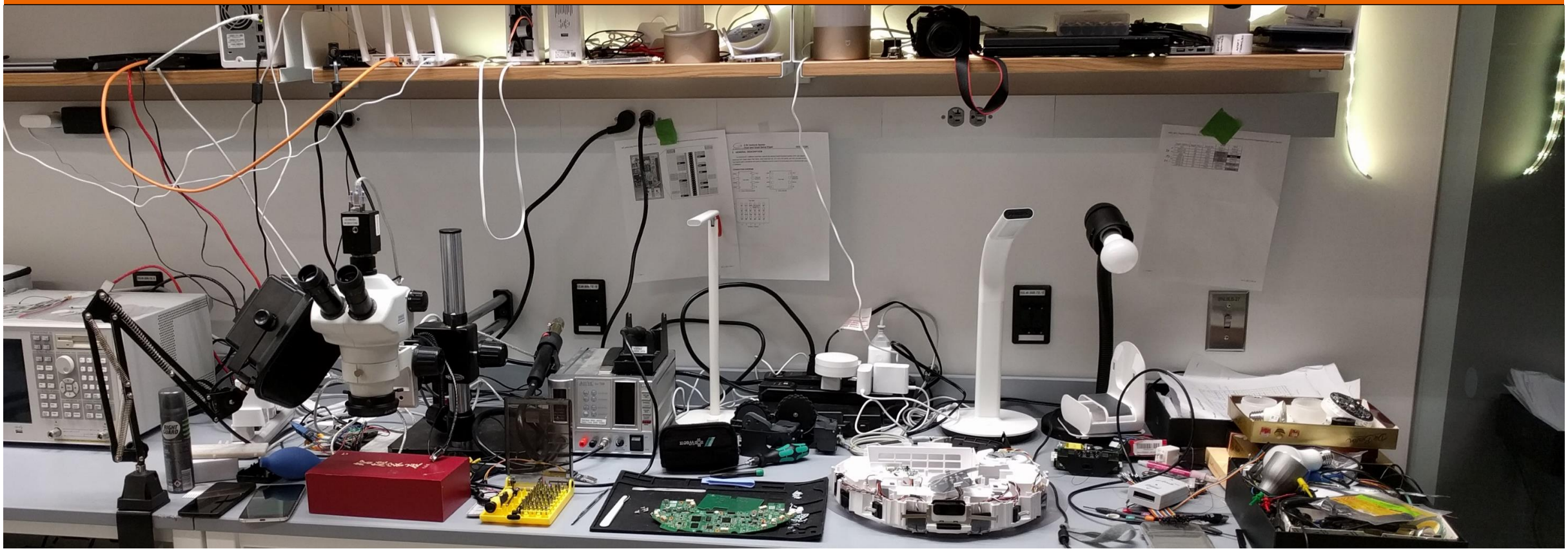
Security Analysis of the Xiaomi IoT Ecosystem

Advisors: Jiska Classen, Guevara Noubir

TECHNISCHE UNIVERSITÄT DARMSTADT

SEEMOO SECURE MOBILE NETWORKING



16.09.2019 | Dennis Giese | dgiese@seemoo.de

Outline

- Introduction
- Methodology
- Analysis of Mi Home App
- Analysis of Devices
- Discussion
- Conclusion

Outline

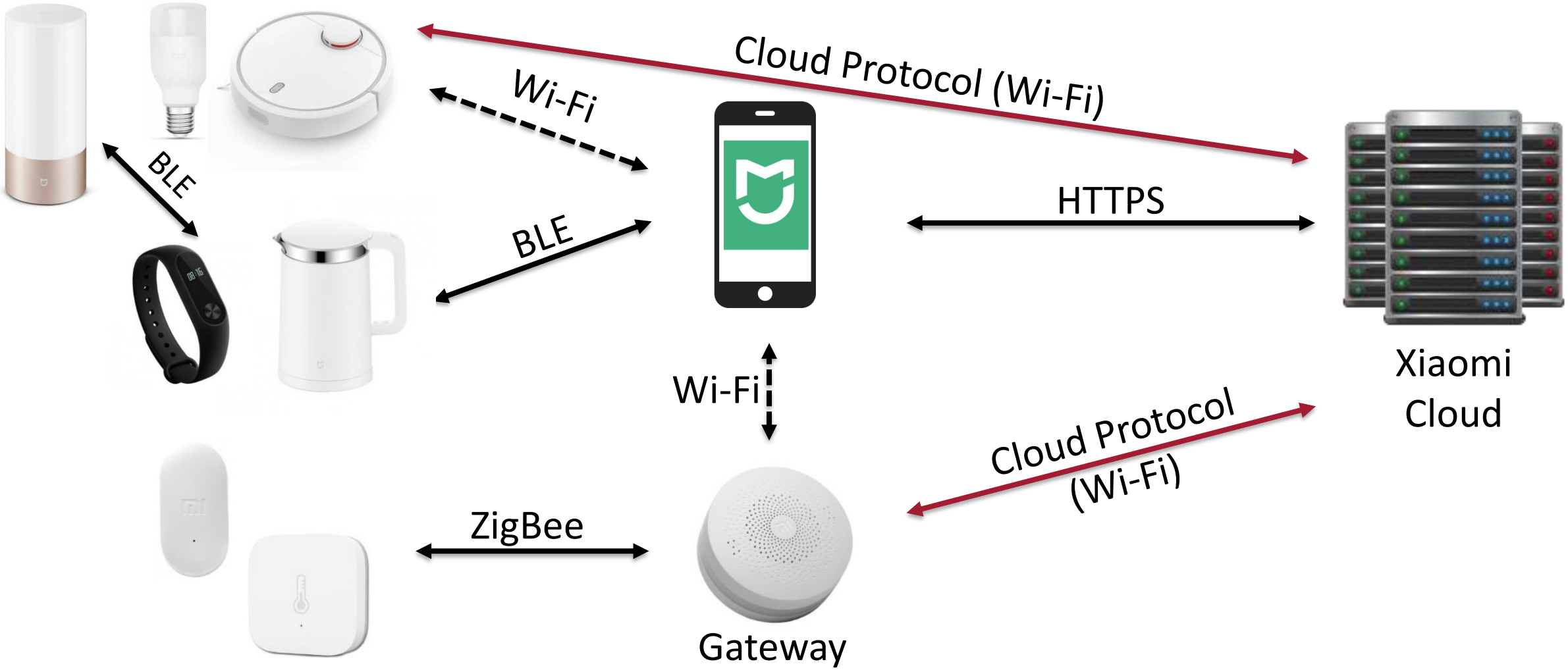
- Introduction
- Methodology
- Analysis of Mi Home App
- Analysis of Devices
- Discussion
- Conclusion

Introduction
The Xiaomi Ecosystem

- Xiaomi mostly known for Smartphones (4th worldwide)
- They claim to have the biggest IoT ecosystem worldwide
  - 171 Million Devices, 820 different models (June 2019)
- Different Vendors, **one ecosystem**
  - named „Mijia“
  - Same communication protocol
  - Different technologies supported
  - Implementation differs from manufacturer to manufacturer
    - Software quality very different
    - Custom features added to firmware

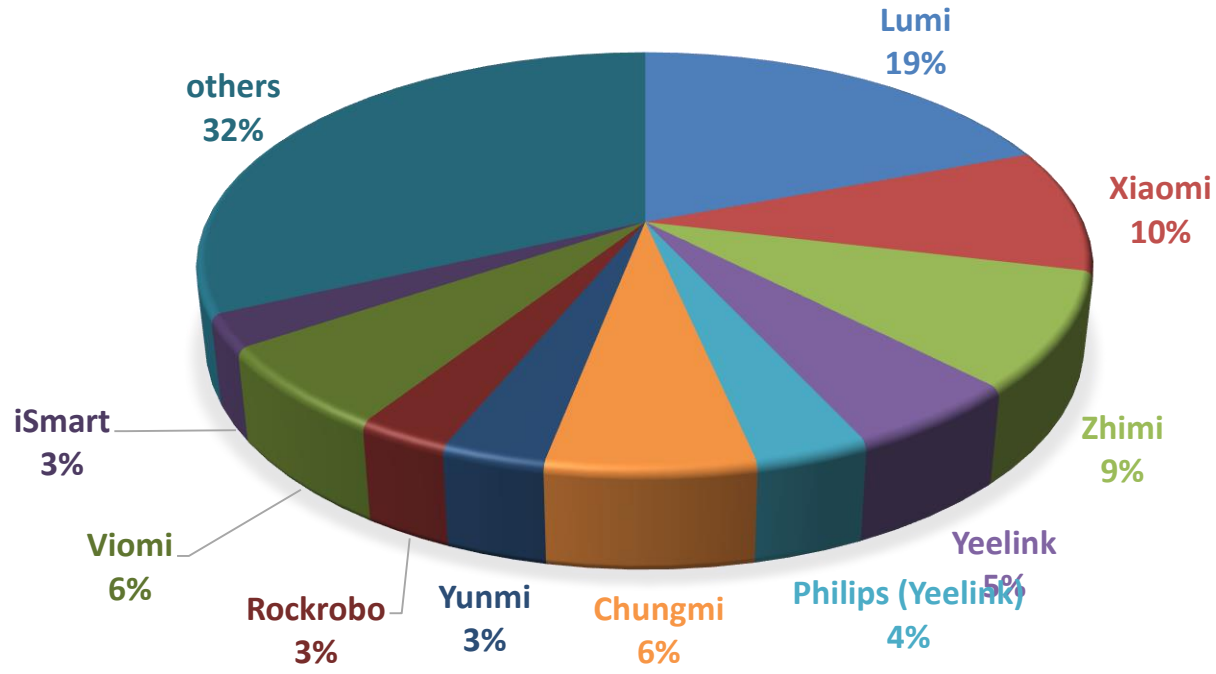
Introduction
Products

Introduction
Communication relations



Introduction
Different vendors in one ecosystem

- ~820 different models supported (Wi-Fi + Zigbee + BLE)
- Depending on selected server location
  - Mainland China
  - Singapore (Worldwide)
  - Russia
  - US
  - Germany (Europe)
  - India
- models might be region-blocked



Values estimated, Mi Home 5.6, Mainland China Server

Motivation

- IoT devices have high impact in the daily life
  - Smart home devices gain more importance and are common
  - Devices have much computation power
  - IoT means that devices are connected to the Internet
  - Devices may collect much private data
  - However: User cannot inspect functionality of the device
- Xiaomi Ecosystem is a good target for security analysis
  - Due to market share impact on many customers
  - Many different implementations can have security vulnerabilities
  - Same protocol makes knowledge transferable to many devices
    - Mijia SDK is shared for all the devices

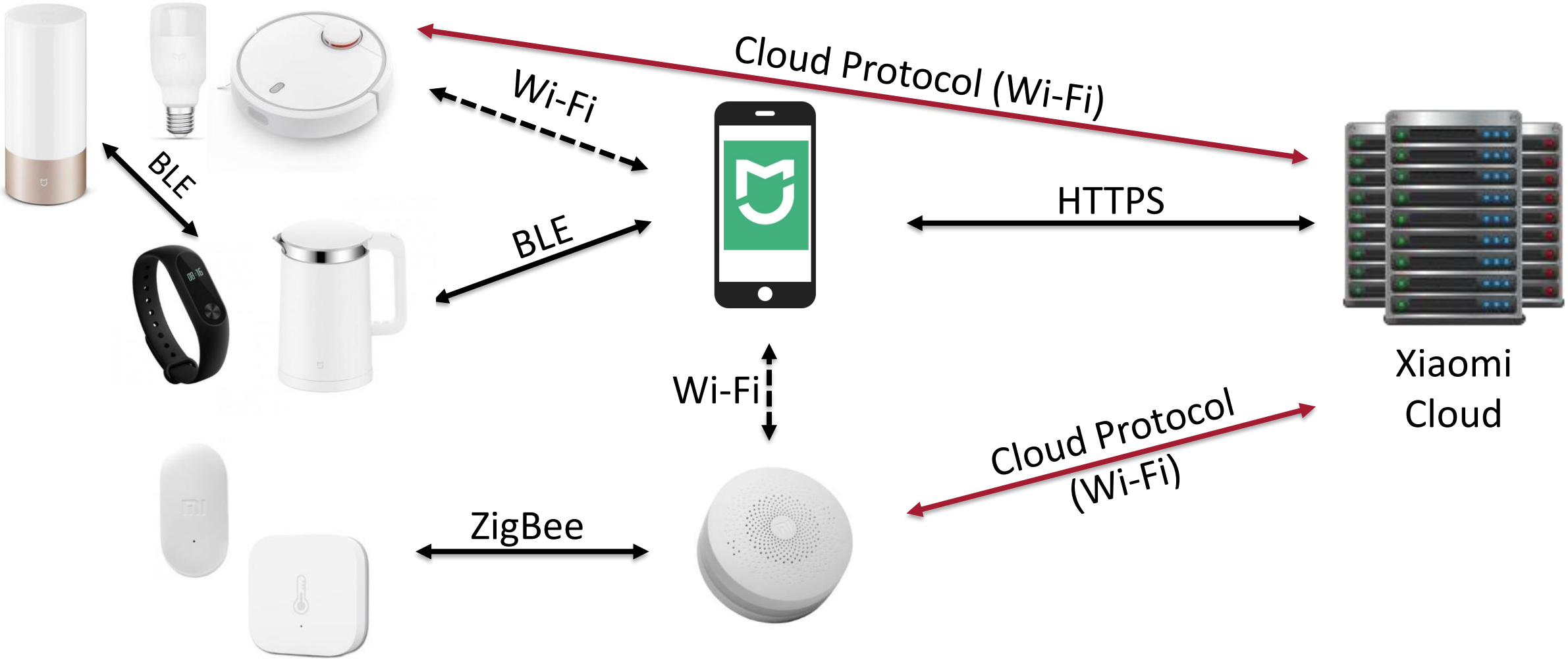
Goals

- Research question: How secure is the implementation of the ecosystem of the IoT market leader Xiaomi?
- Subgoals:
  - Analyze and understand functionality
  - Find potential vulnerabilities
  - Analyze the impact on the users privacy
  - Enable users to take control over their own devices
- Focus: ARM based devices with Wi-Fi

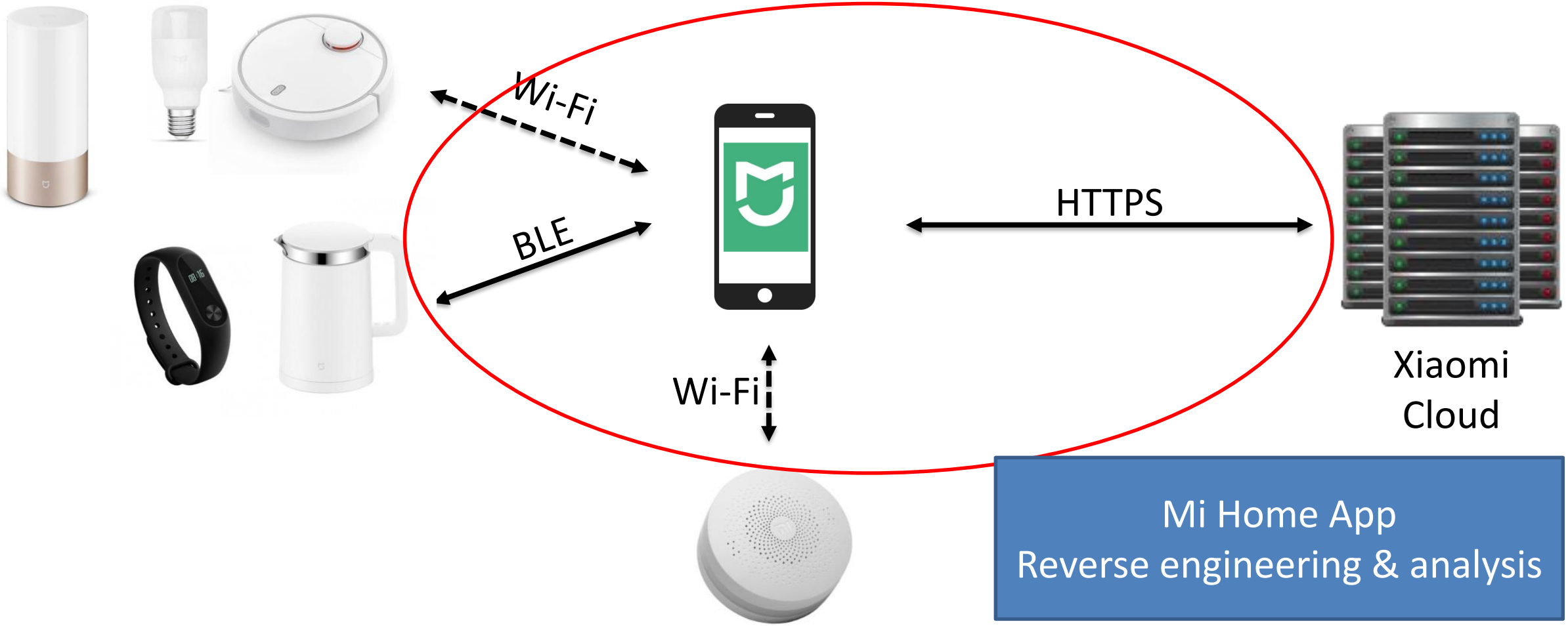
Outline

- ✓ Introduction
- ➢ Methodology
- Analysis of Mi Home App
- Analysis of Devices
- Discussion
- Conclusion

Methodology
Approaches



Methodology
Approaches: App



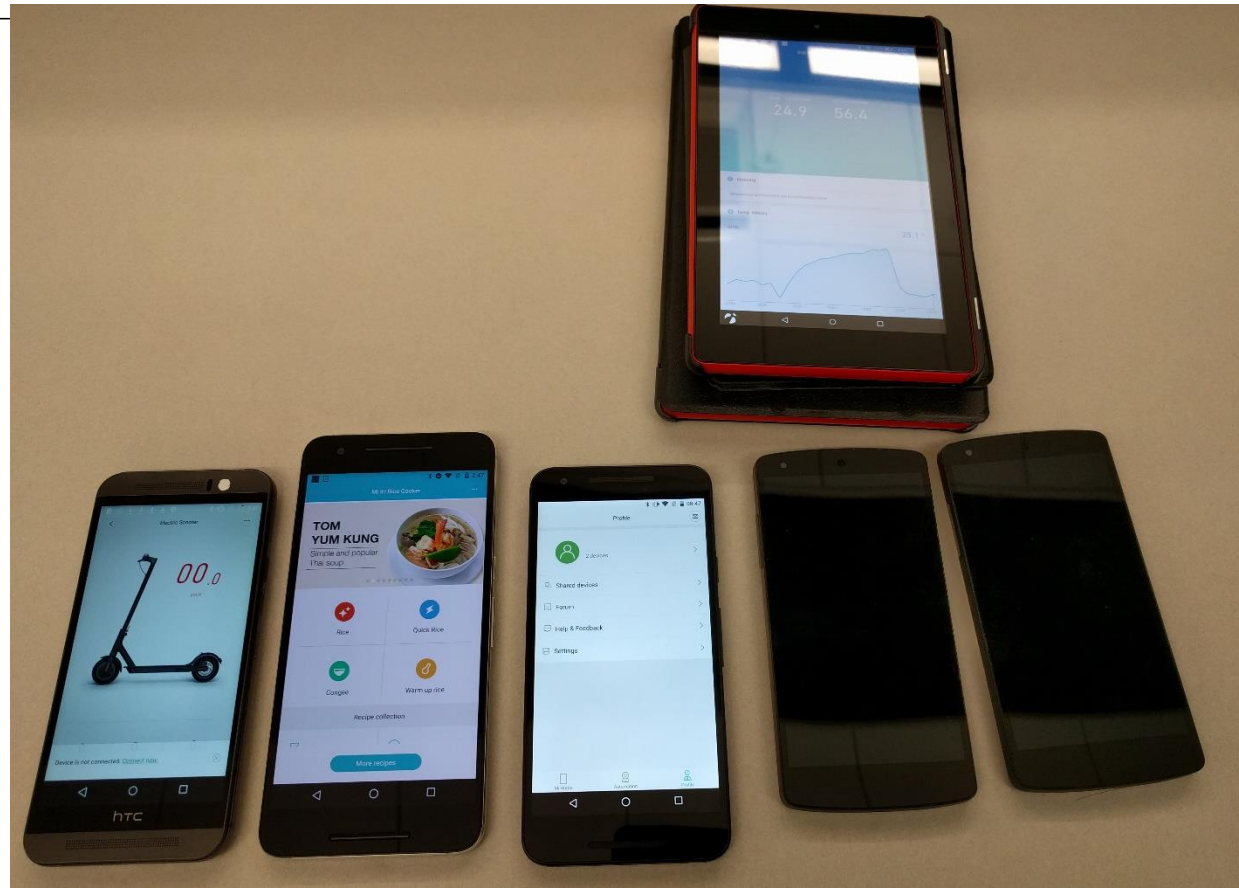
Methodology
App Reverse Engineering

- Idea: Understand interaction between app and phone, and app and cloud
- Advantage: device data is displayed inside the app -> app needs to know how to interpret it
- Methods:
  - Disassembly: Jadx (APK to Java)
  - Modification: Apktool (APK to smalicode, rebuilding)
  - Monitoring: Logcat (monitoring Android log files)
  - Interception: Xposed framework (modifying flows while execution)

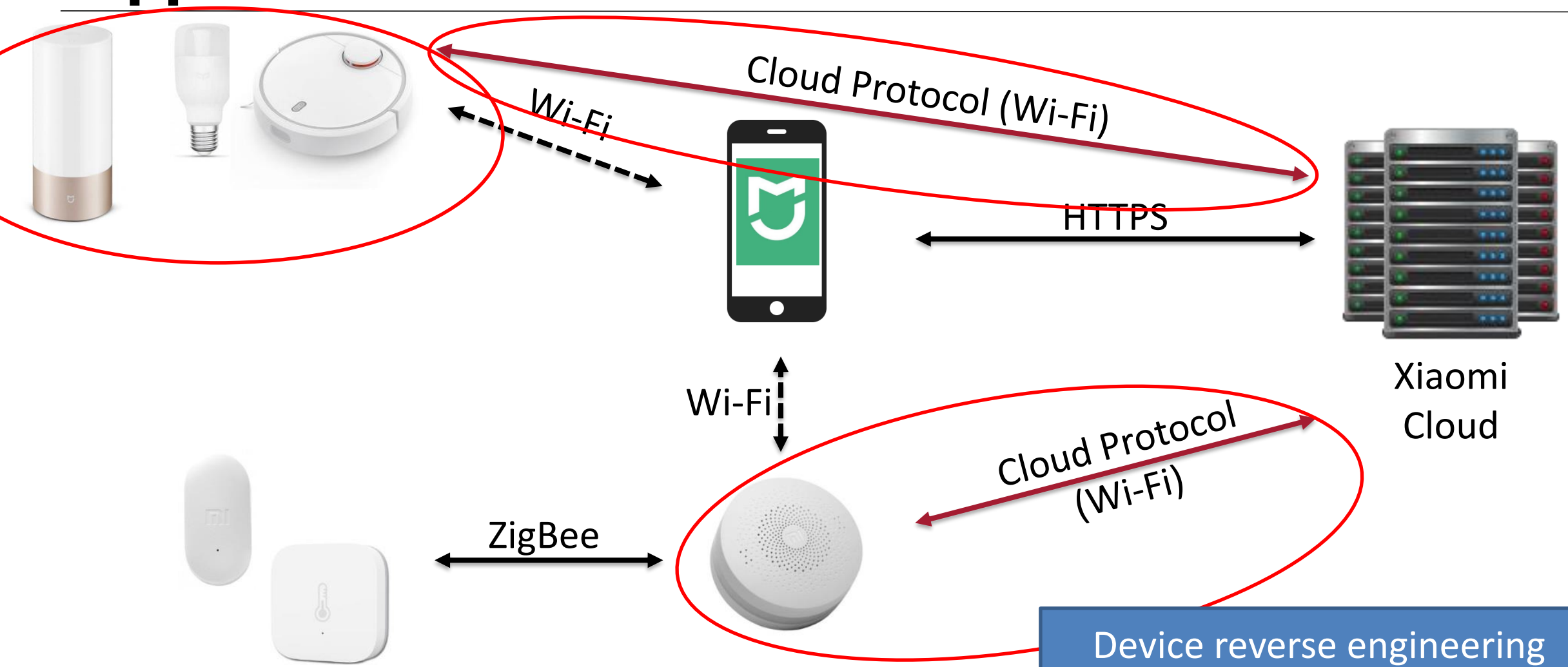
Methodology

How we stay undetected?

- Multiple smartphones/tablets
  - Different Xiaomi accounts
  - Different server location
  - Spoofed GPS coordinates
- Wi-Fi Network
  - Separate Wi-Fi access points
  - VPNs to Hong Kong, China
  - TOR
- No mixture between different accounts and devices



Methodology
Approaches: Devices



Methodology
Device Reverse Engineering

- Idea: Understand function and design of devices (physical hardware)
- Advantage: Data can be obtained directly from the device, transport encryption can be avoided
- Methods (Workflow):
  - Retrieving firmware before purchasing
  - Disassembly of the device and PCB analysis
  - Identification of platform and components
  - Desoldering flash and dumping contents
  - Network traffic analysis
  - Obtaining root access
  - Verify collected user information on devices

Methodology
Device Procurement

- ~170 devices

Methodology
Device selection

- ARM based devices mit Wi-Fi
- Multiple devices for each model
  - One reference
  - One to disassemble and root
- Selection by usefulness and size
  - No fridges, washing machines, ... ☹

Comparison

App

- App can be downloaded for free
- Requires Cloud interaction -> legal issues
- Information can be obtained for a large number of models
- Analysis reveals vulnerabilities in cloud APIs
- Vulnerabilities can be fixed by the cloud provider easily

Devices

- Requires procurement of devices
- Any attack can be done (even destructive ones)
- Information is valid for a specific set of models
- Analysis reveals vulnerabilities on devices
- Vulnerabilities can be fixed by firmware updates from the vendor, which requires user interaction

Methodology
Comparison

App

- App can be downloaded for free
- Requires Cloud interaction -> legal issues
- Information can be obtained for a large number of models
- Analysis reveals vulnerabilities in cloud APIs
- Vulnerabilities can be fixed by the cloud provider easily

Devices

- Requires procurement of devices
- Any attack can be done (even destructive ones)
- Information is valid for a specific set of models
- Analysis reveals vulnerabilities on devices
- Vulnerabilities can be fixed by firmware updates from the vendor, which requires user interaction

Preferred method

Outline

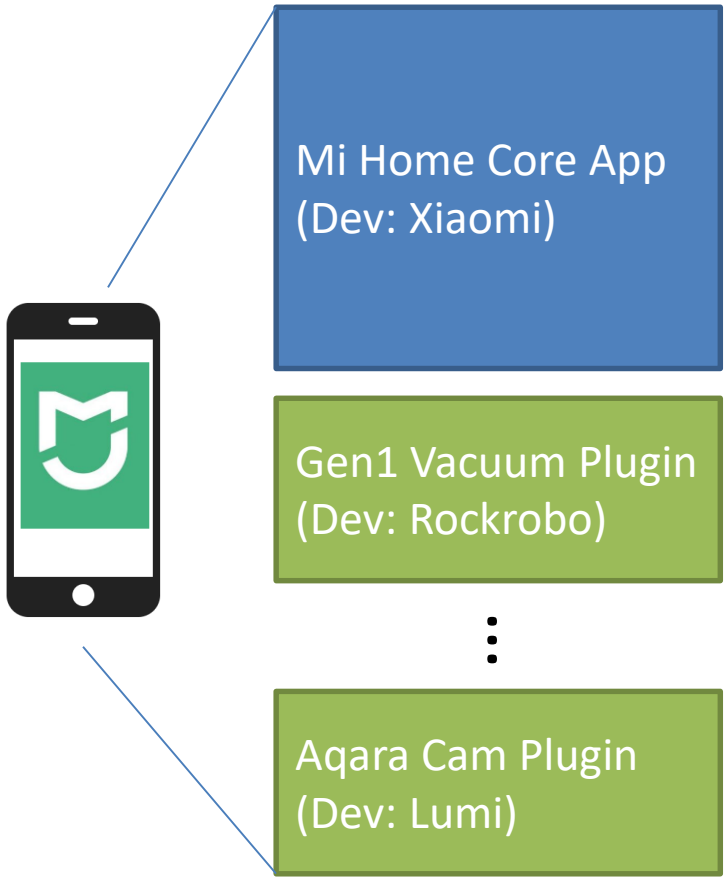
- ✓ Introduction
- ✓ Methodology
- ➢ Analysis of Mi Home App
- Analysis of Devices
- Discussion
- Conclusion

Analysis of App
Mi Home App (Android)

- App partially obfuscated, usage of native libraries
- Device specific functions: provided by Plugins (APK or JS-Bundles)
- Communication to cloud:
  - Authentication via OAuth
  - Layered encryption
    - Outside: HTTPS
    - Inside: AES using a session key
  - Message format: JSON RPC

➢ Contribution: PHP implementation of App to Cloud API

Analysis of App
App Structure



Core functionality:
- Device add, delete
- Device list
- Device Updates via Cloud
- Plugin Management
- Permission Management

Example Endpoints:
/home/device_list {“ssid”: “…”, bssid: “…”}
/home/multi_checkversion {"dids":[…]}
/home/devupgrade {"did":“…"}
/device/devicepass {"did":“…"}

Device specific functionality:
- Device commands
- Map download and presentation

Endpoints:
/home/rpc/#did# {“method”: …}

Device specific functionality:
- Device commands
- Camera feed view (via P2P)
- Zigbee device control

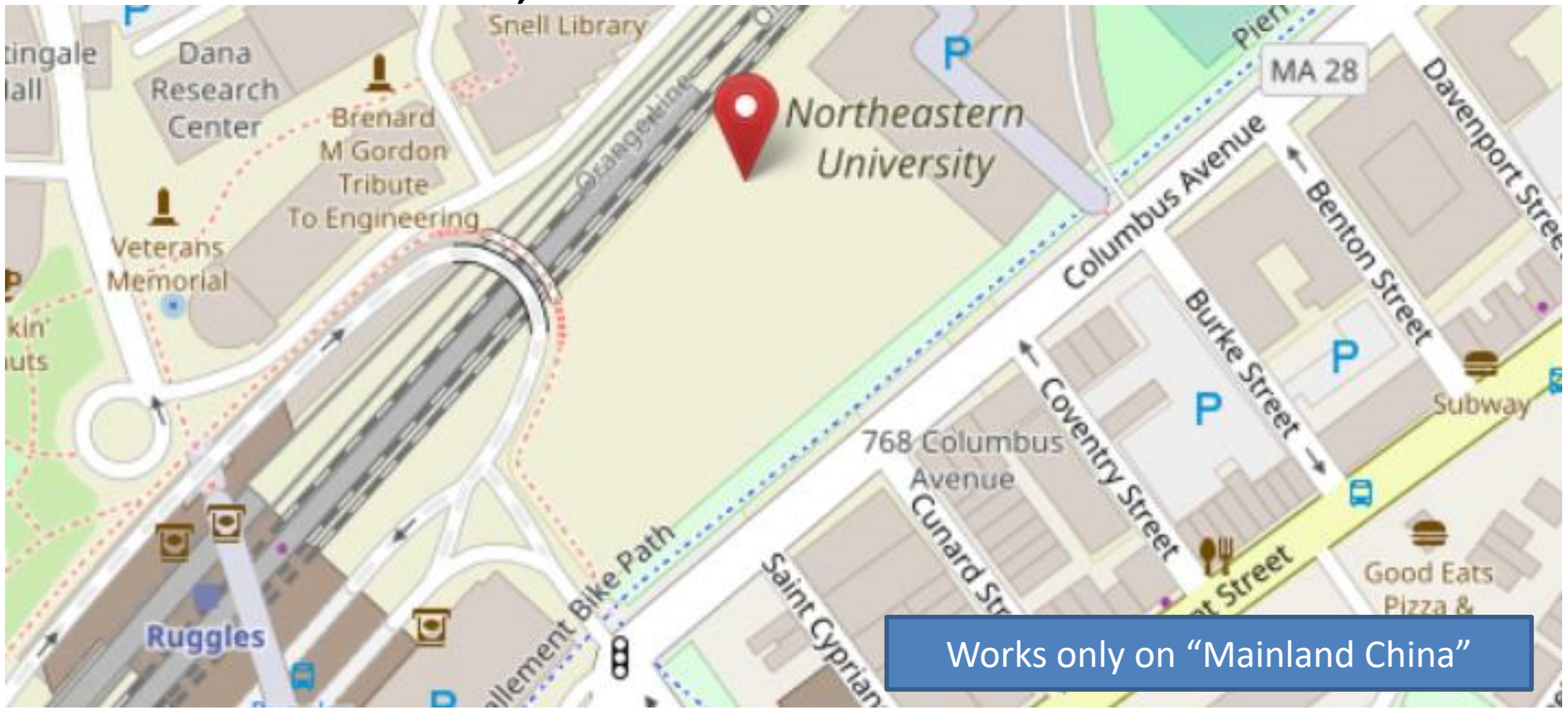
Analysis of App
Example of intercepted cloud api call

- 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}

Analysis of App
Example of intercepted cloud api call

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



Source: Openstreetmaps

Analysis of App
App handling of user permission

- Plugin determines permission based on flags

  "adminFlag":1,"shareFlag":0,"permitLevel":16

  User is owner of device

  Device is not shared

  Privilege level (device dependent)

  – User can update firmware, set settings, share device, etc

Analysis of App
App handling of user permission

- Plugin determines permission based on flags

  "adminFlag":0,"shareFlag":1,"permitLevel":4, “uid”: 123

  User not owner of device

  Device is shared

  Privilege level (device dependent)

  – User can only view device, other options are not visible

Analysis of App
App to Device via Cloud RPC

Please do not distribute this information yet

Analysis of App
Device management

- App retrieves JSON file with all supported devices
  - List acts as a whitelist
  - List depends on region and permission
- Devices detected via Wi-Fi SSID format
- Required for device provisioning: Wi-Fi credentials, UserID, Token

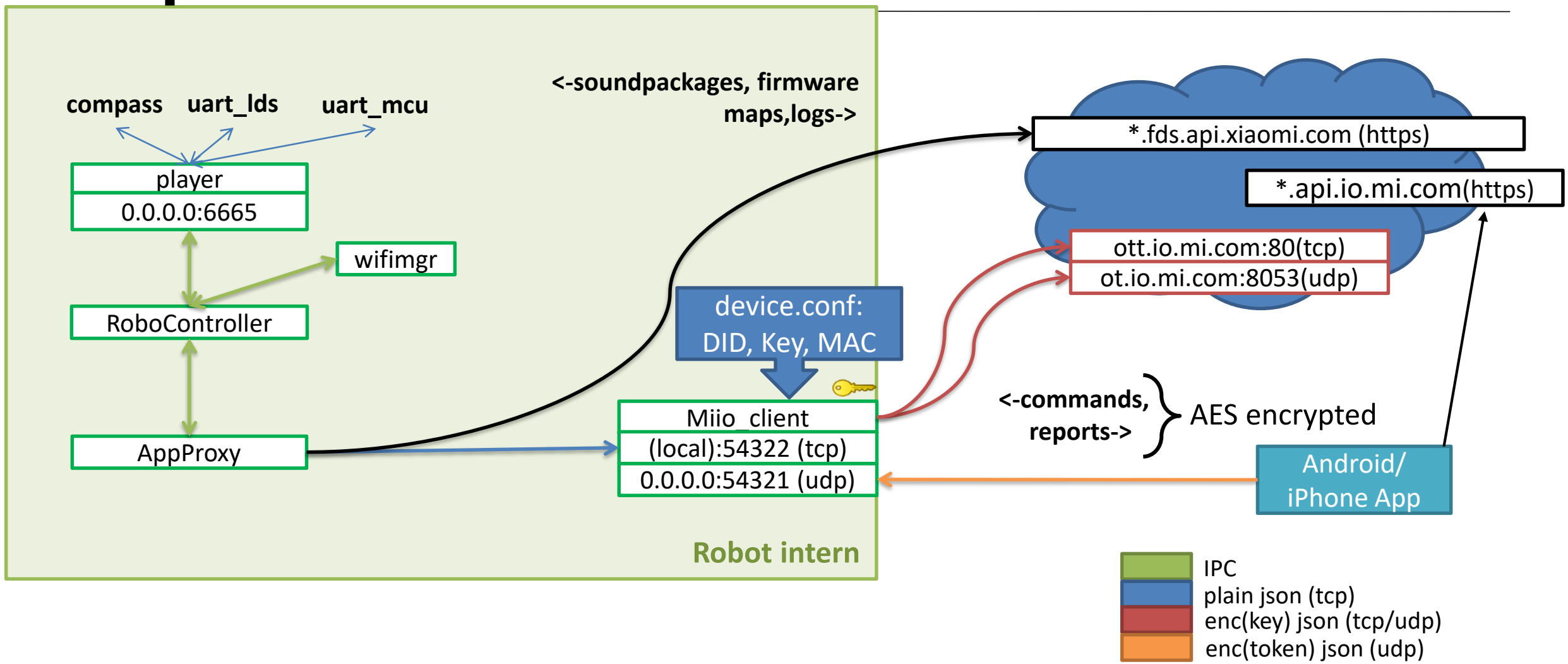
➢ Contribution:

  - List for collecting information about new devices and features
  - Collection of historic information (2017-2019: 3600 devices)
  - Add devices to unsupported regions

Outline

- ✓ Introduction
- ✓ Methodology
- ✓ Analysis of Mi Home App
- ➢ Analysis of Devices
- Discussion
- Conclusion

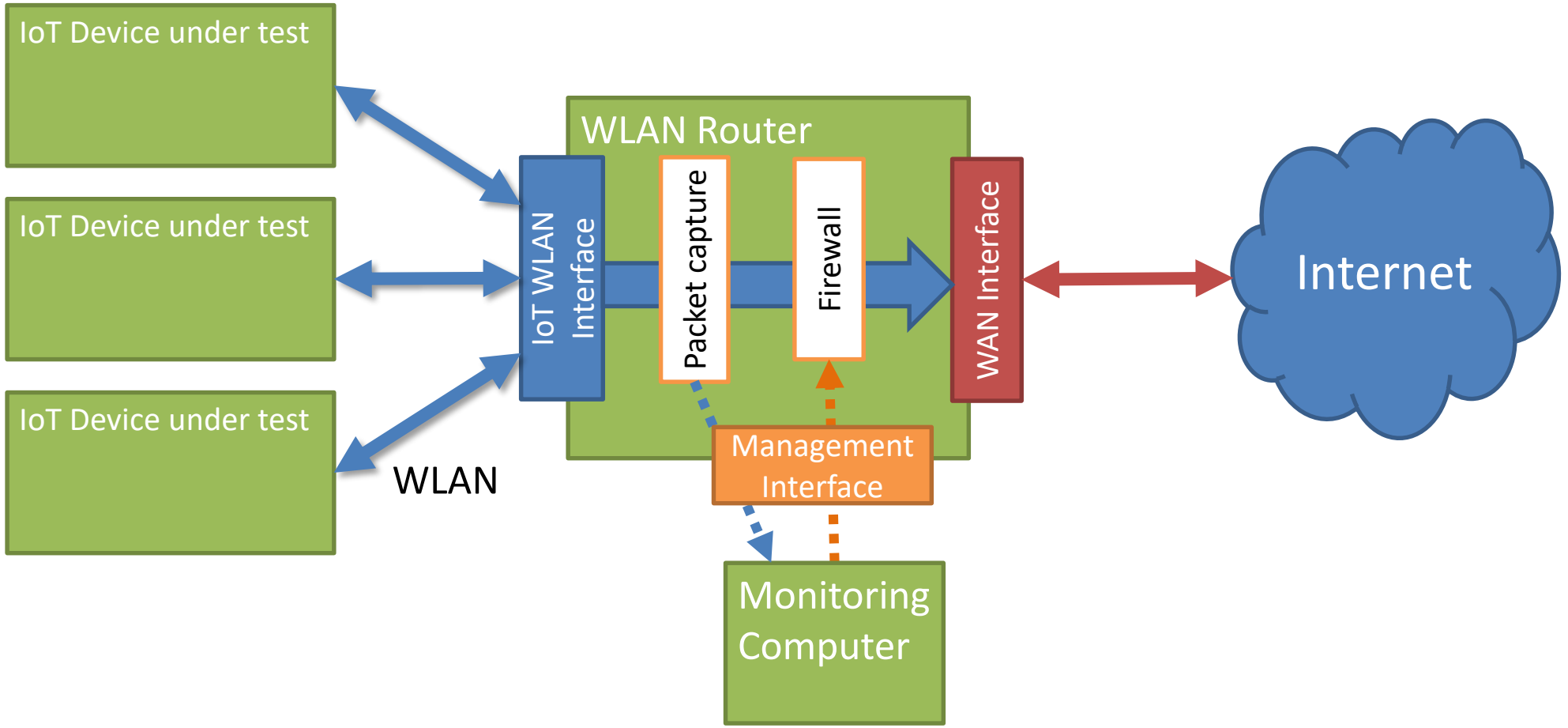
Analysis of Devices
Deeper Look at Communication relations



Device to Cloud Communication

- DeviceID
  - Unique per device
- Keys
  - Cloud key (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 Wi-Fi)

Analysis of Devices
Network Setup



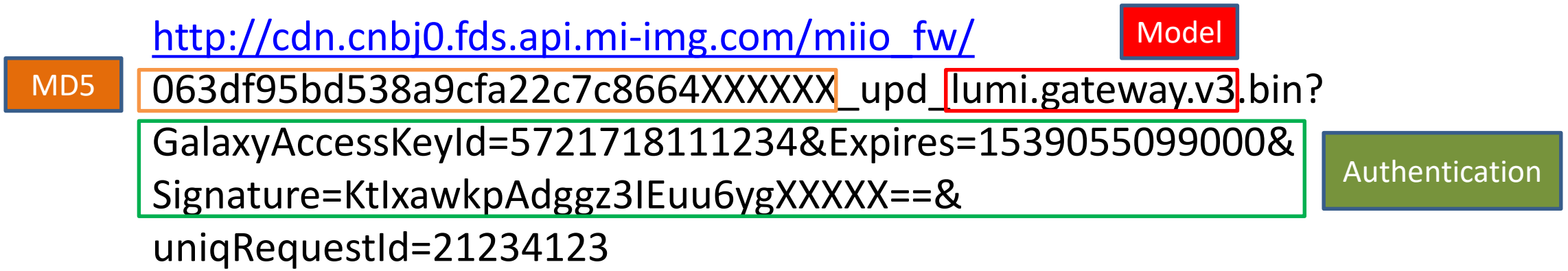
Firmware retrieval

- Dumping Flash memory
  - JTAG, SWD or desolder Flash
  - Helpful tool: Raspberry Pi with OpenOCD and flashrom
- Intercepting traffic while Firmware Update
  - It is advised to actually block the Update
    - Sneaky: If DNS fails then direct IP is used
  - If SSL is used: so far a fake certificate worked ☺
  - Goal: Retrieve special URL for Firmware update

Analysis of Devices

Firmware downloads

- Filenames not easy guessable
- CDN is using URL authentication



Analysis of Devices
How to get Firmwares?

- Problem: Retrieving Firmware is difficult
  - Need of owning the device
- Easy solution: Impersonating devices
  - Model ID initially not fixed in cloud backend -> we can modify it (per region)
  - On rooted device:
    - change model, modify version number to “0.0.0”
    - trigger firmware update from smart phone app
    - Get authenticated firmware URL ☺

➢ Contribution: collection of firmware versions over a long time (2018-2019: 870)
  - Sharing with other researcher for development of open source implementations

Analysis of Devices
Collection of firmwares and device info

modelname. : **roborock.vacuum.s5**
pid : 0
feat_bt_gateway : 0
feat_mesh_gateway : 0
hasBT : -1
hasWiFi : 1
has5GWiFi : 0
hasZigbee : -1
OS : Ubuntu 14.04
RAM : 512MByte
FLASH : 4GByte eMMC
SOC : Allwinner R16
MCU : STM32F103VCT6
SOC-ARCH : ARM Cortex-A7 (4x)
MCU-ARCH : ARM Cortex-M3
WiFi-Chipset : RTL8189ETV
FW-Format : dd image AES encrypted (ccrypt, key: rockrobo)

Region	cn	de	ru	sg	us
first seen	2019-03-30	2019-03-30	2019-03-30	2019-03-30	2019-03-30

Type	**MD5**	**Filename**	**Version**	**Datetime**	**Regions**
app	9e2c0809cebc892c60c6723b30d76016	v11_001768.fullos.pkg	3.3.9_001768	2019-03-27 11:57:00	cn,de,ru,sg,us
app	e7c6f4062b6717d9b7ea1cebeb48f3a8	v11_001720.fullos.pkg	3.3.9_001720	2019-05-23 02:23:00	de,sg,us
app	3d04e386856129a0c0a9508c40e577b7	v11_001864.fullos.lmn09e8u2.pkg	3.3.9_001864	2019-05-31 05:51:00	cn,de,ru,sg
aplugin	77a1d4cfc186aaec8757a27e12d04d88	com.roborock.rubys.app_2019061715280736461.zip	188	2019-06-17 07:28:00	de,sg,us
aplugin	e5d96f0f89b5d8fecdfbd26b829849d4	com.roborock.rubys.app_2019062414482451501.zip	191	2019-06-24 06:48:00	cn

Analysis of Devices
Devices under test

- 21 models selected for test
  - Different regions
  - Different versions

Device name	Region	Mijia model	Vendor	Release	Price (USD)
Aqara Gateway (Homekit)	CN	lumi.gateway.aqhm01	Lumi	Q2 2018	50
Aqara Gateway (Homekit)	US	lumi.gateway.aqhm02	Lumi	Q1 2019	50
Aqara Smart Home Gateway	TW	lumi.gateway.mitw01	Lumi	Q1 2018	35
Aqara Smart IP Camera	CN	lumi.camera.aq1	Lumi	Q4 2017	35
Lumi Smart Home Gateway	CN	lumi.gateway.v3	Lumi	Q3 2016	30
Philips Ceiling Lamp	CN	philips.light.ceiling	Yeelight	Q2 2017	70
Roborock S50	EU	roborock.vacuum.s5	Roborock	Q1 2018	400
Roborock S50	CN	roborock.vacuum.s5	Roborock	Q4 2017	350
Roborock T61	CN	roborock.vacuum.t6	Roborock	Q1 2019	450
Roborock S61	EU	roborock.vacuum.s6	Roborock	Q1 2019	550
Xiaomi Mi Vacuum Robot	CN	rockrobo.vacuum.v1	Roborock	Q4 2016	280
Xiaomi Mi WiFi Speaker	CN	xiaomi.wifispeaker.v1	Xiaomi	Q4 2016	85
Xiaomi WiFi Plug	CN	chuangmi.plug.m1	Chuangmi	Q2 2016	15
Yeelink Bedside lamp	CN	yeelink.light.bslamp1	Yeelight	Q4 2017	25
Yeelink Bedside lamp	TW	yeelink.light.bslamp1	Yeelight	Q1 2018	30
Yeelink Ceiling Lamp	CN	yeelink.light.ceiling1	Yeelight	Q3 2017	65
Yeelink Light Color	CN	yeelink.light.color1	Yeelight	Q4 2016	10
Yeelink Light Mono1	CN	yeelink.light.mono1	Yeelight	Q4 2016	10
Yeelink Light Strip	CN	yeelink.light.strip1	Yeelight	Q4 2016	15
Yeelink Smart White Bulb	EU	yeelink.light.ct2	Yeelight	Q2 2018	15
Yeelink Smart RGB Bulb	EU	yeelink.light.color2	Yeelight	Q2 2018	15

Analysis of Devices
Devices under test

- 21 models selected for test
  - Different regions
  - Different versions

Device name	Region	Mijia model	Vendor	Release	Price (USD)
Aqara Gateway (Homekit)	CN	lumi.gateway.aqhm01	Lumi	Q2 2018	50
Aqara Gateway (Homekit)	US	lumi.gateway.aqhm02	Lumi	Q1 2019	50
Aqara Smart Home Gateway	TW	lumi.gateway.mitw01	Lumi	Q1 2018	35
Aqara Smart IP Camera	CN	lumi.camera.aq1	Lumi	Q4 2017	35
Lumi Smart Home Gateway	CN	lumi.gateway.v3	Lumi	Q3 2016	30
Philips Ceiling Lamp	CN	philips.light.ceiling	Yeelight	Q2 2017	70
Roborock S50	EU	roborock.vacuum.s5	Roborock	Q1 2018	400
Roborock S50	CN	roborock.vacuum.s5	Roborock	Q4 2017	350
Roborock T61	CN	roborock.vacuum.t6	Roborock	Q1 2019	450
Roborock S61	EU	roborock.vacuum.s6	Roborock	Q1 2019	550
Xiaomi Mi Vacuum Robot	CN	rockrobo.vacuum.v1	Roborock	Q4 2016	280
Xiaomi Mi WiFi Speaker	CN	xiaomi.wifispeaker.v1	Xiaomi	Q4 2016	85
Xiaomi WiFi Plug	CN	chuangmi.plug.m1	Chuangmi	Q2 2016	15
Yeelink Bedside lamp	CN	yeelink.light.bslamp1	Yeelight	Q4 2017	25
Yeelink Bedside lamp	TW	yeelink.light.bslamp1	Yeelight	Q1 2018	30
Yeelink Ceiling Lamp	CN	yeelink.light.ceiling1	Yeelight	Q3 2017	65
Yeelink Light Color	CN	yeelink.light.color1	Yeelight	Q4 2016	10
Yeelink Light Mono1	CN	yeelink.light.mono1	Yeelight	Q4 2016	10
Yeelink Light Strip	CN	yeelink.light.strip1	Yeelight	Q4 2016	15
Yeelink Smart White Bulb	EU	yeelink.light.ct2	Yeelight	Q2 2018	15
Yeelink Smart RGB Bulb	EU	yeelink.light.color2	Yeelight	Q2 2018	15

Analysis of Devices
Mi Vacuum Cleaning Robot (Gen1)

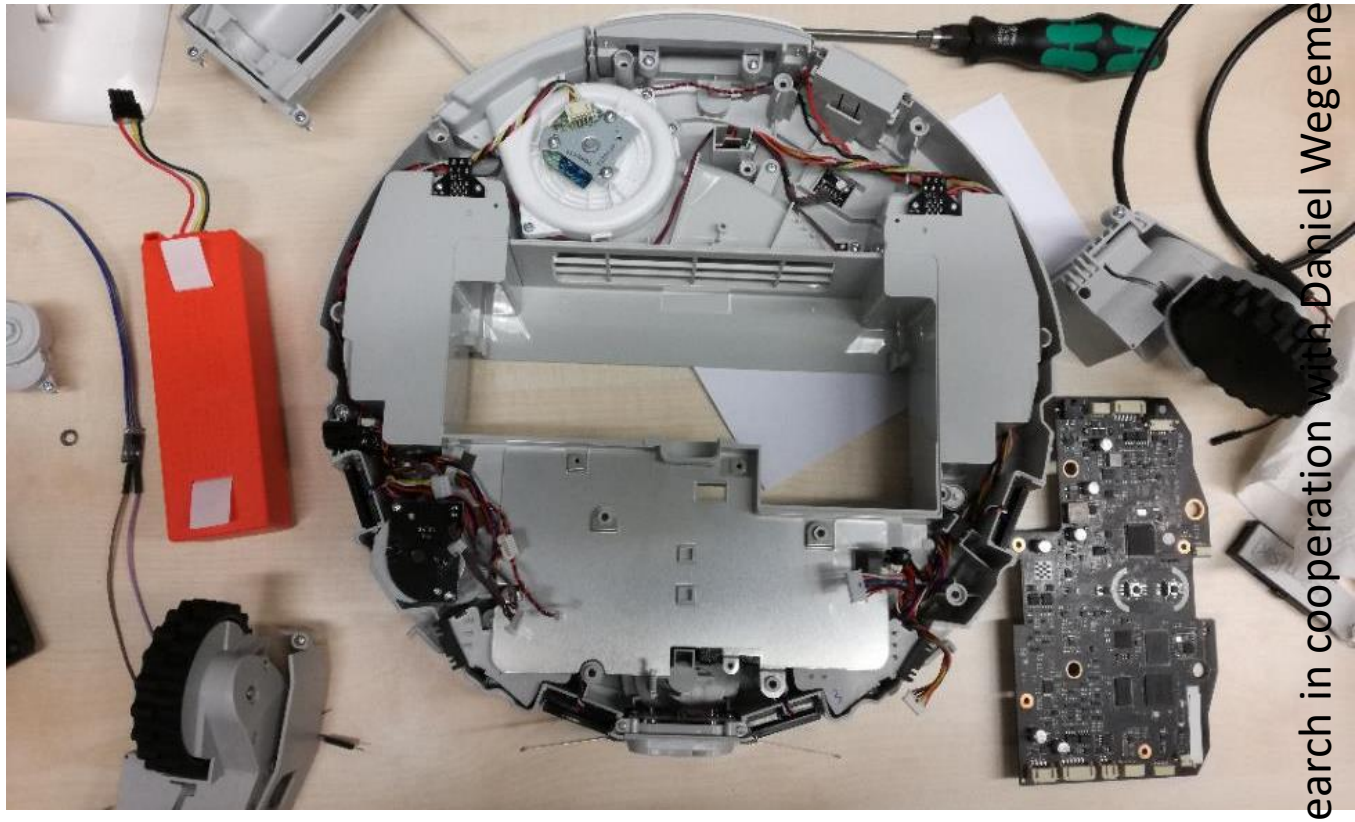
Source: Xiaomi advertisment

Research in cooperation with Daniel Wegemer

Analysis of Devices

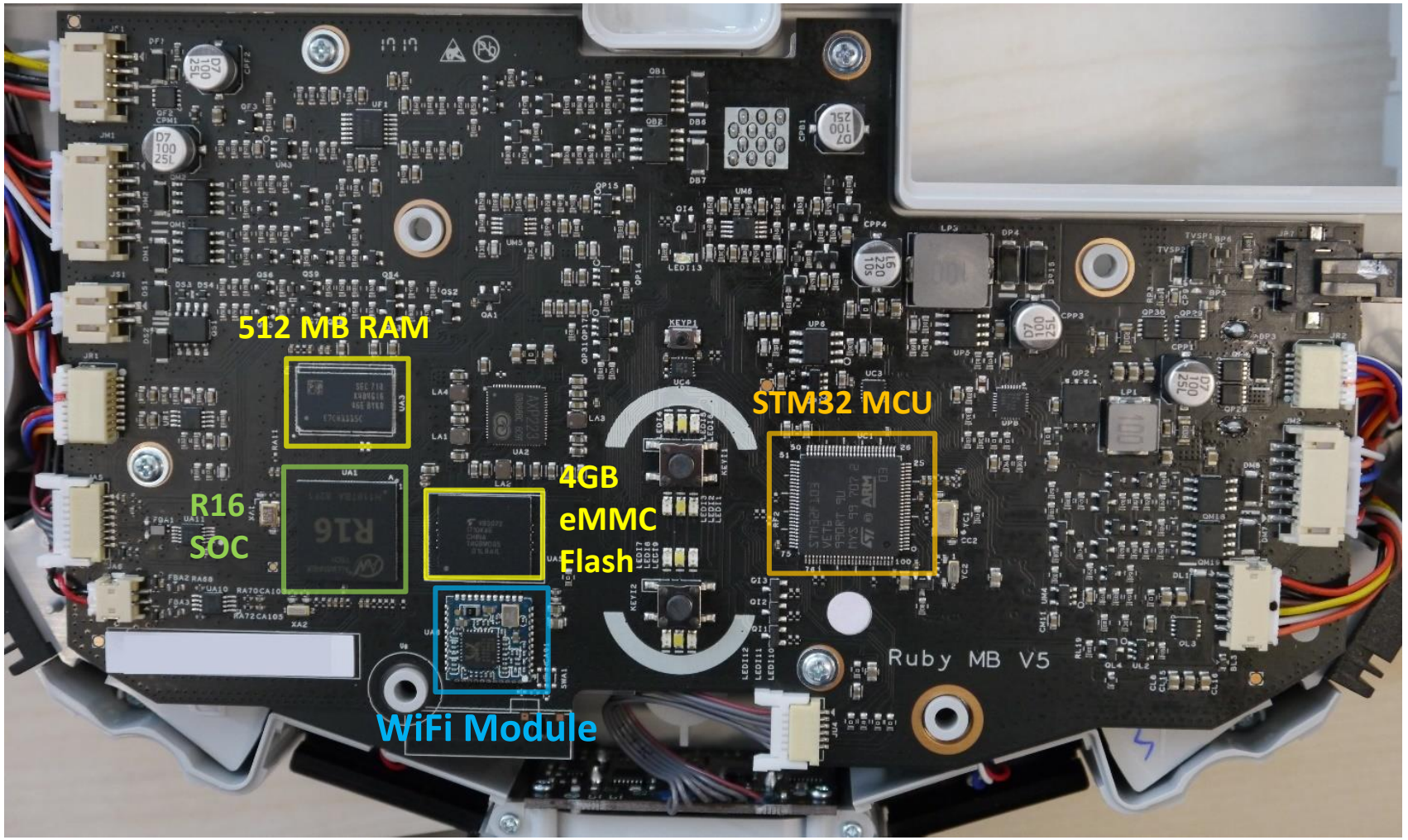
Mi Vacuum Cleaning Robot

- Released 2016
- Hardware:
  - Quadcore ARM SOC
  - 512 MB DDR3 RAM
  - 4GB eMMC Flash
- OS: Ubuntu 14.04
- Protections:
  - Firmware encrypted, debug ports require authentication

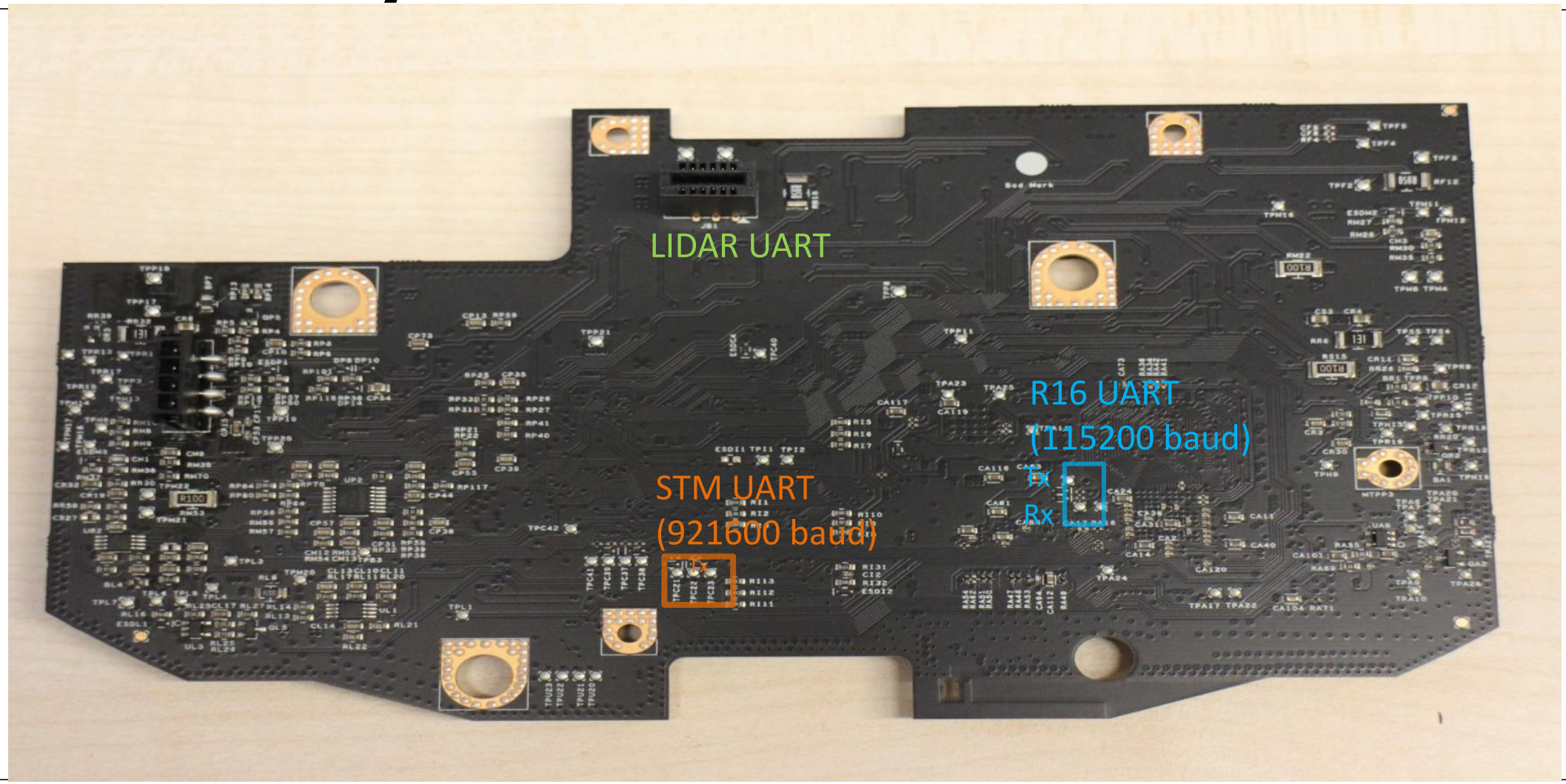


Research in cooperation with Daniel Wegemer

Analysis of Devices
Frontside layout mainboard



Analysis of Devices
Backside layout mainboard



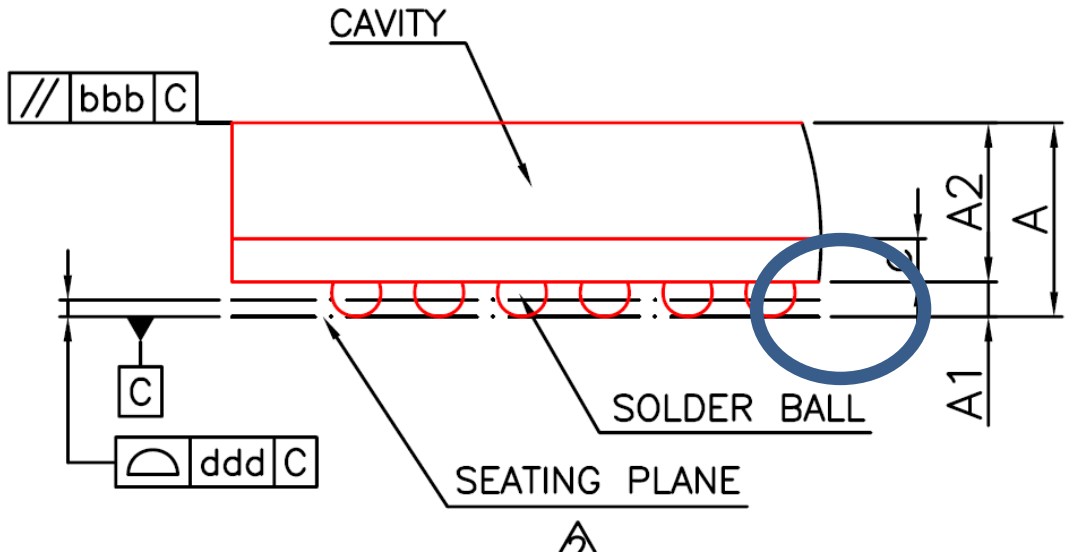
Analysis of Devices
Gaining Root access

- Approach: Fault injection on eMMC flash to enable BOOTROM
  - Usage of aluminum foil to shortcut data pins under the BGA chip
  - Uploading of custom tool via USB and dumping flash
  - Modification and rewriting flash content
- Analysis of firmware and extraction of keys
  - Usage of IDA Pro to extract firmware encryption keys
  - Developing tools for custom firmware and message decryption

➢ Contribution: First published rooting method, description of functions and hardware, reverse engineered data formats and cloud protocol
  - Current usage of rooted vacuum cleaners > 30000
  - Used by researchers for 5G and Wi-Fi experiments, teaching robotics students

Aluminium fault injection attack

- First use of aluminum foil to trigger bootloaders on BGA chips
  - Cheap and simple method
  - Reduced risk in comparison to BGA soldering



Symbol	Dimension in mm		
	MIN	NOM	MAX
A	---	---	1.29
A1	0.25	0.30	0.35
A2	0.84	0.89	0.94
c	0.32	0.36	0.40
D	13.90	14.00	14.10
E	13.90	14.00	14.10
D1	---	12.80	---
E1	---	12.80	---
e	---	0.80	---
b	0.35	0.40	0.45
aaa	0.15		
bbb	0.10		
ddd	0.10		

Analysis of Devices
Available data on device

- Data
  - Logfiles (syslogs, stats, Wi-Fi credentials)
  - Maps
- Data is uploaded to cloud
- Wi-Fi reset
  - Does not delete data: maps, logs still exist
  - Only Wi-Fi credentials are removed, however still exist in logs
- Factory reset
  - Formats user data partition, but is partially recoverable

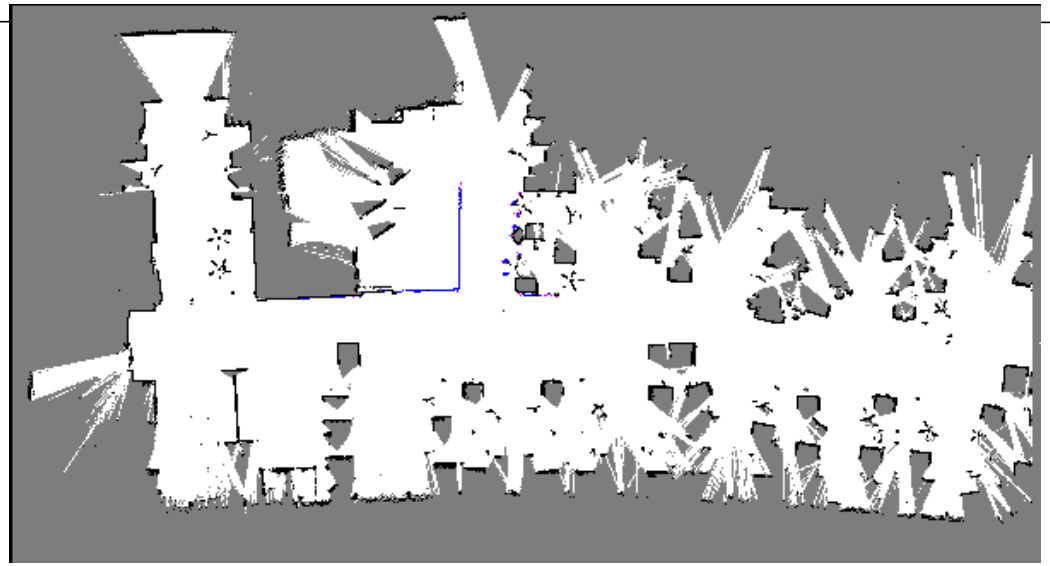
➢ Contribution: Documentation of the usage and collection of data

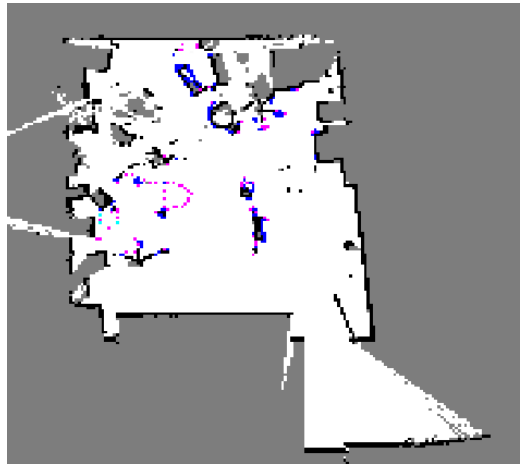
~100 Gbyte writes per Year

Analysis of Devices

Available data on device

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





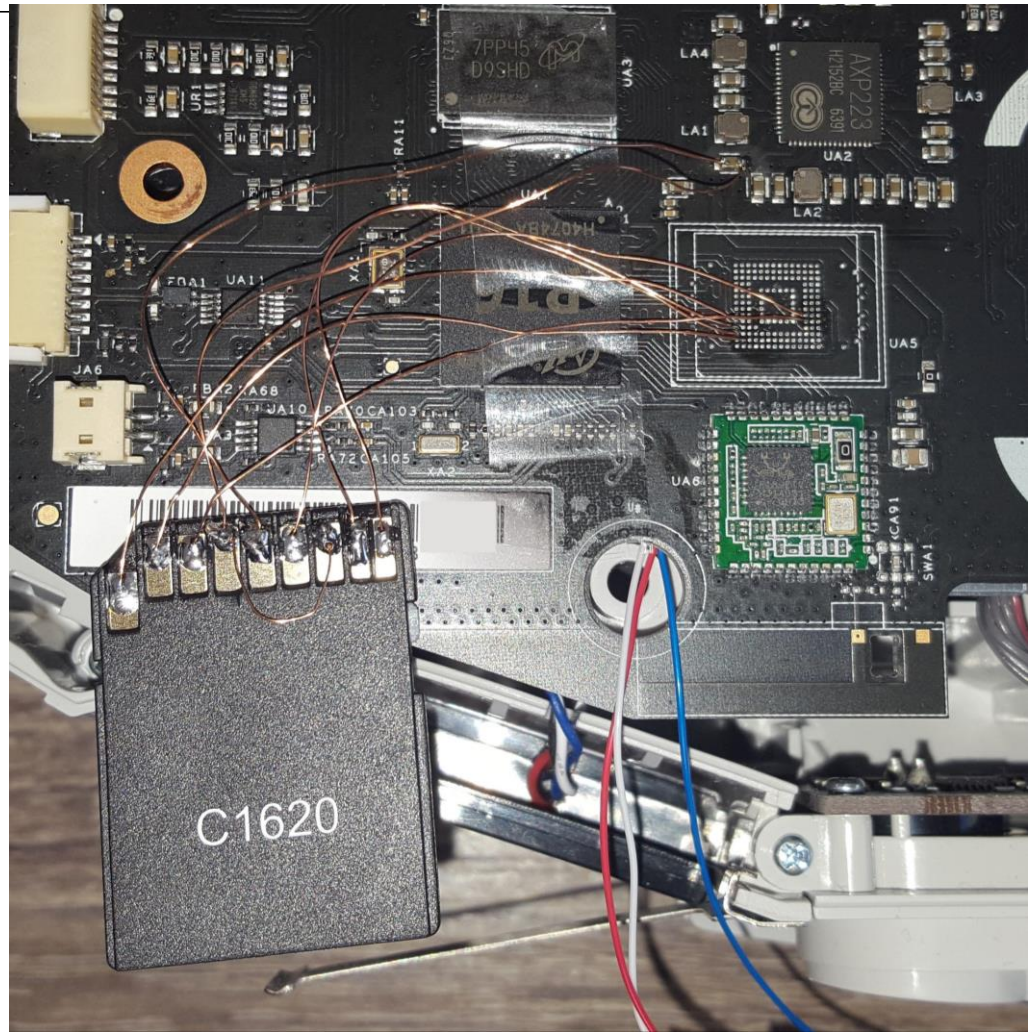
➢ Contribution: Tools for map interpretation

  - Base for all open source implementations

Custom mod of Gen1

- Custom mod enables usage of bigger software (e.g. ROS)

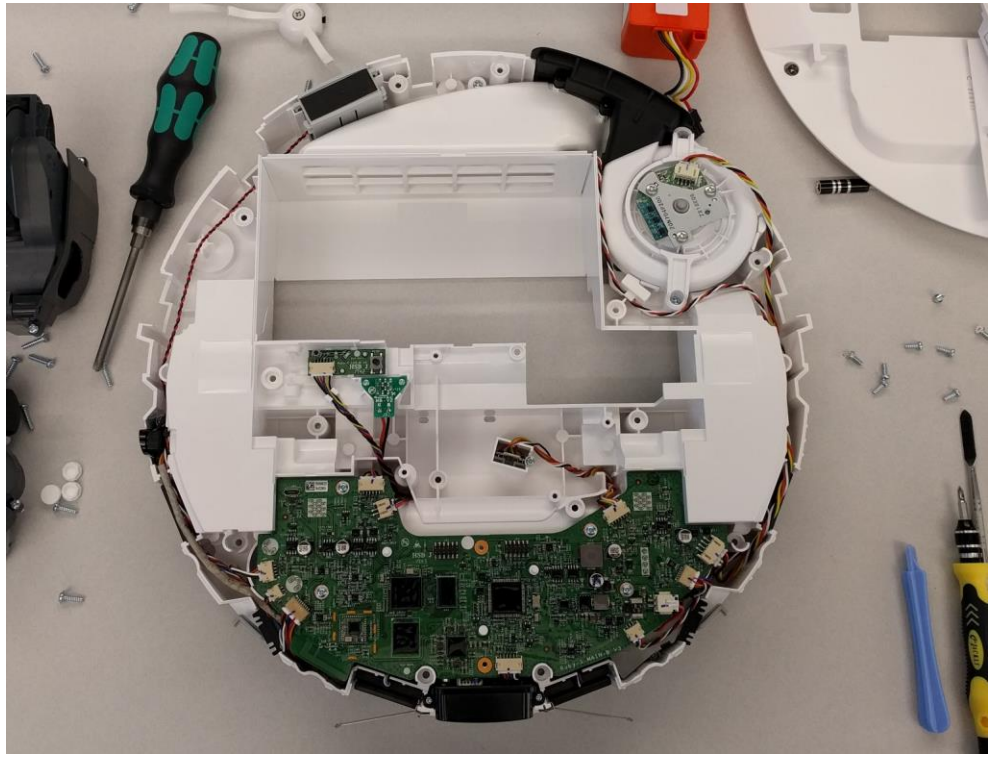
```
[mmc]: ----------------mmc->clock 50000000--
[mmc]: ----------------mmc->bus width 4-----
[mmc]: SD/MMC Card: 4bit, capacity: 7600MB
[mmc]: boot0 capacity: 0KB,boot1 capacity:
[mmc]: ************SD/MMC 2 init OK!!!*****
```



Analysis of Devices

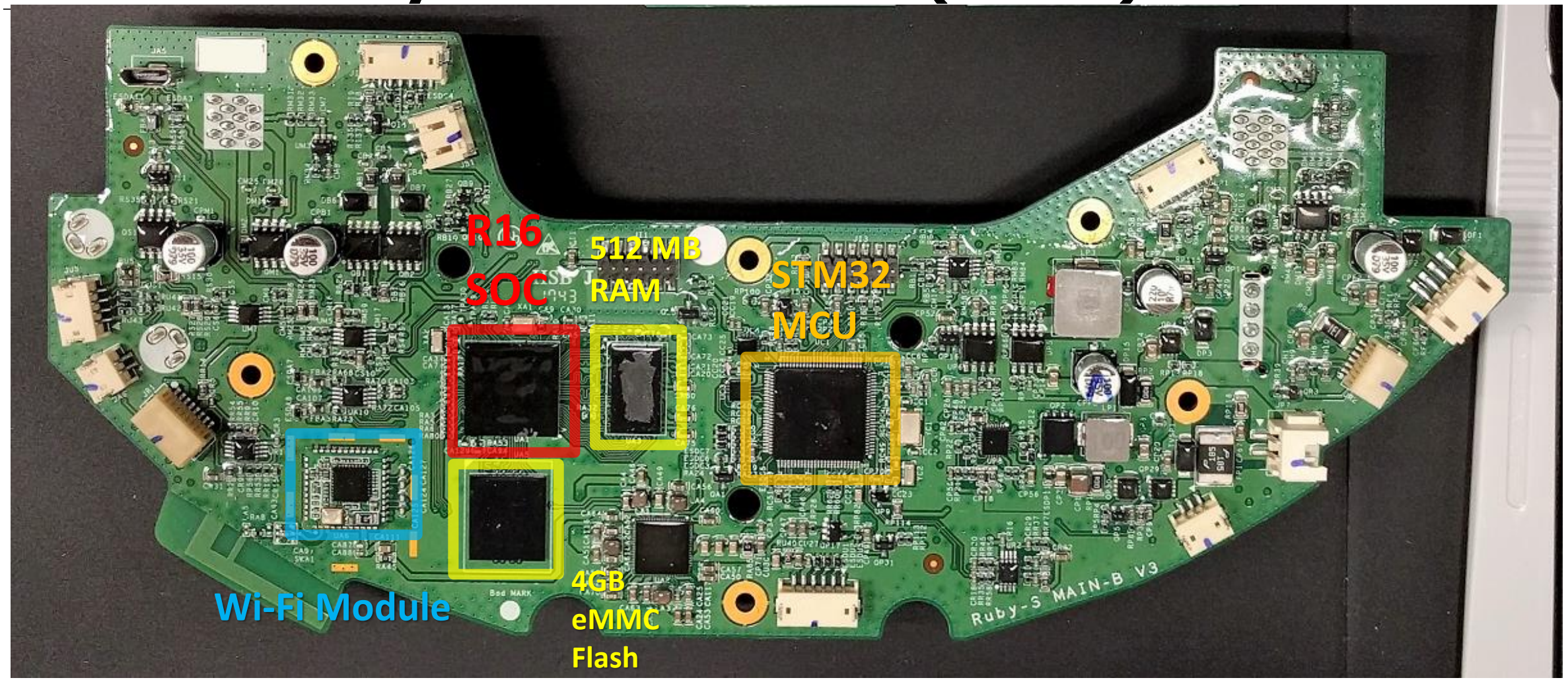
Roborock S50 (Gen2)

- Released 2018
- Same hardware and software base as Mi Robot Vacuum
  - Improvements in software
  - Supports mopping of floors
  - Small hardware modifications
- Same firmware keys as Gen1



Analysis of Devices

Frontside layout mainboard (Gen2)



Introduced Countermeasures in Gen2

- Encrypting/Obfuscating the log-files and maps
- RRlogd uses AES encryption functions from OpenSSL library
  - Imported as dynamic library
  - Interesting function: EVP_EncryptInit_ex(...)
  - Ltrace can be used to intercept calls and extract arguments

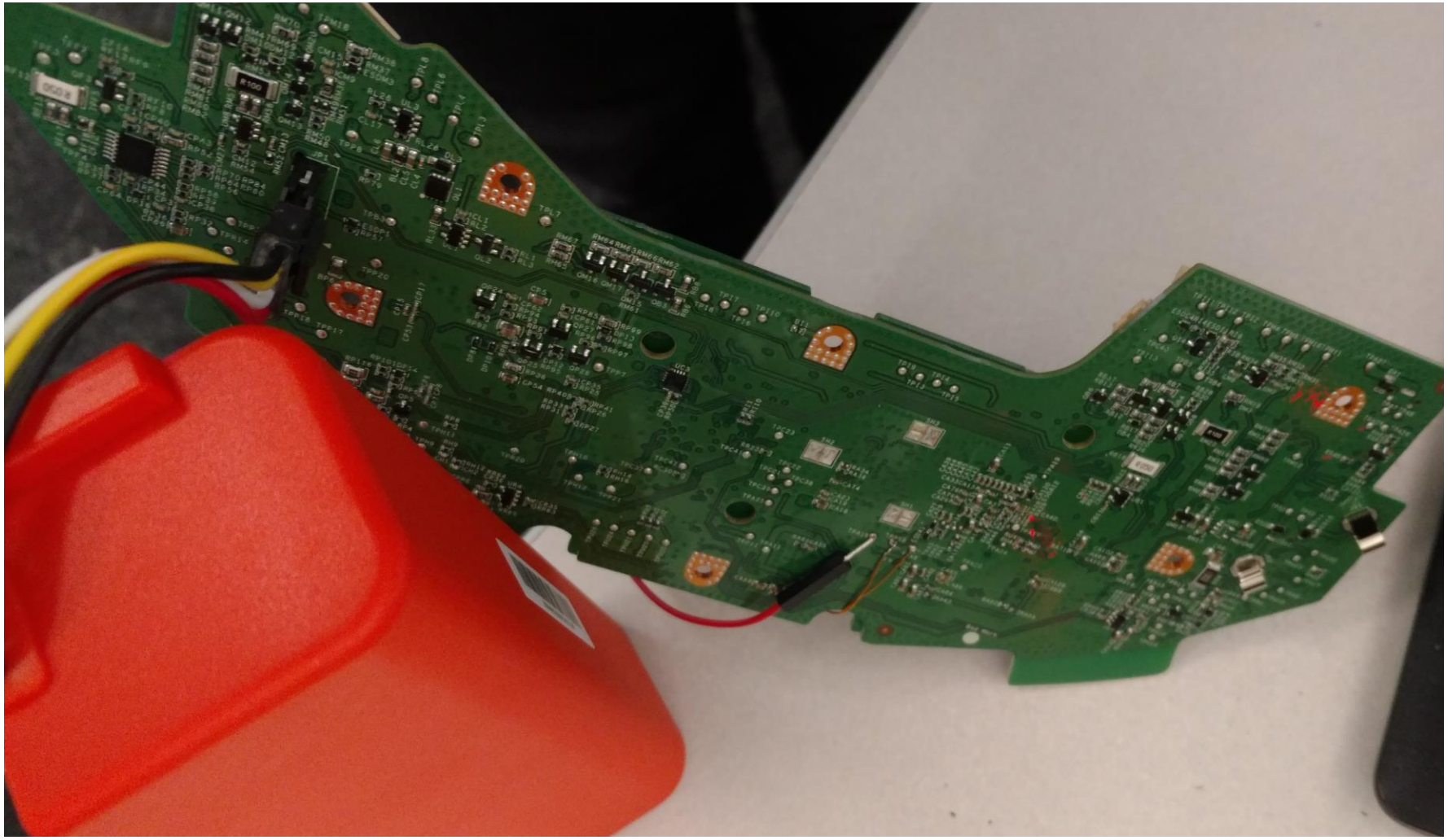
➢ Contribution: AES128CBC-key: “RoCKR0B0@BEIJING”, documentation of firmware, backporting features to Gen1

Analysis of Devices
Roborock S60/T60

- Released 2019
- Same hardware and software base as S50
  - Improvements in software
  - Supports multiple floors
  - Small hardware modifications
- Firmware keys were changed
- Local OTA updates are blocked
- Firmware and configuration is now signed
- Region lock enforced

Analysis of Devices

Roborock S60/T60 UART setup



Roborock S60/T60 UART setup

- Roborock did not fix a vulnerability in U-Boot
  - Root password derivation mechanism remained the same
  - Login over UART possible, however watchdog triggers
  - Watchdog can be disabled in a racing condition
- Firmware is now signed and encrypted
  - Encryption keys and signature public keys obfuscated

➢ Contribution: Extraction of new encryption keys, development of new rooting method, development of automatic tool

Analysis of Devices
All results

	Debug Interfaces			Firmware			Network		Physical		Data	
Device name	UART	JTAG/SWD	Telnet/SSH	Encrypted	Signed	Verified	HTTPS	Certificate checked	Tamper resistant	Tamper evident	User data not on device~	secure unprovisioning
Aqara Gateway (Homekit)	✓		×	×	×	✓	✓	✓	×	✓	×	×
Aqara Smart Home Gateway	✓		×	×	×	✓	✓	✓	×	✓	×	×
Aqara Smart IP Camera	✓	×	✓	×	×	✓	×	×	×	×	×	×
Lumi Smart Home Gateway	✓	✓	×	×	×	×	×	×	×	✓	✓	✓
Philips Ceiling Lamp	✓	✓	×	×	×	×	✓	×	×	✓	✓	✓
Roborock S50	✓		✓	✓	×	✓	✓	✓	×	✓	×	×
Roborock S6/T61	✓		✓	✓	✓	✓	✓	✓	×	✓	×	✓
Xiaomi Mi Vacuum Robot	✓		✓	✓	×	✓	✓	✓	×	✓	×	×
Xiaomi Mi WiFi Speaker	✓		×	×	×	✓	✓	✓	×	×	×	×
Xiaomi WiFi Plug	✓	✓	×	×	×	×	×	×	✓	✓	✓	✓
Yeelink Bedside lamp	✓	✓	×	×	×	×	✓	×	✓	✓	✓	✓
Yeelink Ceiling Lamp	✓	✓	×	×	×	×	×	×	×	×	✓	✓
Yeelink Light Color	✓	✓	×	×	×	×	×	×	✓	✓	✓	✓
Yeelink Light Mono1	✓	✓	×	×	×	×	×	×	✓	✓	✓	✓
Yeelink Light Strip	✓	✓	×	×	×	×	×	×	×	✓	✓	✓
Yeelink Smart White Bulb	✓	✓	×	×	×	×	×	×	✓	✓	✓	✓
Yeelink Smart RGB Bulb	✓	✓	×	×	×	×	×	×	✓	✓	✓	✓

Outline

- ✓ Introduction
- ✓ Methodology
- ✓ Analysis of Mi Home App
- ✓ Analysis of Devices
- ➢ Discussion
- Conclusion

Discussion
Mi Home App

- Xiaomi puts effort in securing the API and the APP
- Reported vulnerabilities were fixed
- Apps and Plugins are updated on a regular base
- However:
  - Functionality seems more important than security
  - Plugins by vendors introduce new risks
  - Historically grown ecosystem leaves many deprecated APIs
  - Too much trust in the security of the app, missing checks in the cloud

Discussion
Devices

- Xiaomi SDK enables secure communication with the cloud
  - confidentiality, integrity, and availability ensured by design (as long as device specific keys are not leaked)
- Implementations of vendors vary in quality, many contain vulnerabilities
- Vendors try to lock out users and try to restrict devices in a region
- User data is not stored securely, factory resets are not sufficiently done
- Unprovisioned devices are vulnerable due to missing firmware signature and verifications
  - Linux version: cannot detect if a OTA update is pushed from cloud or from local network
  - However: enables user to gain access on their own devices
- Developers lack knowledge of secure implementations of features
- Development time seems to be limited: many firmwares with debug symbols

Outline

- ✓ Introduction
- ✓ Methodology
- ✓ Analysis of Mi Home App
- ✓ Analysis of Devices
- ✓ Discussion
- ➢ Conclusion

Contributions

- Describing Mijia Ecosystem and API
- Analysis and documentation of many different devices
- Development and publication of rooting methods
  - Custom firmwares for all analyzed devices
- Analysis of data usage and life-cycle
- (For Cortex-M based devices: ported Nexmon framework to Marvel and Mediatek based Xiaomi devices)

Conclusion
Key findings

- Legacy API and design in Mi Home App enables unrestricted access
- Missing filtering and permission checking of commands in cloud
- Non Cortex-M devices leave sensitive information after factory reset
- Many devices do not implement HTTPS correctly
- Firmware signatures are rare
- Broken firmware verifications
- All devices have some kind of vulnerabilities
  - Enables user to take control over own device
  - Leaves risk of remote attackers
- In discussions with vendors: missing understanding for risks

Conclusion
Answering the research question

Research question: How secure is the implementation of the ecosystem of the IoT market leader Xiaomi?

- Mijia devices have less interfaces, therefore a smaller attack surface
- Xiaomi puts effort in security and privacy, but there are fundamental issues in the design of the app and APIs
- While the SDK is secure, the additional implementations of vendors introduce vulnerabilities
- Compared to other ecosystems in the same market segment, the implementations are more secure
- Security is often limited by pricing or knowledge constrains
- Rooting methods enable the users to verify security and privacy themselves

Future Work

- Analysis of new Xiaomi Vacuum robot
  - Uses camera
  - Trust Zone, Secure Boot, AVB, SE Linux, LUKS, encrypted RAM
  - Successful root was possible 1 week after submission of this thesis
- Analysis of new Cloud protocol
  - Will be introduced in November 2019
- Using the same methods for other big ecosystems


Questions?