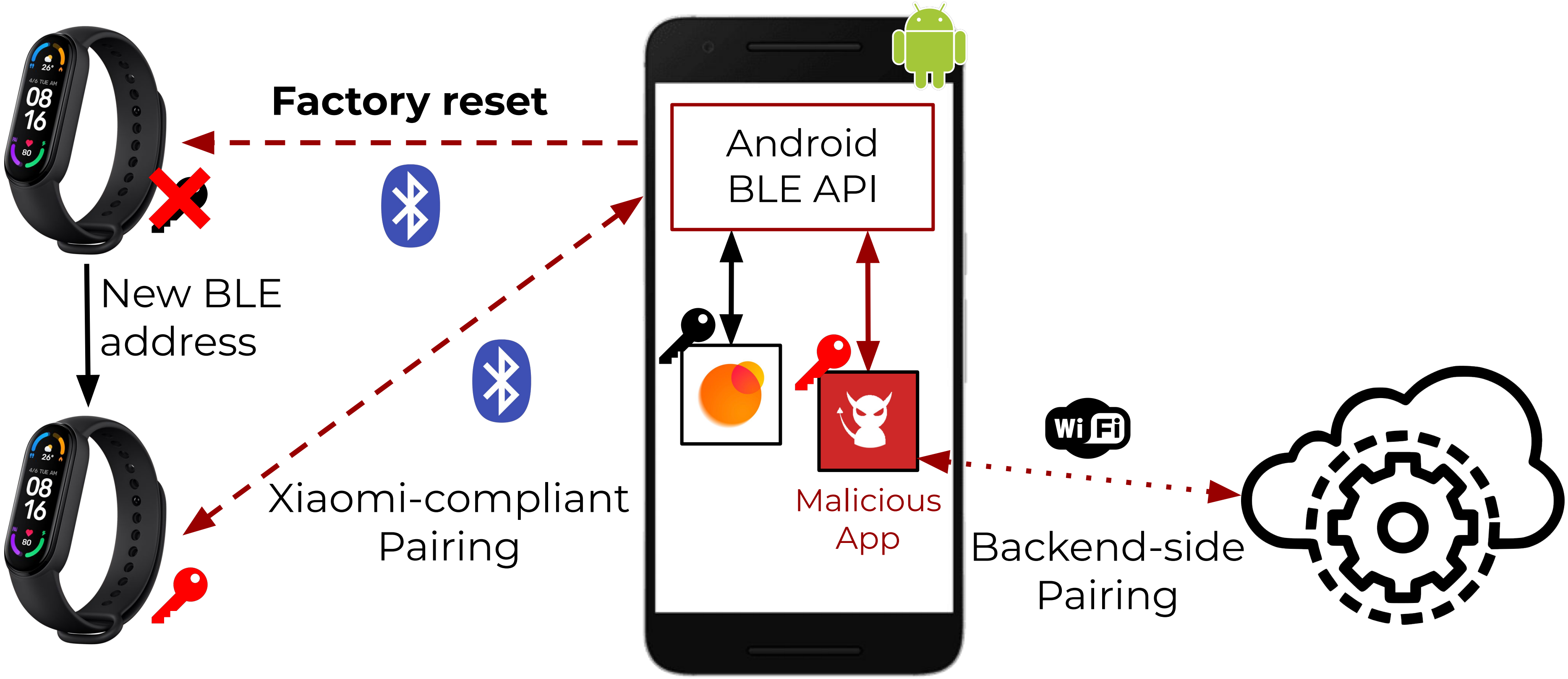


V1) Pairing Key sent in clear

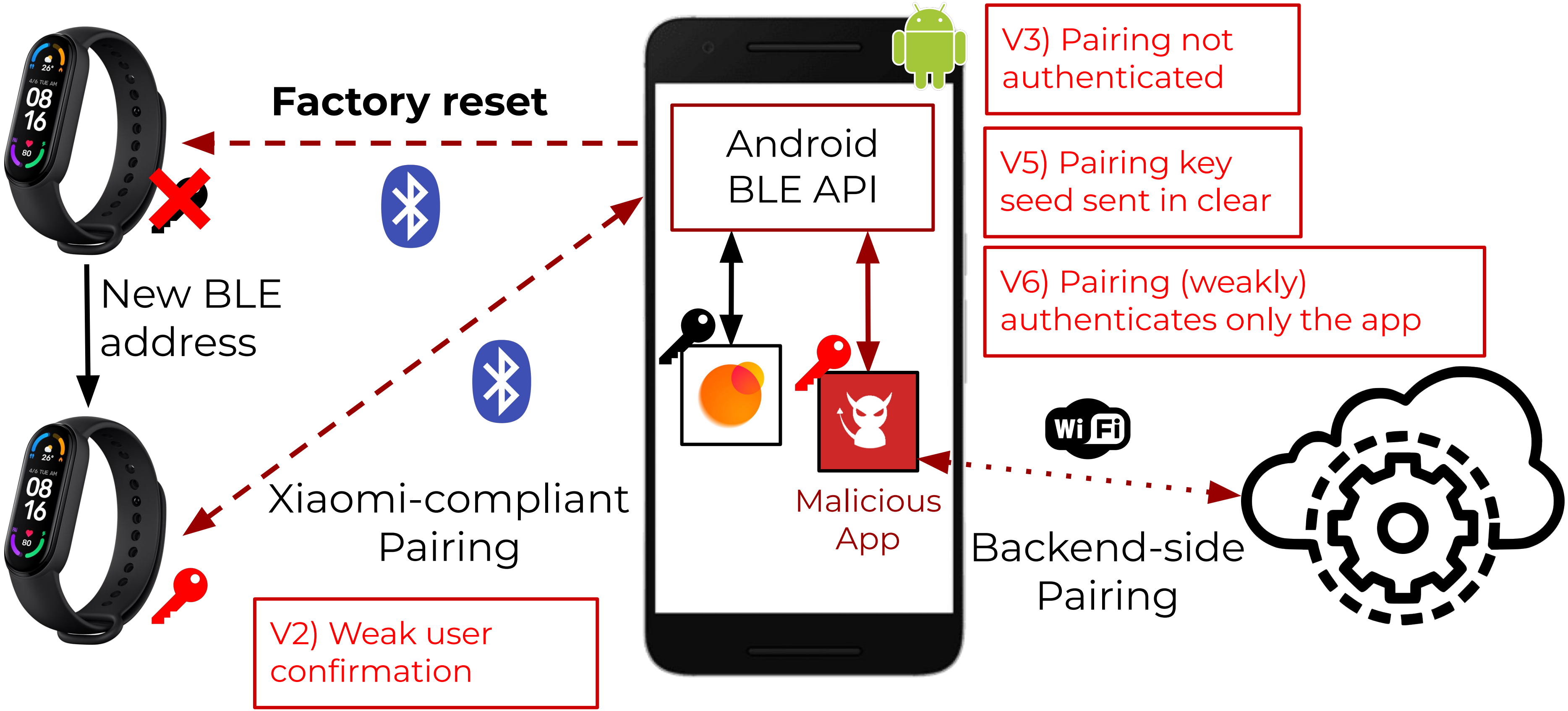
V5) Pairing Key Seed sent in clear

V9) No encryption

Remote App Impersonation



Remote App Impersonation



Factory reset

New BLE address

Xiaomi-compliant Pairing

Android BLE API

Malicious App

Backend-side Pairing

V3) Pairing not authenticated

V5) Pairing key seed sent in clear

V6) Pairing (weakly) authenticates only the app

V2) Weak user confirmation

Evaluation Setup (Trackers)

Tracker	Release Year	Pairing Version	Bluetooth Version	LE Secure Conn.	Link-Layer Security
Mi Band 2	2016	1	4.2	X	✔
Mi Band 3	2018	1	4.2	X	✔
Cor 2	2019	1	4.2	X	✔
Mi Band 4	2019	2	5.0	✔	✔
Mi Band 5	2020	2	5.0	✔	✔
Mi Band 6	2021	2	5.0	✔	✔

Evaluation Setup (Companion Apps)

App	App Version	Year	OS
Zepp Life (formerly Mi Fit)	4.8.1	2020	Android
Zepp (formerly Amazfit)	5.9.2	2021	Android

Evaluation Results

	Proximity Attacks				Remote Attacks	
	Trac Imp.	App Imp.	MitM	Eavesdr.	App Imp.	Eavesdr.
Zepp Life app	n/a	✔	✔	✔	✔	n/a
Zepp app	n/a	✔	✔	✔	✔	n/a
Mi Band 2	✔	n/a	✔	✔	n/a	✔
Mi Band 3	✔	n/a	✔	✔	n/a	✔
Amazfit Cor 2	✔	n/a	✔	✔	n/a	✔
Mi Band 4	✔	n/a	✔	✔	n/a	✔
Mi Band 5	✔	n/a	✔	✔	n/a	✔
Mi Band 6	✔	n/a	✔	✔	n/a	✔

Evaluation Results (Android Versions)

Smartphone	Android Version	Remote Attacks	
		Eavesdropping	App Impersonation
Pixel 4A	12 (23.58%)	*	*
Pixel 2XL	11 (27.96%)	✔	✔
Pixel 2XL	10 (20.98%)	✔	✔
Galaxy J5	9 (10.58%)	✔	✔
Redmi 5 Plus	8 (8.08%)	✔	✔
Galaxy S5	6 (2.25%)	✔	✔

* Requires dangerous runtime permission `BLUETOOTH_CONNECT`

BREAKMI TOOLKIT

BreakMi Toolkit

- BreakMi
  - **Proximity** attacks via NodeJS
  - **Remote** attacks via Android app
  - Xiaomi protocol **dissectors**
  - Frida DBA **hooks** for Zepp and Zepp Life
  - Links to our attacks demos
  - **Open-source** via BreakMi GitHub repo

Proximity Attacks Implementation

- **Bleno** and **Noble** (NodeJS modules)
  - BLE Peripheral to spoof tracker
  - BLE Central to spoof app
  - Must run Node version 8.9.0 to work (`nvm use 8.9.0`)
  - Recommend to install `@abandonware/{bleno,noble}`

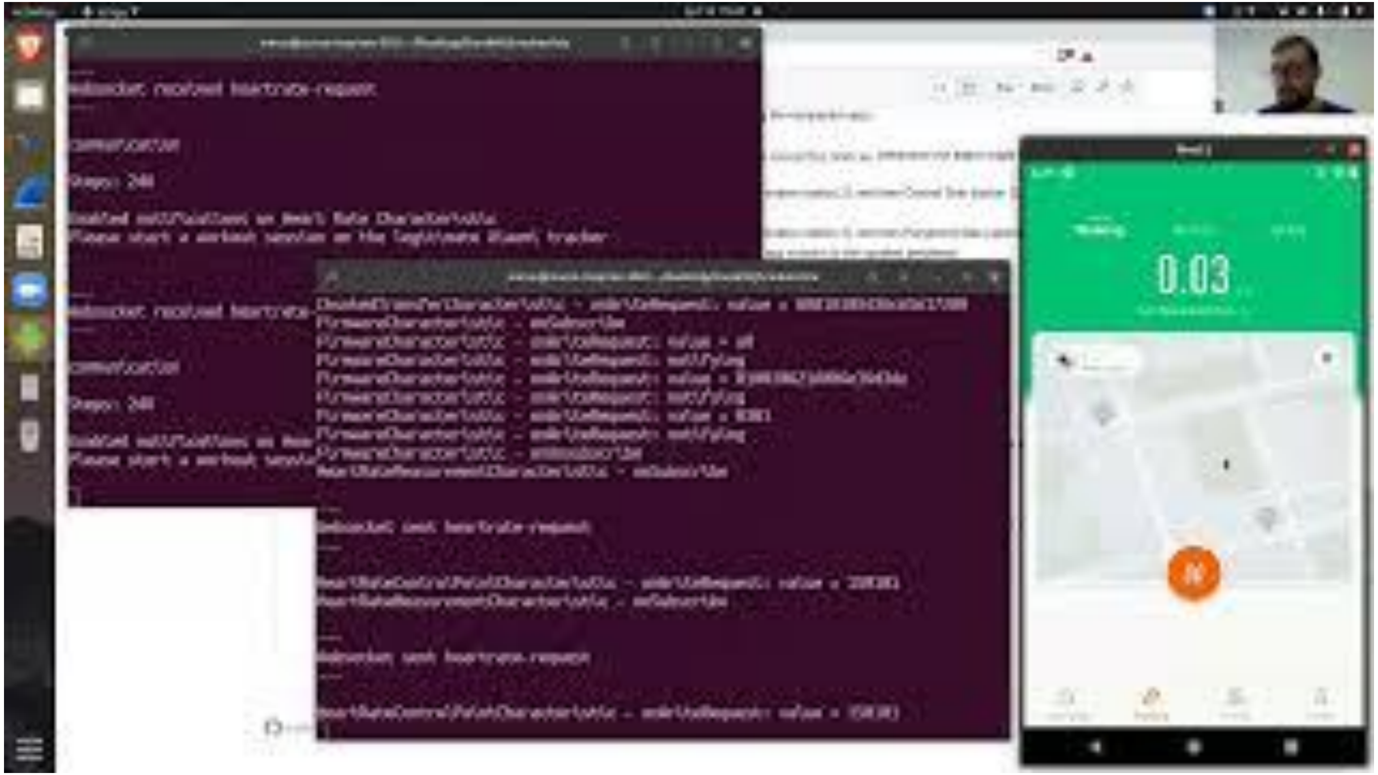
Proximity Impersonation Attacks

- BLE address spoofing
  - Vendor-specific **bdaddr** (CSR8510 A-10 Controller)
- Implement tracker's GATT server
  - E.g., services, characteristics, allowed operations
- Perform service and characteristic discovery
  - Required to send read/write requests to tracker

Proximity Man-in-the-Middle

- Impersonate app and tracker at the **same time**
  - Requires two BLE interfaces
- **Sockets** to forward packets from fake tracker to fake app, and vice versa

Proximity Man-in-the-Middle Demo



```
---
websocket received heartrate-request
---

connection

Steps: 248

Enabled notifications on Heart Rate Characteristic
Please start a workout session on the legitimate Xiaomi tracker


---
websocket received heartrate-request
---

connection

Steps: 248

Enabled notifications on Hear[illegible]
Please start a workout sessio[illegible]
```

```
ChunkedTransferCharacteristic - onWriteRequest: value = [illegible]
FirmwareCharacteristic - onSubscribe
FirmwareCharacteristic - onWriteRequest: value = [illegible]
FirmwareCharacteristic - onWriteRequest: notifying
FirmwareCharacteristic - onWriteRequest: value = [illegible]
FirmwareCharacteristic - onWriteRequest: notifying
FirmwareCharacteristic - onWriteRequest: value = [illegible]
FirmwareCharacteristic - onWriteRequest: notifying
FirmwareCharacteristic - onUnsubscribe
HeartRateMeasurementCharacteristic - onSubscribe

---
websocket sent heartrate-request
---

HeartRateControlPointCharacteristic - onWriteRequest: value = [illegible]
HeartRateMeasurementCharacteristic - onSubscribe

---
websocket sent heartrate-request
---

HeartRateControlPointCharacteristic - onWriteRequest: value = [illegible]
```

Remote Attacks Implementation

- Malicious **Android app** written in Java
  - Exploit Android BLE API
  - All Android apps can read the entire BLE traffic
  - Need for application-layer encryption!

Remote Eavesdropping Demo

FITBIT FT ECOSYSTEM SECURITY EVALUATION

Fitbit FT Ecosystem

- **Similar** ecosystem to Xiaomi
  - Fitness trackers (Charge 2, ...)
  - Companion Android/iOS apps (Fitbit)
  - Backend
- **Proprietary app-layer** protocols over BLE
  - Pairing, Authentication, Communication
- BLE link-layer security is **enabled**
  - Unlike Xiaomi

Fitbit Targets

- **Charge 2** tracker
  - Released in 2014, partially studied
  - Random **static** BLE address
  - Requires different advertising flag when spoofing
- **Fitbit** Android app
  - Backend-side pairing (different from Xiaomi)

Fitbit Proprietary Protocols (1)

- Pairing
  - **Pre-shared** device key (`DK`)
  - Fitbit **backend** generates PK using `Salt` and `DK`
  - App receives PK and `Salt`, used later for Authentication
  - Strong pairing confirmation (**Numeric Comparison**)

Fitbit Proprietary Protocols (2)

- Authentication
  - **Mutual** authentication
  - Use of `Salt`, random `chals`, and a packet counter
  - **MAC** integrity protection
- Communication
  - **Real-time** mode
  - Normal mode that synchronizes with backend

Fitbit Security Highlights

- Stronger security than Xiaomi
  - Mutual authentication
  - Strong pairing confirmation
- Nonetheless, shares many critical vulnerabilities
  - **No** pairing authentication
  - Authentication is **replayable**
  - **Unencrypted** real-time mode communication

Fitbit Proximity App Impersonation Demo

Fitbit Evaluation Results

	Proximity Attacks				Remote Attacks	
	Trac Imp.	App Imp.	MitM	Eavesdr.	App Imp.	Eavesdr.
Fitbit app	n/a	✔	✔	†*	✔	n/a
Fitbit Charge 2	X	n/a	✔	†*	n/a	*

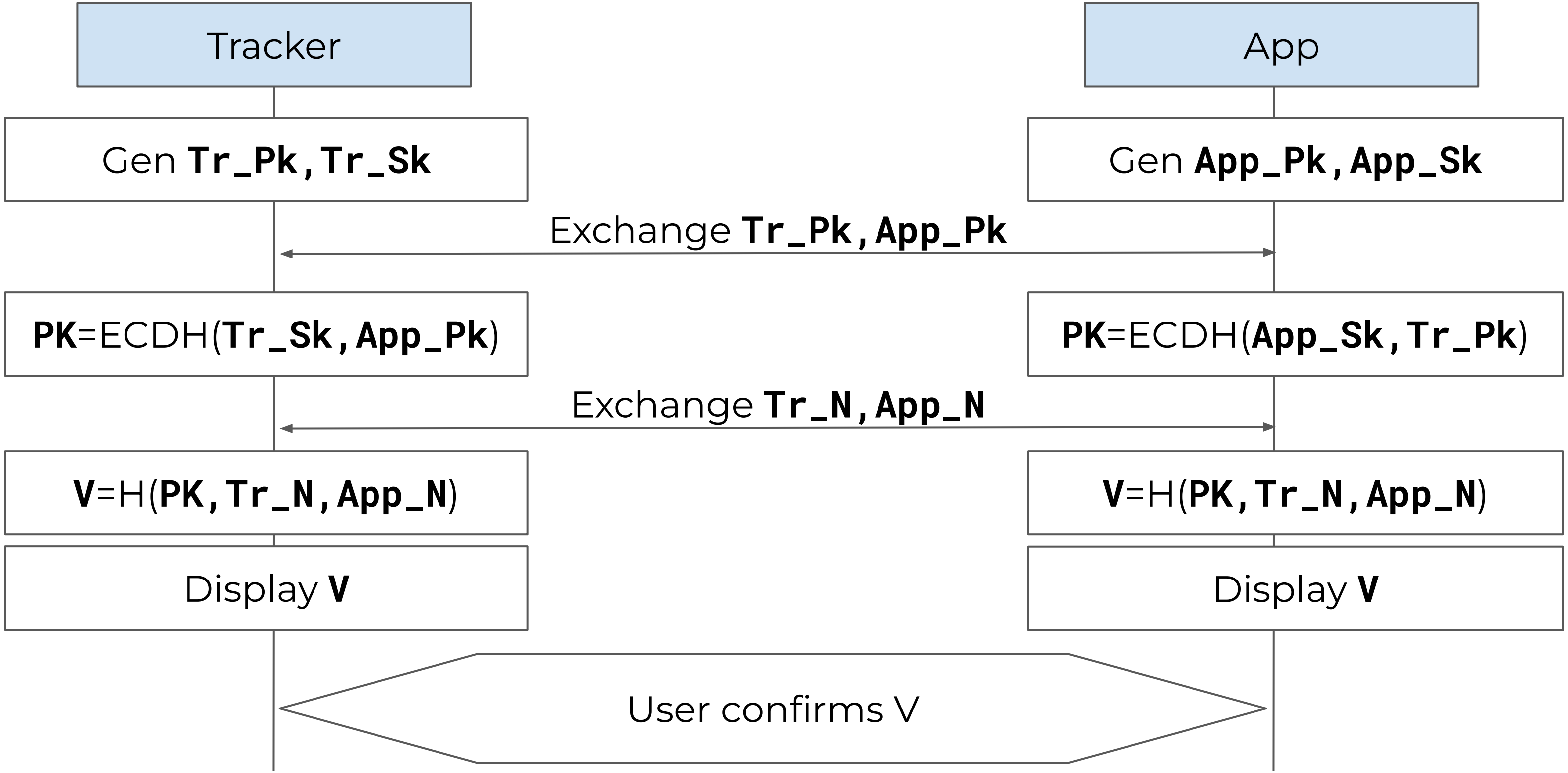
* Only works for real-time unencrypted mode
† Needs link-layer security breach

COUNTERMEASURES AND DISCLOSURE

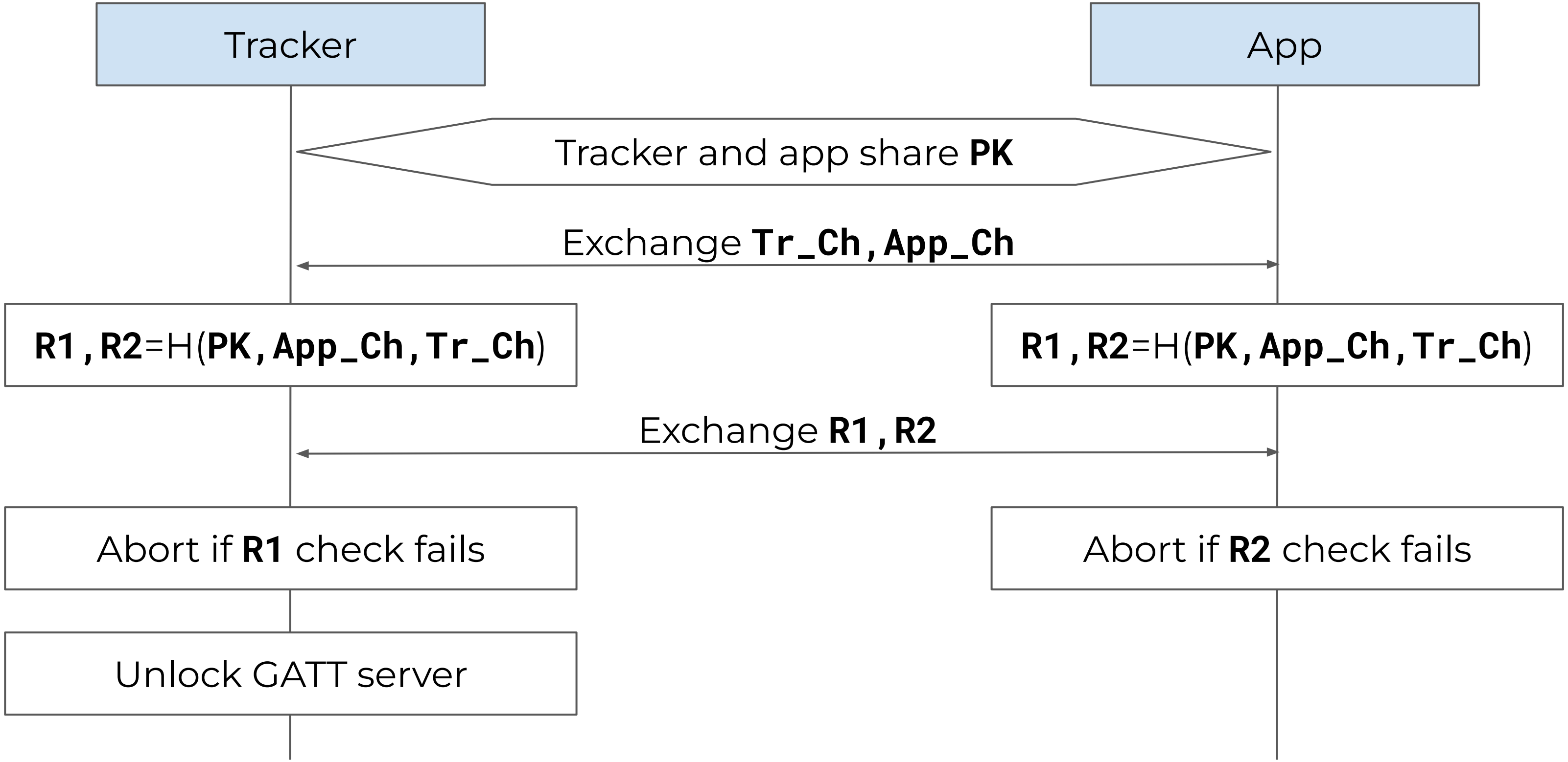
Countermeasures

1. ECDH User-Authenticated Pairing
2. PK Authenticated Session with AE crypto
3. BLE Link-layer security (defense in depth)

ECDH User-Authenticated Pairing


Tracker
App
Gen **Tr_Pk,Tr_Sk**
Gen **App_Pk,App_Sk**
Exchange **Tr_Pk,App_Pk**
PK=ECDH(**Tr_Sk,App_Pk**)
PK=ECDH(**App_Sk,Tr_Pk**)
Exchange **Tr_N,App_N**
V=H(**PK,Tr_N,App_N**)
V=H(**PK,Tr_N,App_N**)
Display **V**
Display **V**
User confirms V

PK Auth Session with AE crypto



Tracker

App

Tracker and app share **PK**

Exchange **Tr_Ch,App_Ch**

R1,R2=H(**PK,App_Ch,Tr_Ch**)

R1,R2=H(**PK,App_Ch,Tr_Ch**)

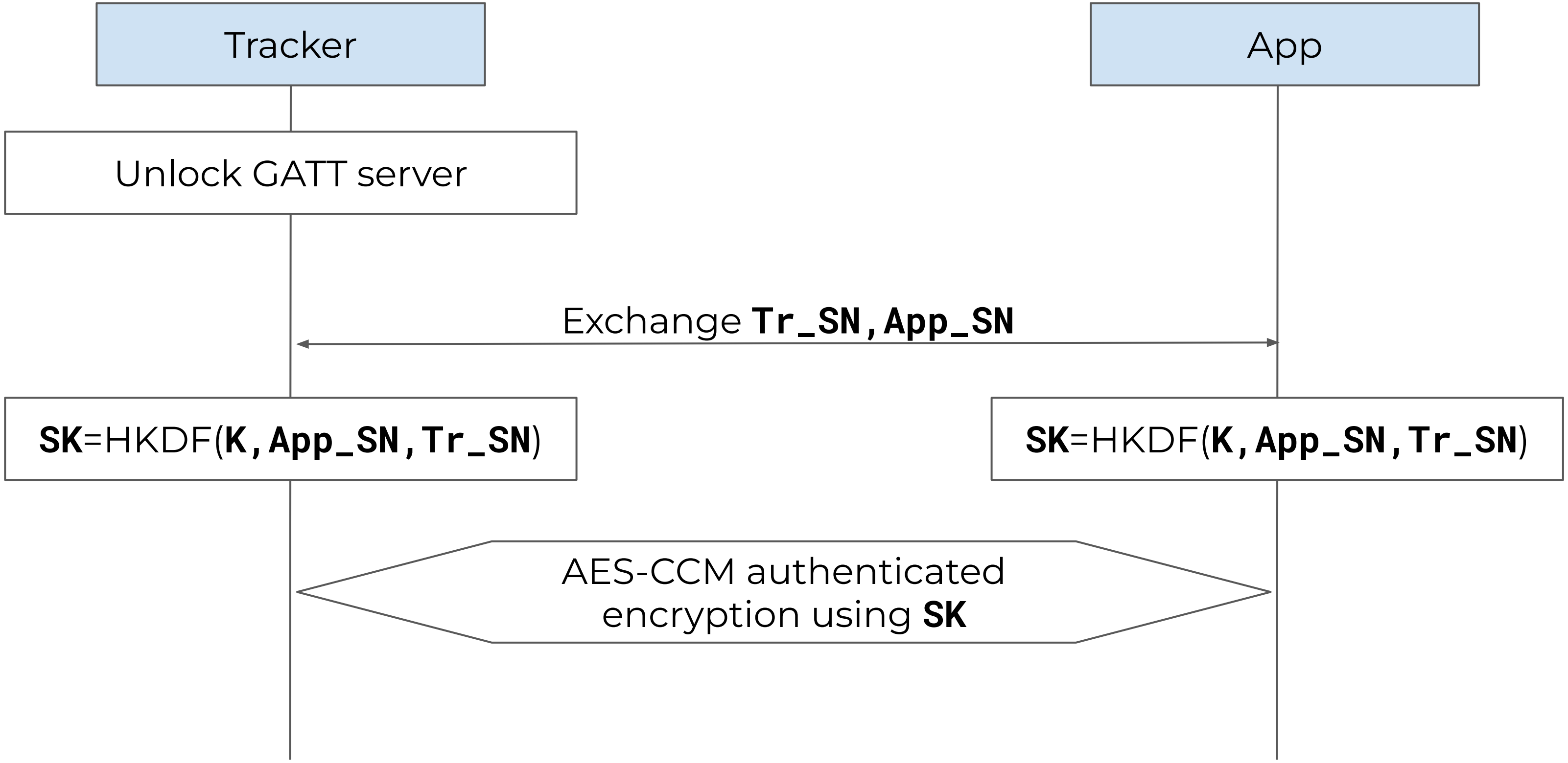
Exchange **R1,R2**

Abort if **R1** check fails

Abort if **R2** check fails

Unlock GATT server

PK Auth Session with AE crypto (2)



Tracker
App
Unlock GATT server
Exchange **Tr_SN,App_SN**
SK=HKDF(**K,App_SN,Tr_SN**)
SK=HKDF(**K,App_SN,Tr_SN**)
AES-CCM authenticated encryption using **SK**

BLE Link-Layer Security

- Trackers and app **support** BLE security
  - Pairing and Session establishment
- Xiaomi should **enable** this feature
  - Defense in depth
  - With **limited overhead**

Responsible Disclosures

- Xiaomi response
  - Identified as a known *“Lack of encryption”* vulnerability
  - When we shared multiple vulns and attacks :(
  - To be fixed at an undisclosed date
- Fitbit (Google) response
  - Acknowledged the findings, released a **fix**
  - Invited to hack next-gen trackers

This is it! Q&A

- Intro on proprietary fitness tracking ecosystems
- Reverse engineering (RE) methodology
- Xiaomi FTE vulns and attacks
- BreakMi OS toolkit and (live) demos
- Fitbit FTE vulns and attacks
- Countermeasures and responsible disclosure
- **More**: CHES paper, slides, poster, video