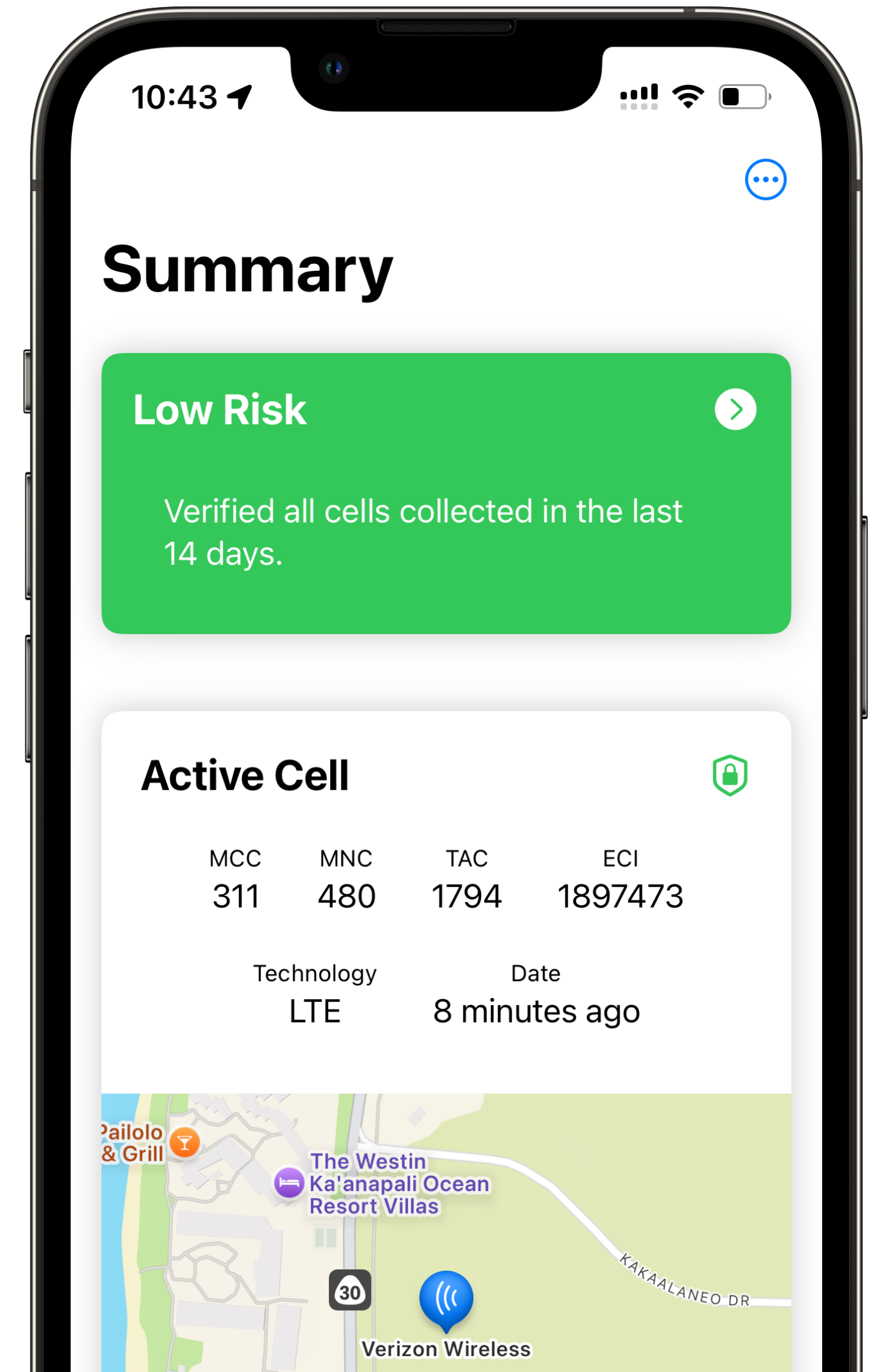




Trace the Base

Unraveling the iPhone's Baseband Architecture to Defend Against Cellular Attacks

Lukas Arnold - December 5th 2024



Basebands

Vulnerabilities every day

Multiple Internet to Baseband Remote Code Execution Vulnerabilities in Exynos Modems

Posted by Tim Willis, Project Zero

In late 2022 and early 2023, Project Zero reported eighteen 0-day vulnerabilities in Exynos Modems produced by Samsung Semiconductor. The four most severe of these eighteen vulnerabilities (CVE-2023-24033, CVE-2023-26496, CVE-2023-26497 and CVE-2023-26498) allowed for Internet-to-baseband remote code execution. Tests conducted by Project Zero confirm that those four vulnerabilities allow an attacker to remotely compromise a phone at the baseband level with no user interaction, and require only that the attacker know the victim's phone number. With limited additional research and development, we believe that skilled attackers would be able to quickly create an operational exploit to compromise affected devices silently and remotely.

CVE-2024-43047

The Register

Qualcomm urges device makers to push patches after 'targeted' exploitation

Given Amnesty's involvement, it's a safe bet spyware is in play

Iain Thomson

Tue 8 Oct 2024 // 21:30 UTC



whoami

lukasarnold

- Hi I'm Lukas 🙋
- Master's Student @ TU Darmstadt
- Student Researcher @ SEEMOO
- Thanks to the Objective-See Foundation 🌊



TECHNISCHE
UNIVERSITÄT
DARMSTADT

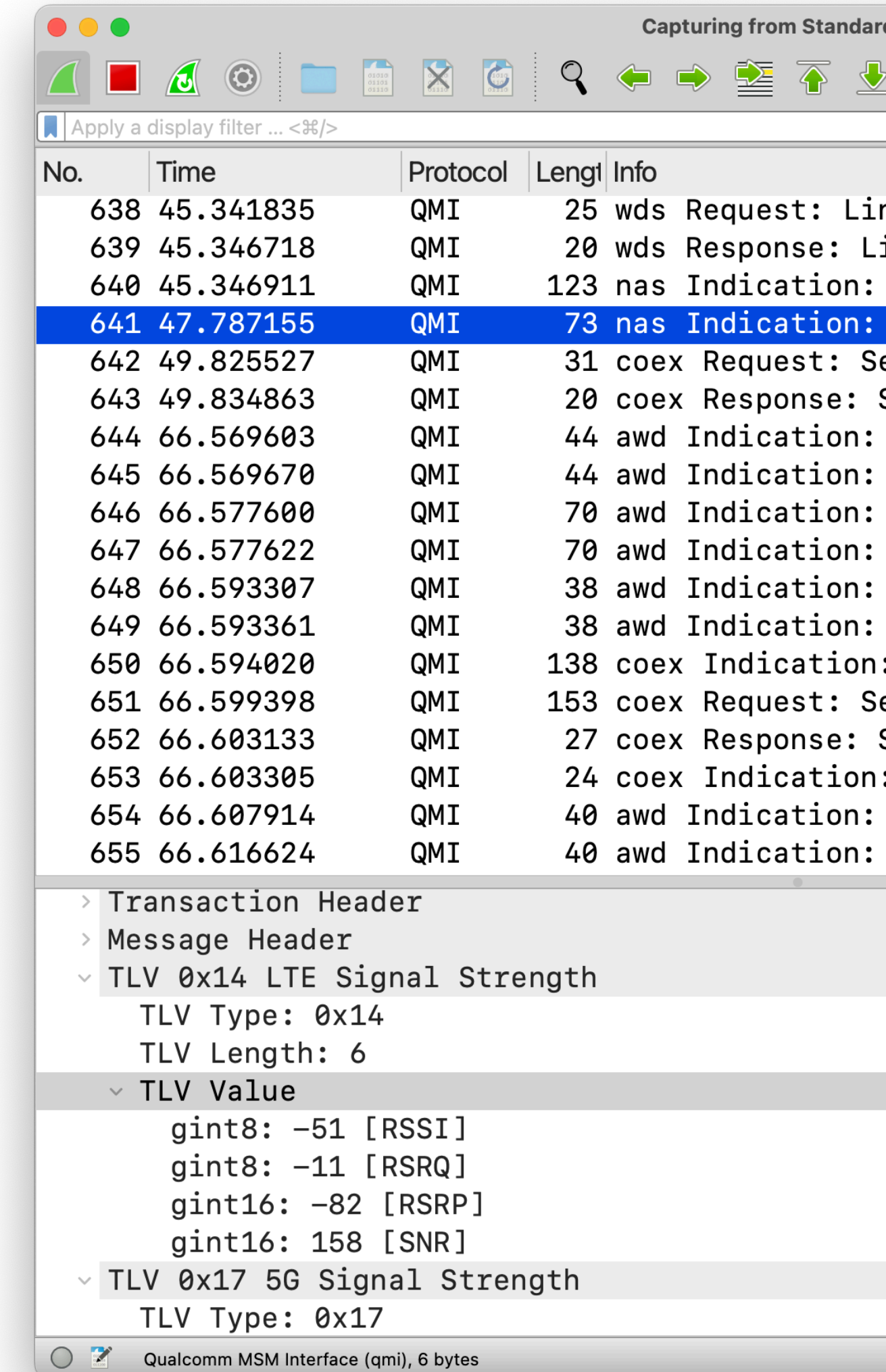


BaseTrace

iOS Cellular Security Analysis Framework



- Inspect and modify various layers of the iOS cellular protocol stack
 - iOS-Baseband communication (QMI)
 - Apple Wireless Diagnostics
 - Qualcomm DIAG for iPhones
- GitHub: [seemoo-lab/BaseTrace](https://github.com/seemoo-lab/BaseTrace)



No.	Time	Protocol	Length	Info
638	45.341835	QMI	25	wds Request: Lin
639	45.346718	QMI	20	wds Response: Li
640	45.346911	QMI	123	nas Indication:
641	47.787155	QMI	73	nas Indication:
642	49.825527	QMI	31	coex Request: Se
643	49.834863	QMI	20	coex Response: S
644	66.569603	QMI	44	awd Indication:
645	66.569670	QMI	44	awd Indication:
646	66.577600	QMI	70	awd Indication:
647	66.577622	QMI	70	awd Indication:
648	66.593307	QMI	38	awd Indication:
649	66.593361	QMI	38	awd Indication:
650	66.594020	QMI	138	coex Indication:
651	66.599398	QMI	153	coex Request: Se
652	66.603133	QMI	27	coex Response: S
653	66.603305	QMI	24	coex Indication:
654	66.607914	QMI	40	awd Indication:
655	66.616624	QMI	40	awd Indication:

> Transaction Header
> Message Header
v TLV 0x14 LTE Signal Strength
TLV Type: 0x14
TLV Length: 6
v TLV Value
gint8: -51 [RSSI]
gint8: -11 [RSRQ]
gint16: -82 [RSRP]
gint16: 158 [SNR]
v TLV 0x17 5G Signal Strength
TLV Type: 0x17

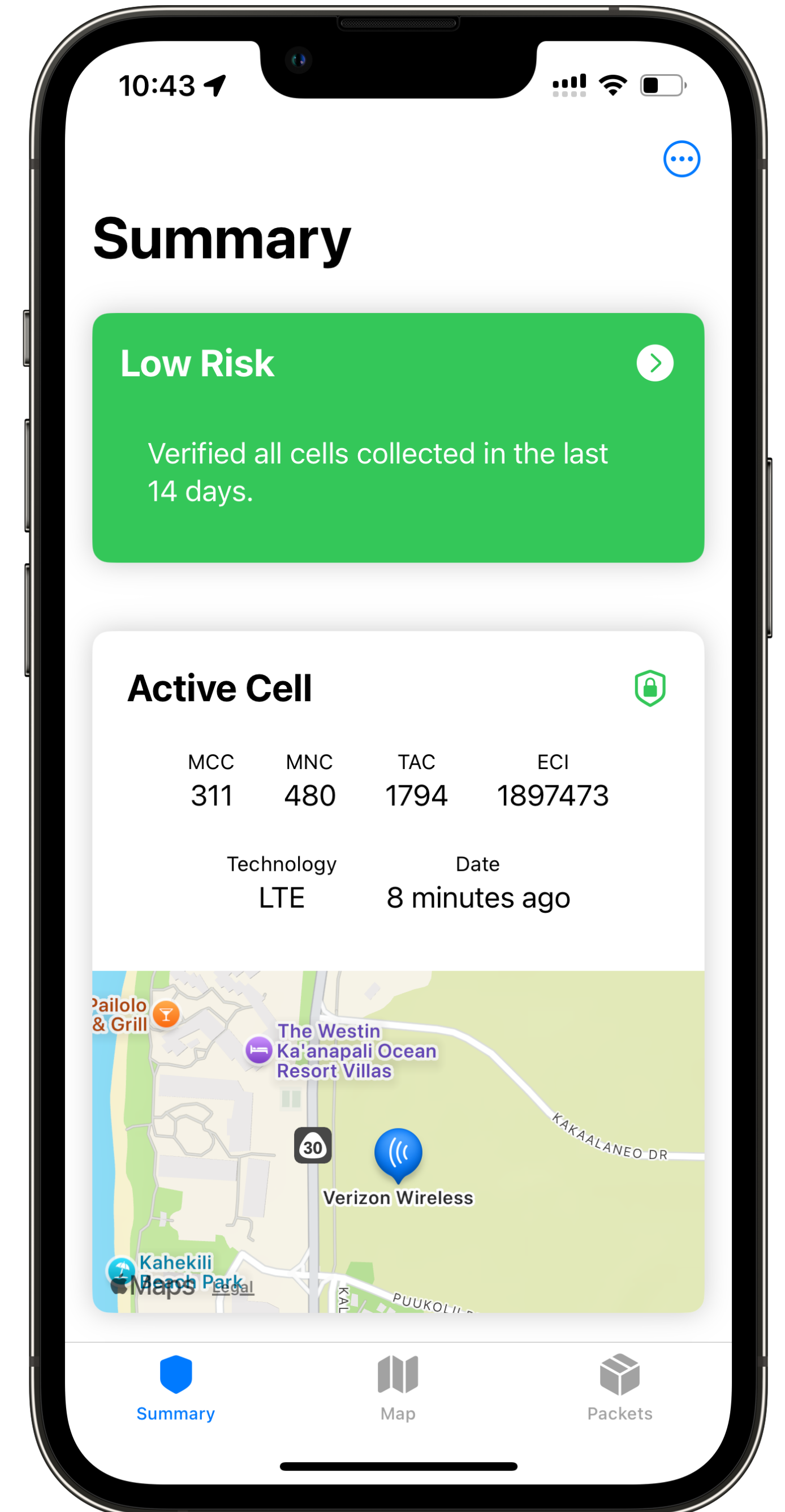
Qualcomm MSM Interface (qmi), 6 bytes

CellGuard

iOS Cellular Security Analysis App

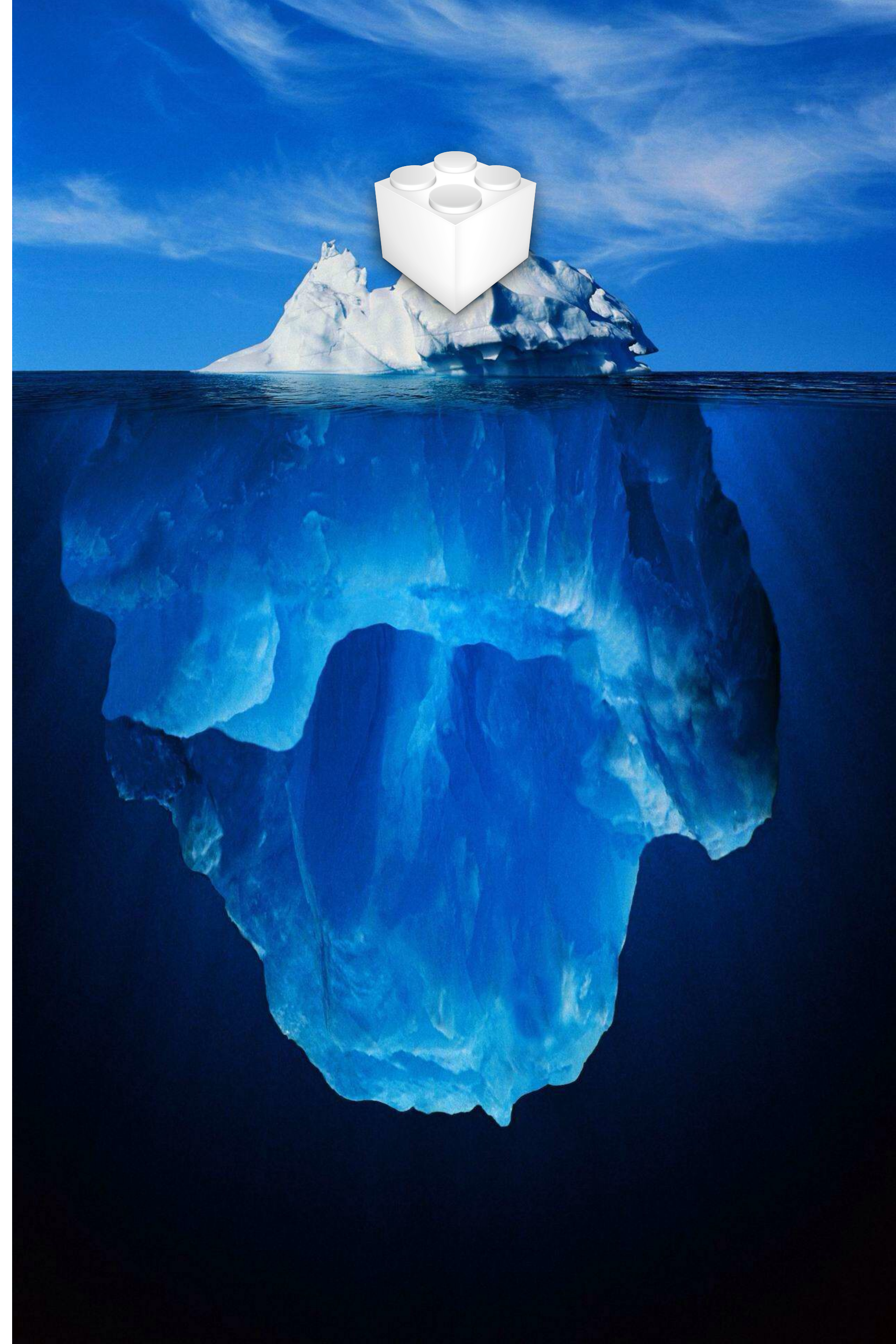


- Trace the cells to which your iPhone connects
- Monitor their parameters for suspicious activity
- Works on every iPhone (iOS 14 - iOS 18)
- GitHub: [seemoo-lab/CellGuard](https://github.com/seemoo-lab/CellGuard)





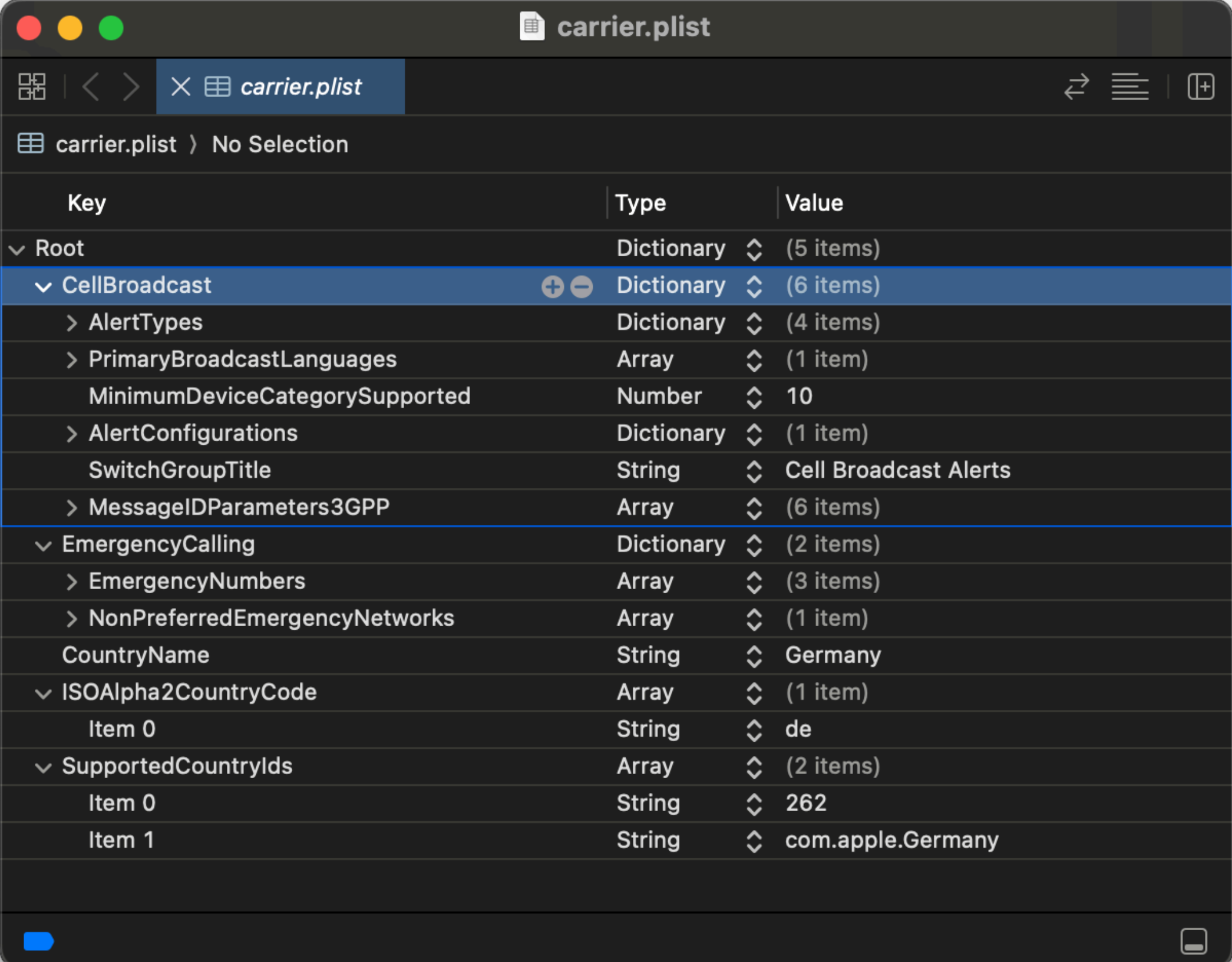
BaseTrace Cellular Bundles



Country Bundles

/System/Library/CountryBundles/iPhone

- General settings for each country
 - Cell broadcast, emergencies, ...
- Delivered with iOS firmware
- iOS queries for out-of-order updates
 - <https://s.mzstatic.com/version>



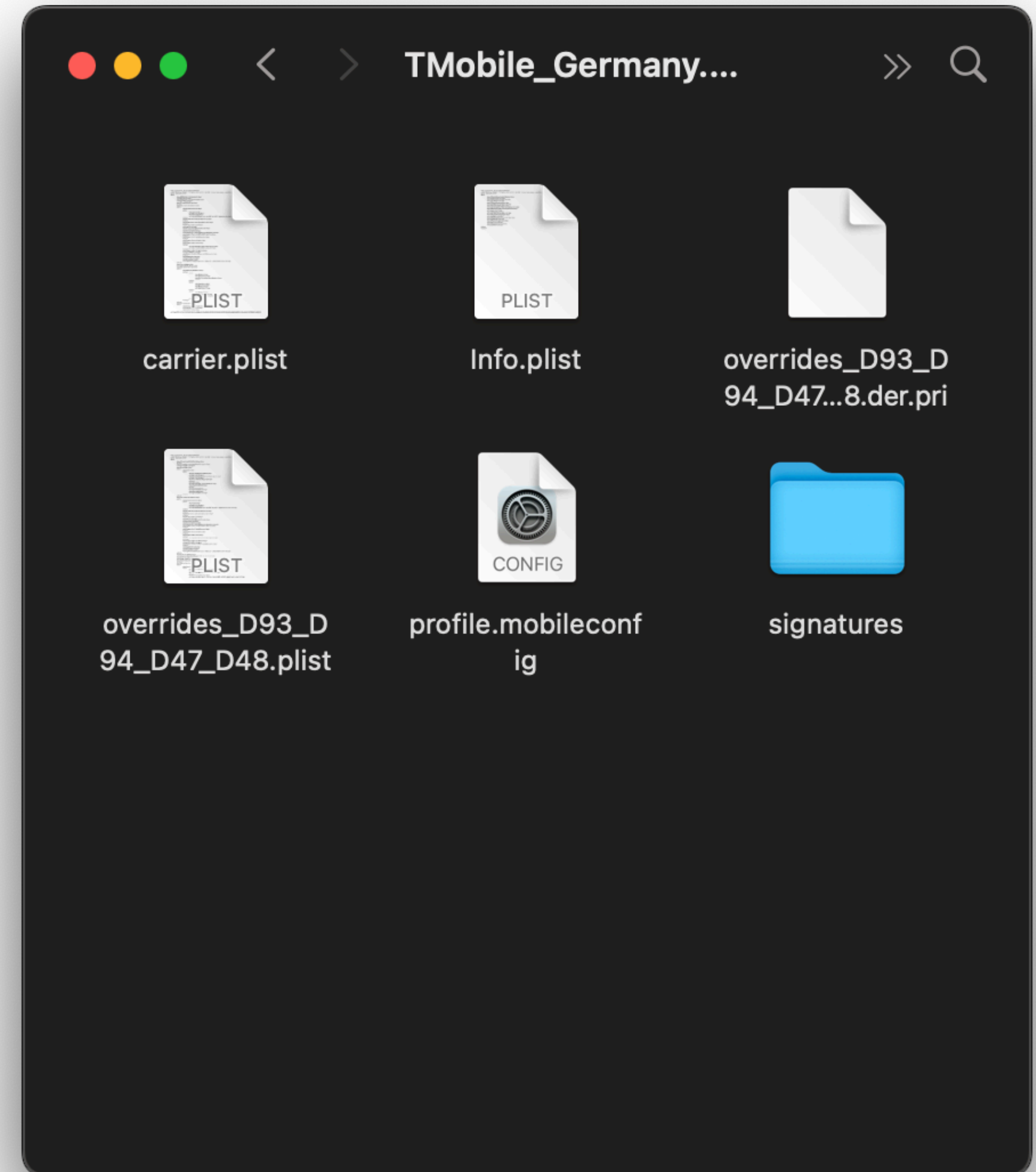
The screenshot shows a plist file viewer displaying the structure of a carrier.plist file. The file is organized into a hierarchy of dictionaries and arrays. The 'CellBroadcast' dictionary is expanded, showing its sub-entries. The 'EmergencyCalling' dictionary is also expanded, showing its sub-entries. The 'CountryName' is 'Germany', the 'ISOAlpha2CountryCode' is 'de', and the 'SupportedCountryIds' array contains two items: '262' and 'com.apple.Germany'.

Key	Type	Value
Root	Dictionary	(5 items)
CellBroadcast	Dictionary	(6 items)
AlertTypes	Dictionary	(4 items)
PrimaryBroadcastLanguages	Array	(1 item)
MinimumDeviceCategorySupported	Number	10
AlertConfigurations	Dictionary	(1 item)
SwitchGroupTitle	String	Cell Broadcast Alerts
MessageIDParameters3GPP	Array	(6 items)
EmergencyCalling	Dictionary	(2 items)
EmergencyNumbers	Array	(3 items)
NonPreferredEmergencyNetworks	Array	(1 item)
CountryName	String	Germany
ISOAlpha2CountryCode	Array	(1 item)
Item 0	String	de
SupportedCountryIds	Array	(2 items)
Item 0	String	262
Item 1	String	com.apple.Germany

Carrier Bundles

/System/Library/Carrier Bundles/iPhone

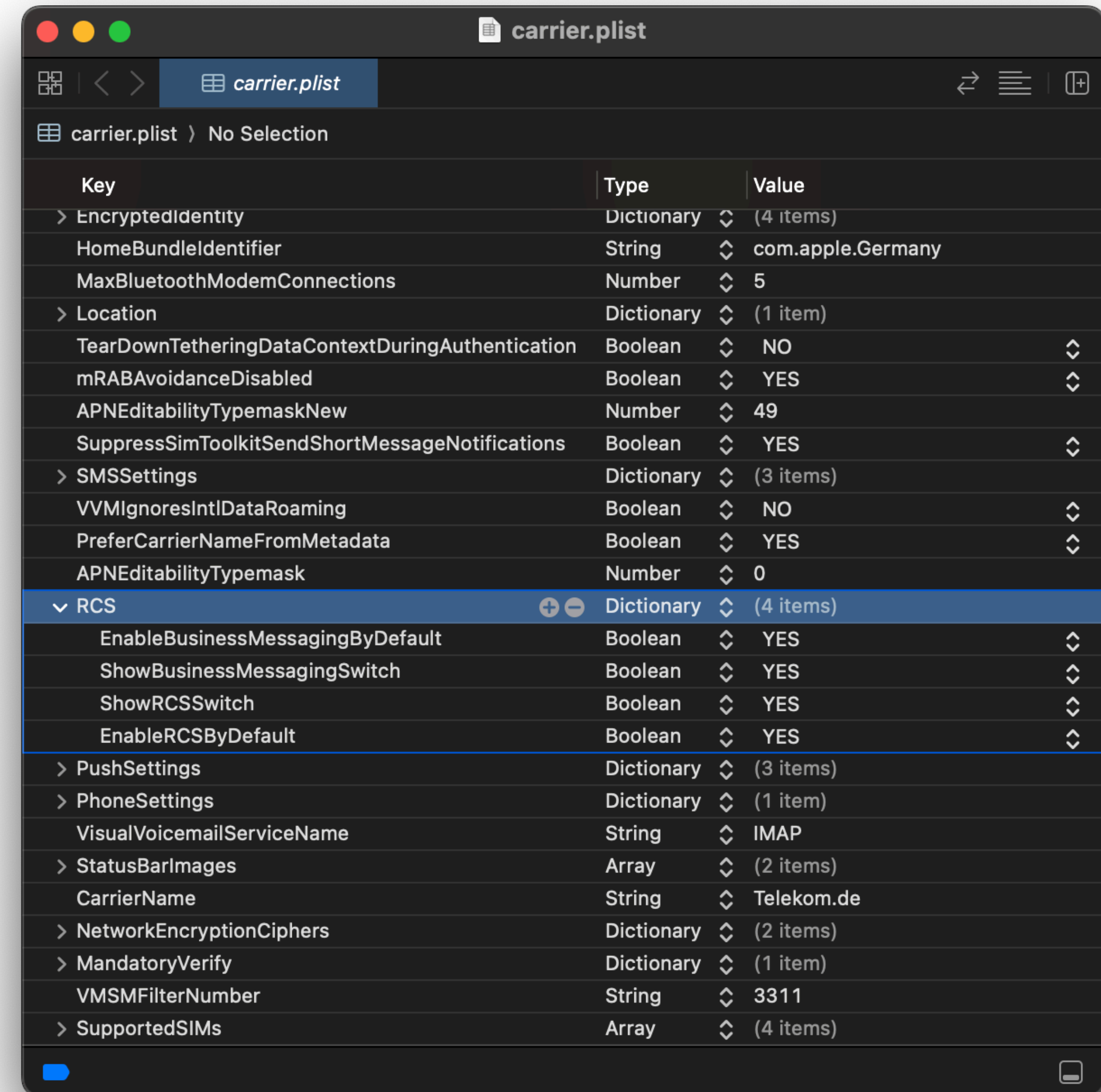
- Same delivery methods as country bundles
- Specific configuration for each operator



Carrier Bundles

/System/Library/Carrier Bundles/iPhone

- Same delivery methods as country bundles
- Specific configuration for each operator
 - 5G Standalone - 50 of 692 bundles
 - RCS support - 51 of 692 bundles
- Carrier WiFi
- ...

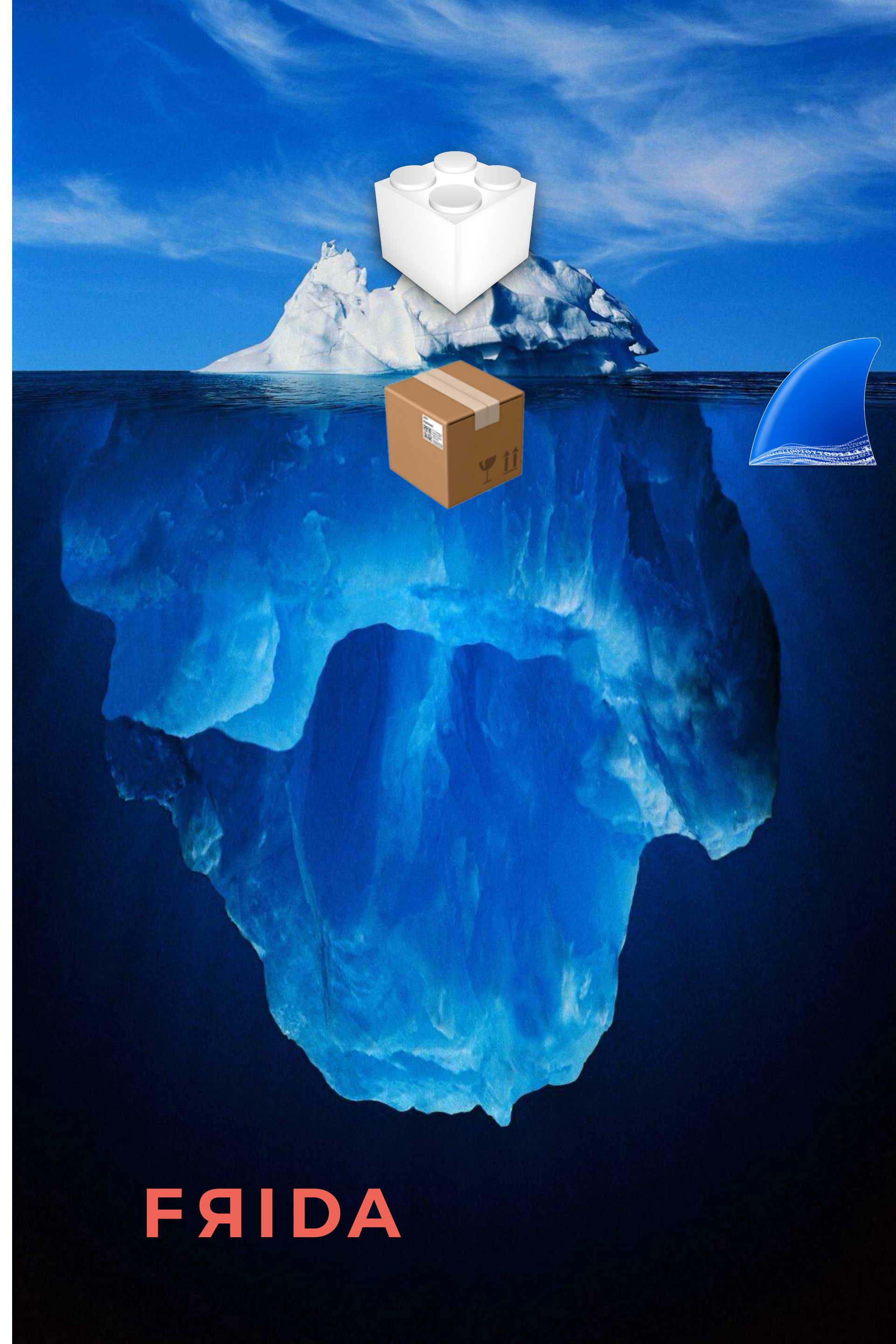


The screenshot shows a plist file viewer for 'carrier.plist'. The file is expanded to show the 'RCS' dictionary, which contains four boolean keys: 'EnableBusinessMessagingByDefault', 'ShowBusinessMessagingSwitch', 'ShowRCSSwitch', and 'EnableRCSByDefault', all set to 'YES'. Other keys in the file include 'EncryptedIdentity', 'HomeBundleIdentifier', 'MaxBluetoothModemConnections', 'Location', 'TearDownTetheringDataContextDuringAuthentication', 'mRABAvoidanceDisabled', 'APNEditabilityTypemaskNew', 'SuppressSimToolkitSendShortMessageNotifications', 'SMSSettings', 'VVMIgnoresIntlDataRoaming', 'PreferCarrierNameFromMetadata', 'APNEditabilityTypemask', 'PushSettings', 'PhoneSettings', 'VisualVoicemailServiceName', 'StatusBarImages', 'CarrierName', 'NetworkEncryptionCiphers', 'MandatoryVerify', 'VMSMFilterNumber', and 'SupportedSIMs'.

Key	Type	Value
> EncryptedIdentity	Dictionary	(4 items)
HomeBundleIdentifier	String	com.apple.Germany
MaxBluetoothModemConnections	Number	5
> Location	Dictionary	(1 item)
TearDownTetheringDataContextDuringAuthentication	Boolean	NO
mRABAvoidanceDisabled	Boolean	YES
APNEditabilityTypemaskNew	Number	49
SuppressSimToolkitSendShortMessageNotifications	Boolean	YES
> SMSSettings	Dictionary	(3 items)
VVMIgnoresIntlDataRoaming	Boolean	NO
PreferCarrierNameFromMetadata	Boolean	YES
APNEditabilityTypemask	Number	0
▼ RCS	Dictionary	(4 items)
EnableBusinessMessagingByDefault	Boolean	YES
ShowBusinessMessagingSwitch	Boolean	YES
ShowRCSSwitch	Boolean	YES
EnableRCSByDefault	Boolean	YES
> PushSettings	Dictionary	(3 items)
> PhoneSettings	Dictionary	(1 item)
VisualVoicemailServiceName	String	IMAP
> StatusBarImages	Array	(2 items)
CarrierName	String	Telekom.de
> NetworkEncryptionCiphers	Dictionary	(2 items)
> MandatoryVerify	Dictionary	(1 item)
VMSMFilterNumber	String	3311
> SupportedSIMs	Array	(4 items)

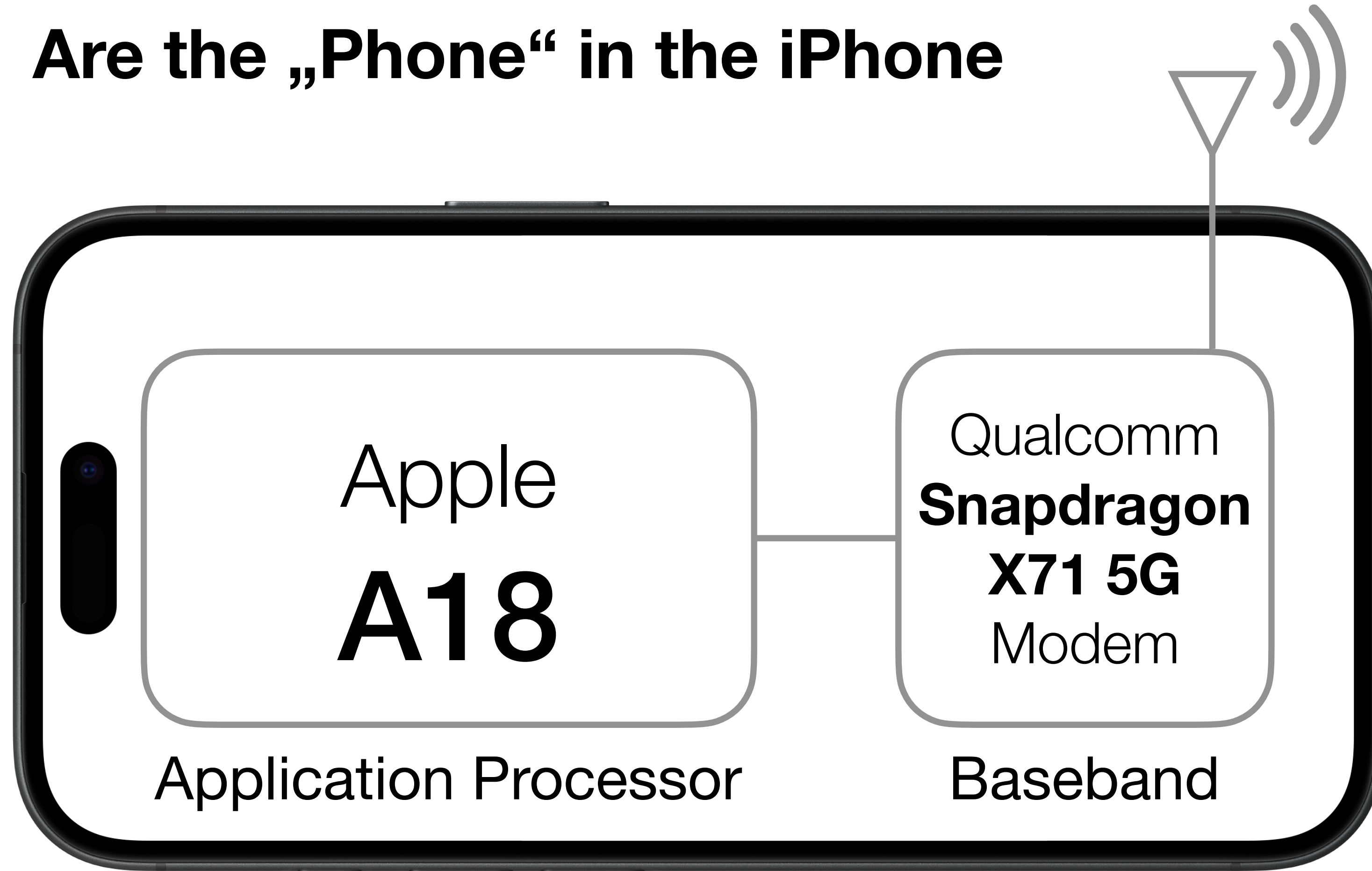


BaseTrace Baseband Protocols



iPhone Basebands

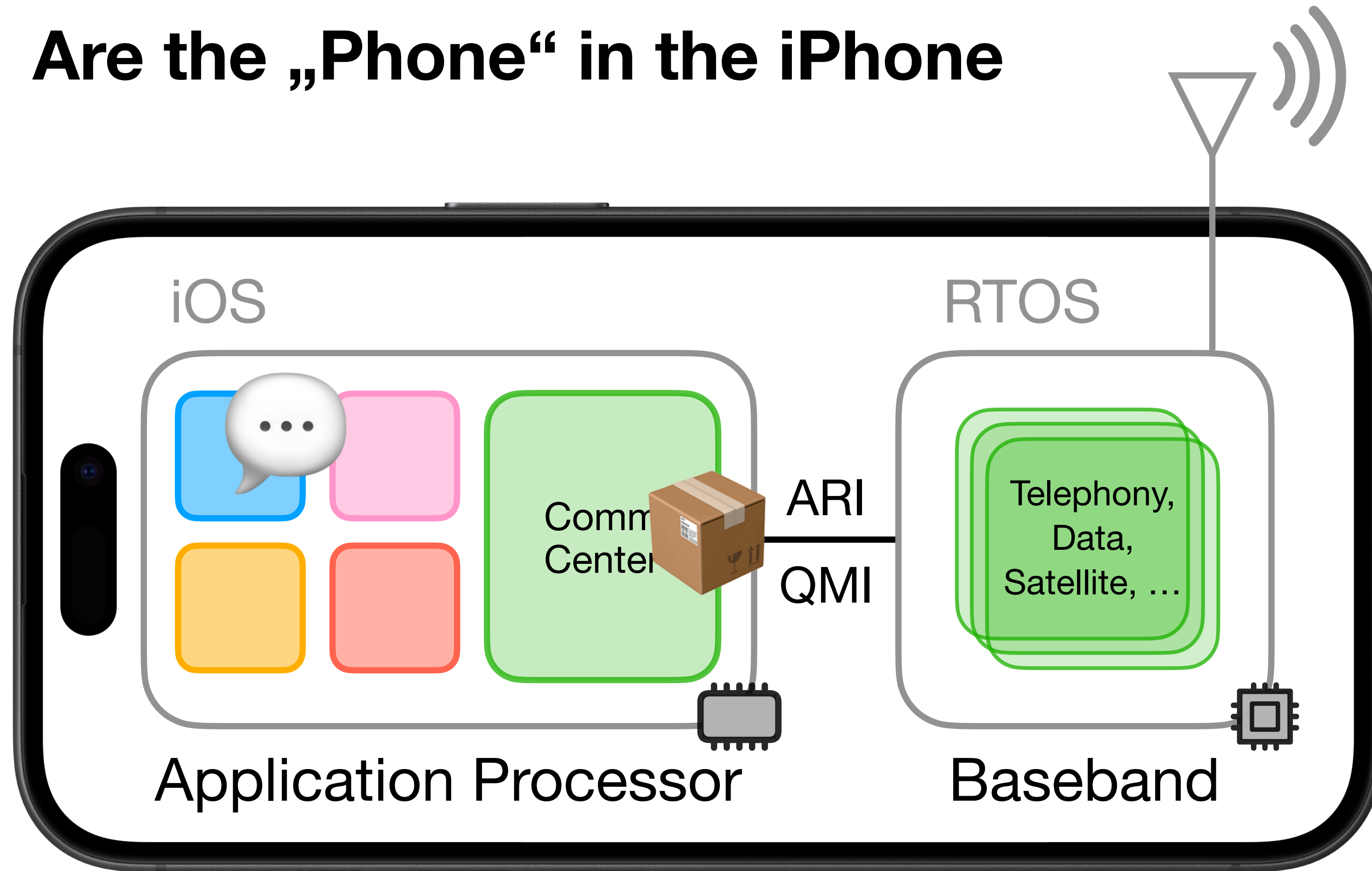
Are the „Phone“ in the iPhone



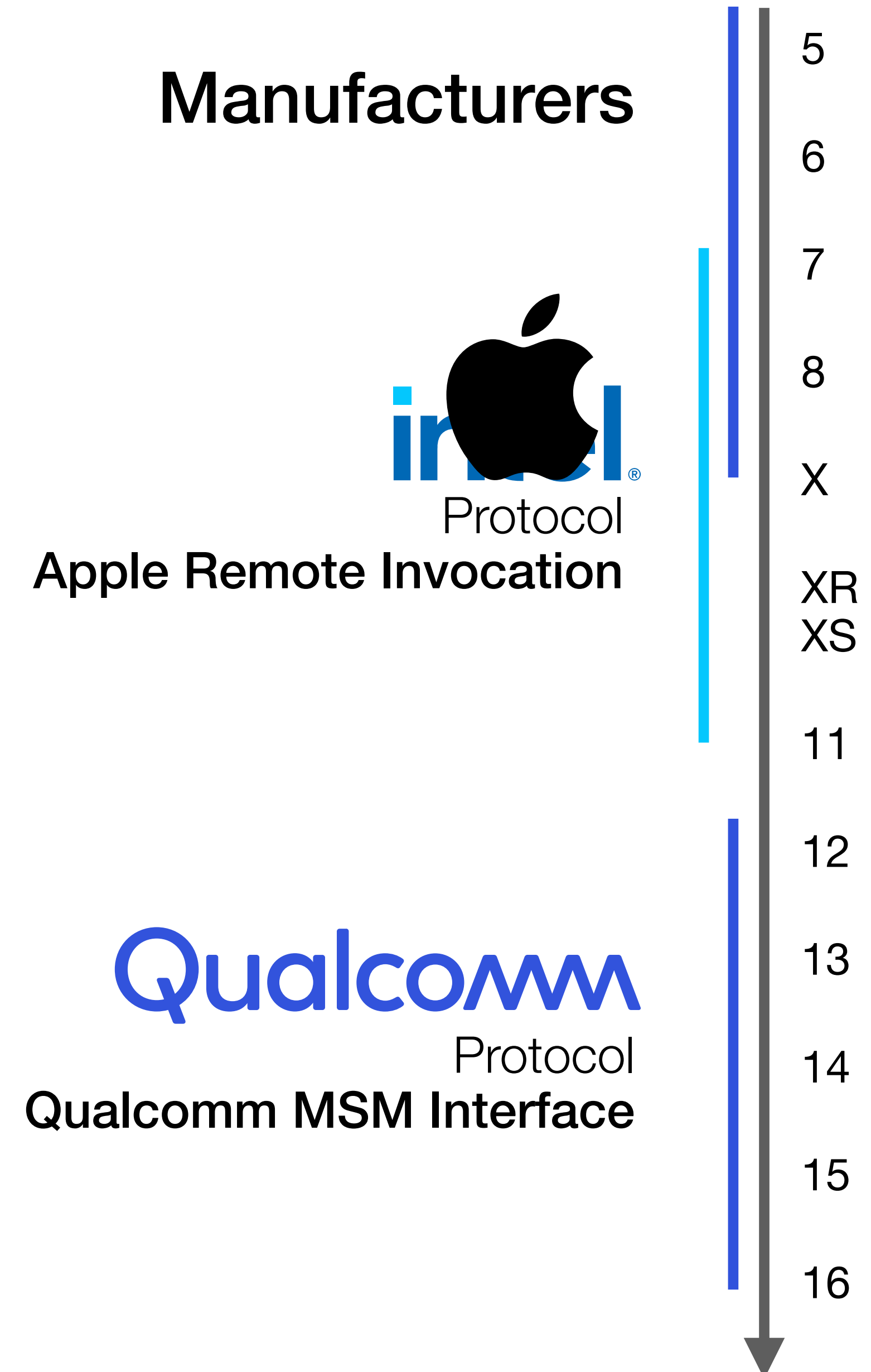
iPhone 16

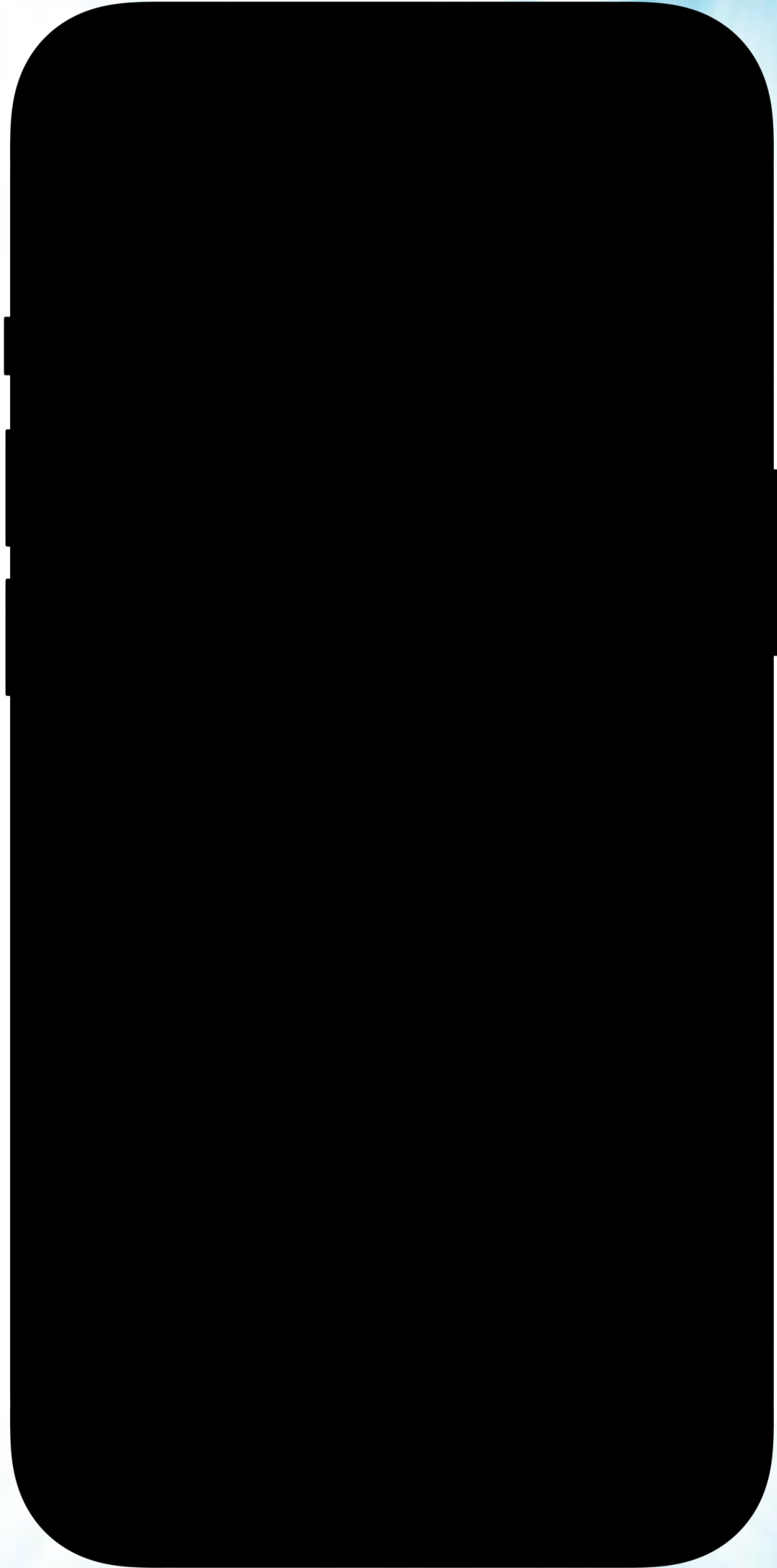
iPhone Basebands

Are the „Phone“ in the iPhone



Basebands provide a **packet-based interface** for the OS

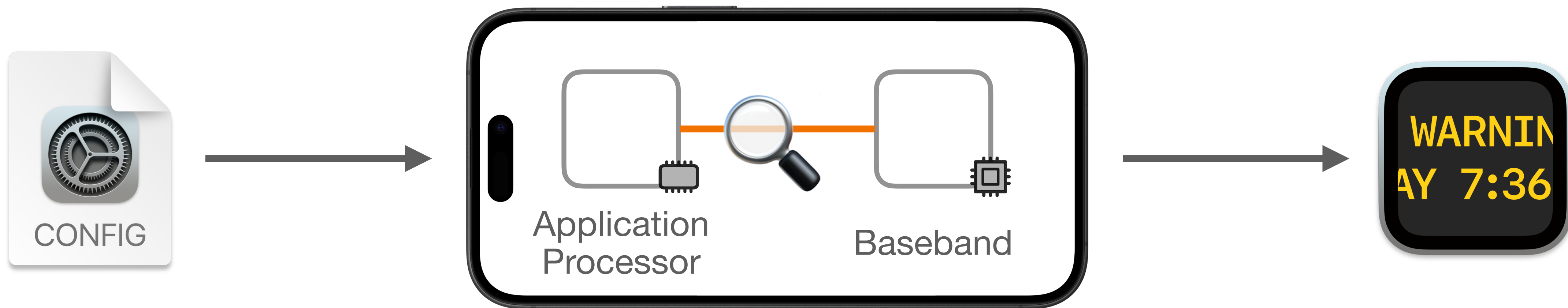




POKÉMON

Baseband Debug Profile

Retrieve raw AP-baseband communication



```
QMI: Svc=0x03(NAS) Req MsgId=0x004f Sim=1 Bin=['01 0C 00 00 03 0B 00 AE 06 4F...
QMI: Svc=0x03(NAS) Req MsgId=0x006c Sim=1 Bin=['01 60 00 00 03 0B 00 AF 06 6C...
QMI: Svc=0xe7(ELQM) Req MsgId=0x0003 Bin=['01 3F 00 00 E7 01 00 5E 0F 03 00 33...
QMI: Svc=0xe7(ELQM) Req MsgId=0x0003 Bin=['01 3F 00 00 E7 01 00 5F 0F 03 00 33...
QMI: Svc=0xe4(AWD) Ind MsgId=0x1012 Bin=['01 2B 00 80 E4 01 04 16 B4 12 10 1F...
QMI: Svc=0xe4(AWD) Ind MsgId=0x1012 Bin=['01 2B 00 80 E4 02 04 16 B4 12 10 1F...
QMI: Svc=0xe4(AWD) Ind MsgId=0x1010 Bin=['01 45 00 80 E4 01 04 17 B4 10 10 39...
```



Apple Remote Invocation

Apple's protocol for Intel basebands

ARISTOTELES

seemoo-lab/aristoteles

- Exclusively used by Intel basebands in iPhone 7 up to iPhone 11
- Novel protocol reverse engineered by Tobias Kröll
- Automatic recovery of structure information from libARI.dylib 🥰
- Based on asString functions and data structures

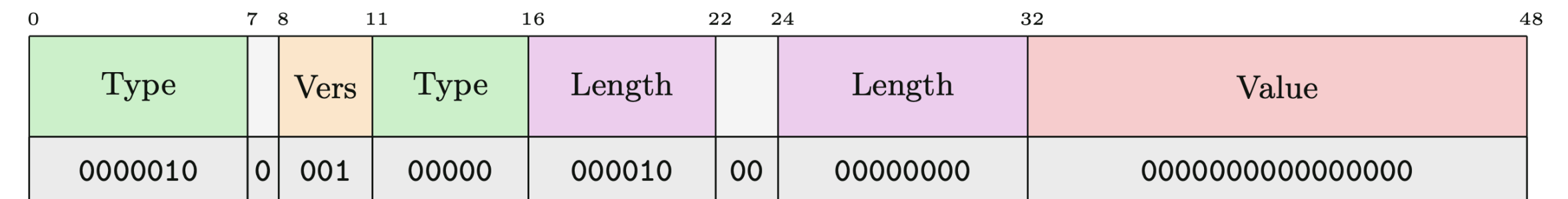
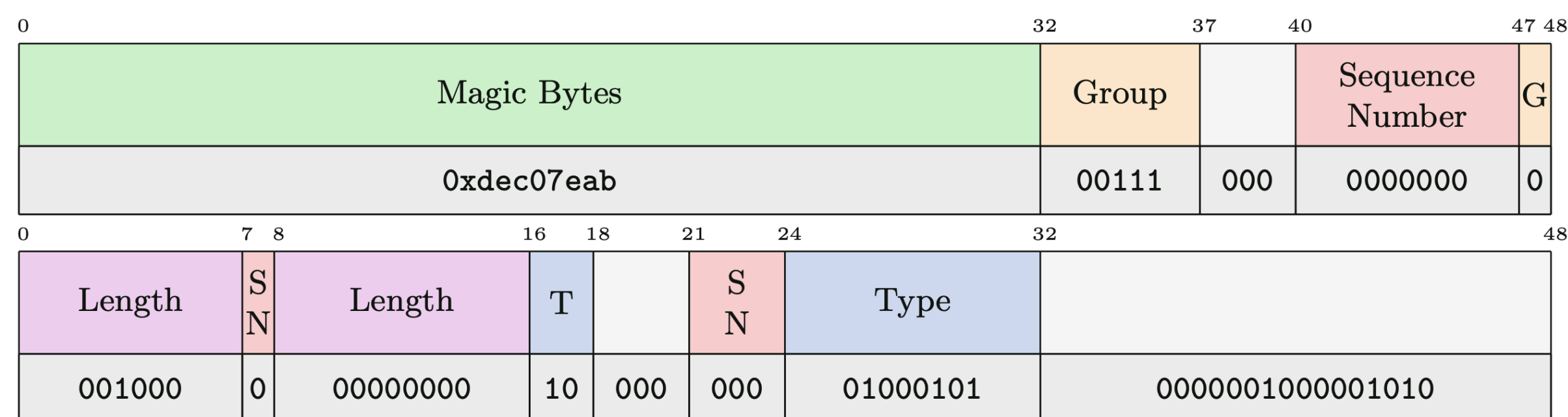


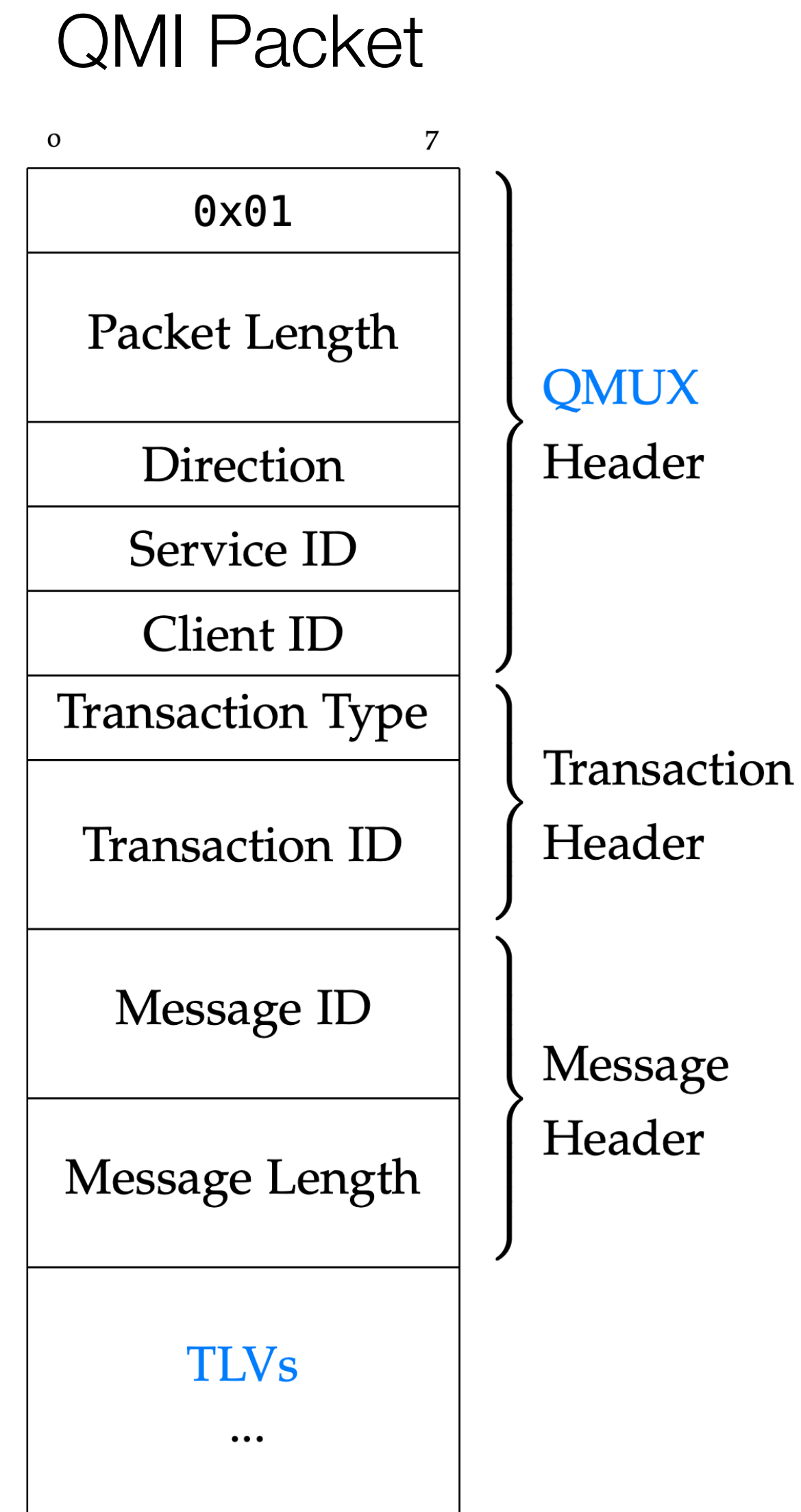
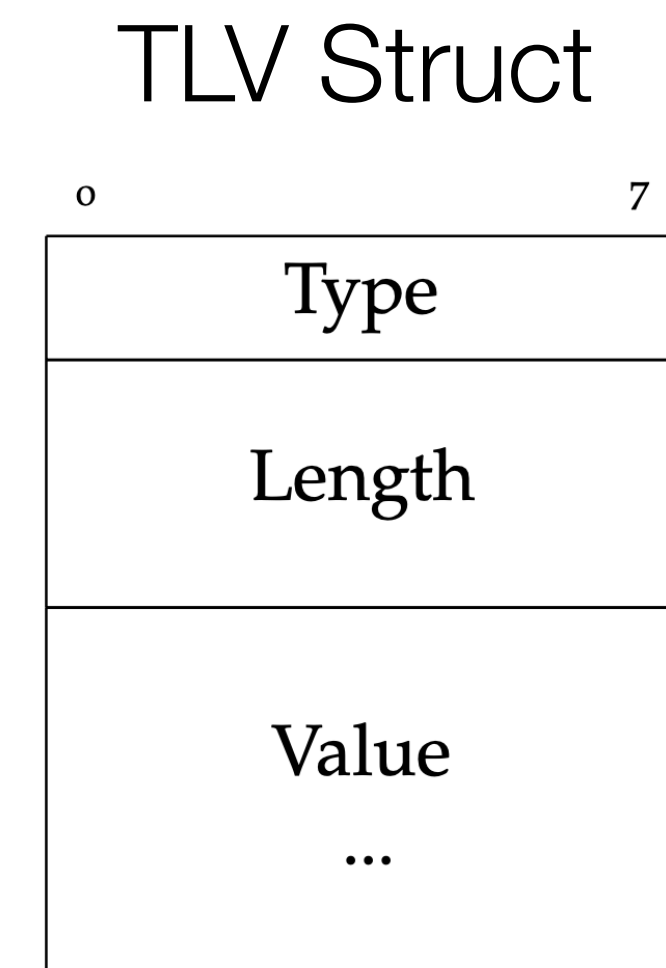
Fig. 3. ARI TLV format with example payloads.

Fig. 2. ARI header format, 12 B split into according bits with example values.

Qualcomm MSM Interface

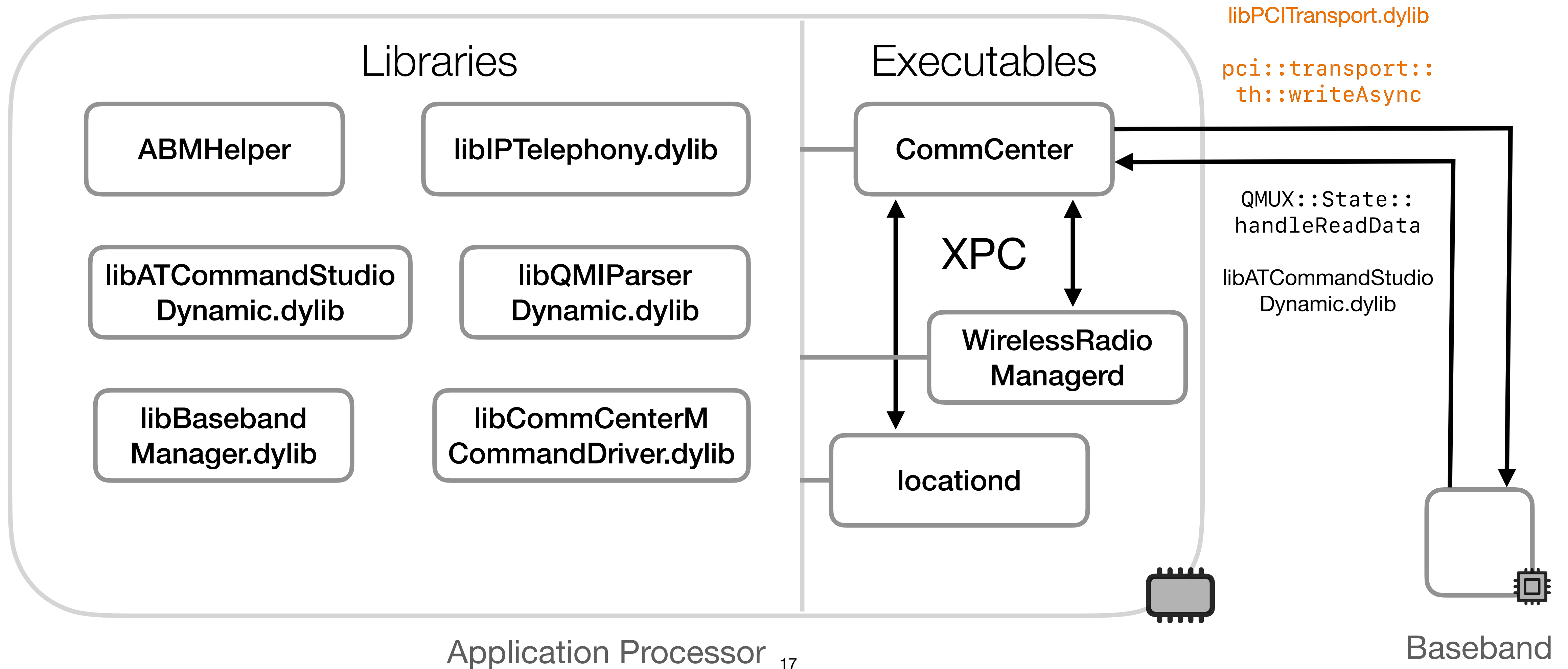
Qualcomm's baseband protocol

- Used for iPhones, Android, USB Modems, ...
- Existing open-source projects, e.g. libqmi
- Mostly iOS-custom service and message types
- Parsers for packets and TLVs spread across many executables and libraries 😞



iOS Baseband Packet Processing

The long journey of QMI packets



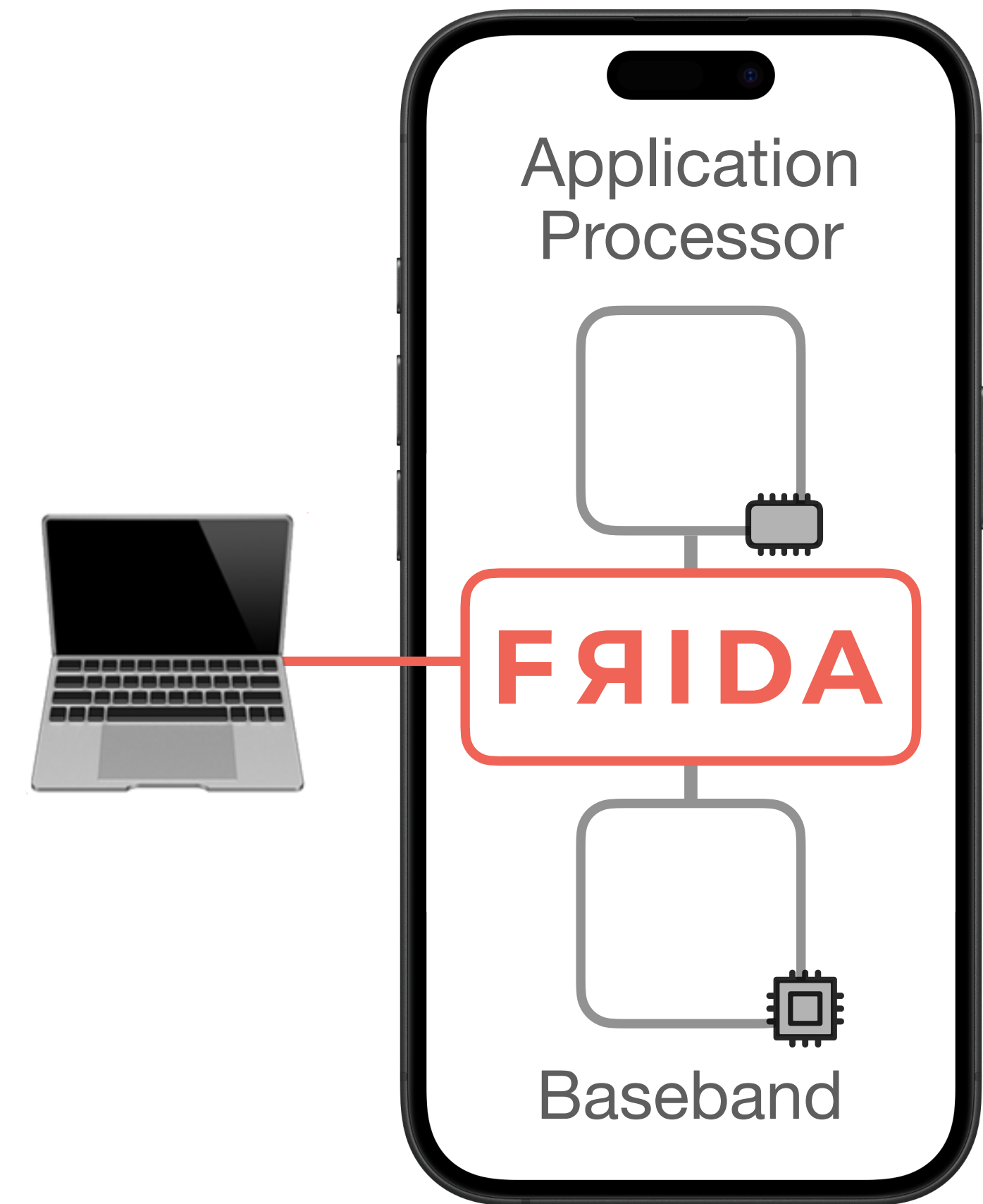
Packet Injection

Directly communicate with the baseband

- Based on FRIDA, thus requires jailbroken iPhone
- Bypass iOS QMI packet processing

```
lukas@debian-thesis: ~/iphone-qmi-glue ㉿ #1
iproxy #1  lukas@debian-thesis:... #2  lukas@debian-thesis:... #3 +
lukas@debian-thesis:~/iphone-qmi-glue$ qmicli -d ./qmux_socket
--get-service-version-info
[/home/lukas/iphone-qmi-glue/qmux_socket] Supported versions:
  ctl (1.5)
  wds (1.177)
  dms (1.79)
  nas (1.25)
  qos (1.17)
  wms (1.10)
  pds (1.18)
  auth (1.14)
  at (1.6)
  voice (2.1)
  cat2 (2.24)
  uim (1.77)
  pbm (1.4)
```

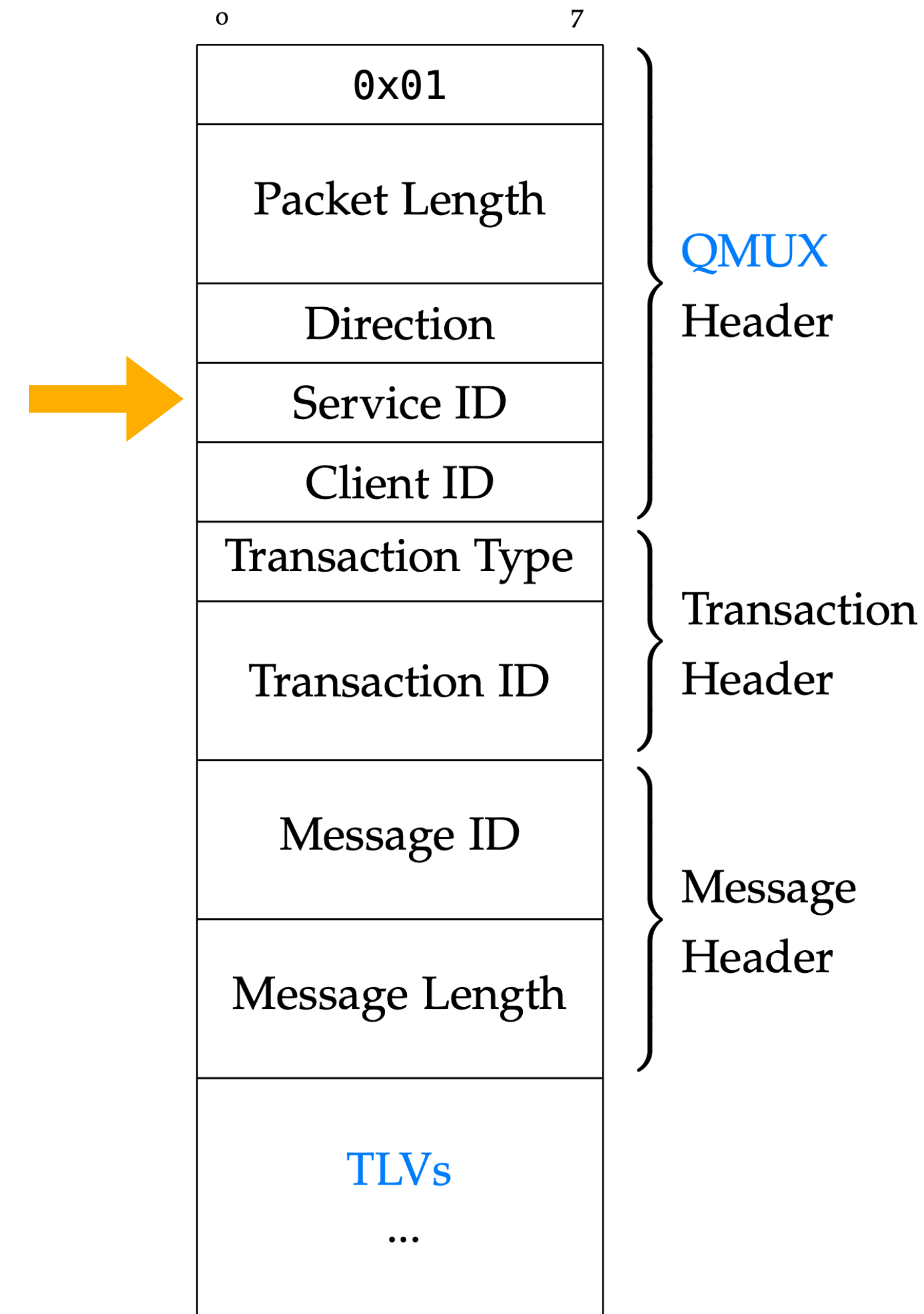
succeed



Symbolicate Packets

Simple in theory, hard to automate 🤖

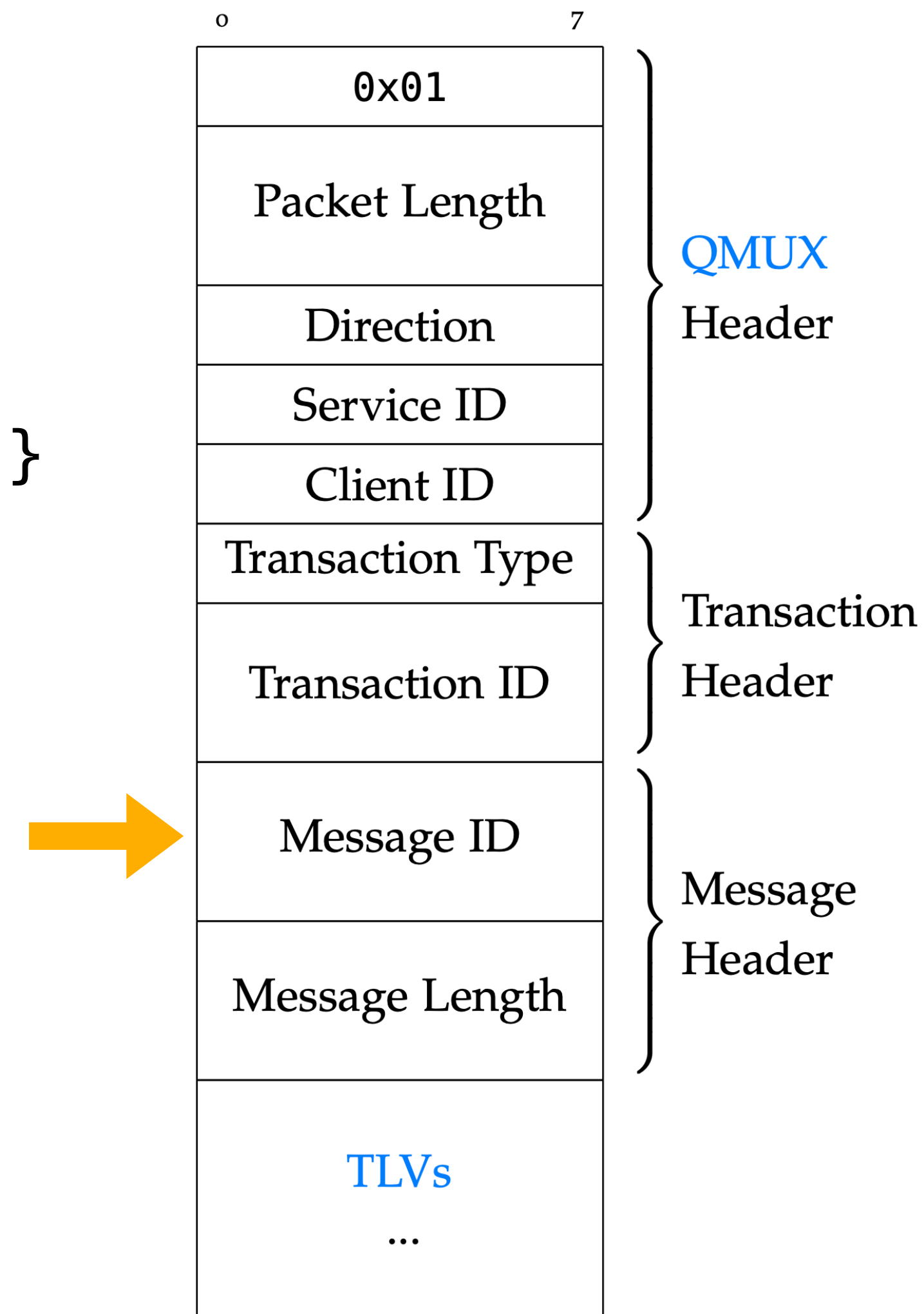
- Services – `as{Short, Long}String`



Symbolicate Packets

Simple in theory, hard to automate 🤖

- Services – `as{Short, Long}String`
- Messages – `MessageBase::{MessageBase, validateMsgId}`



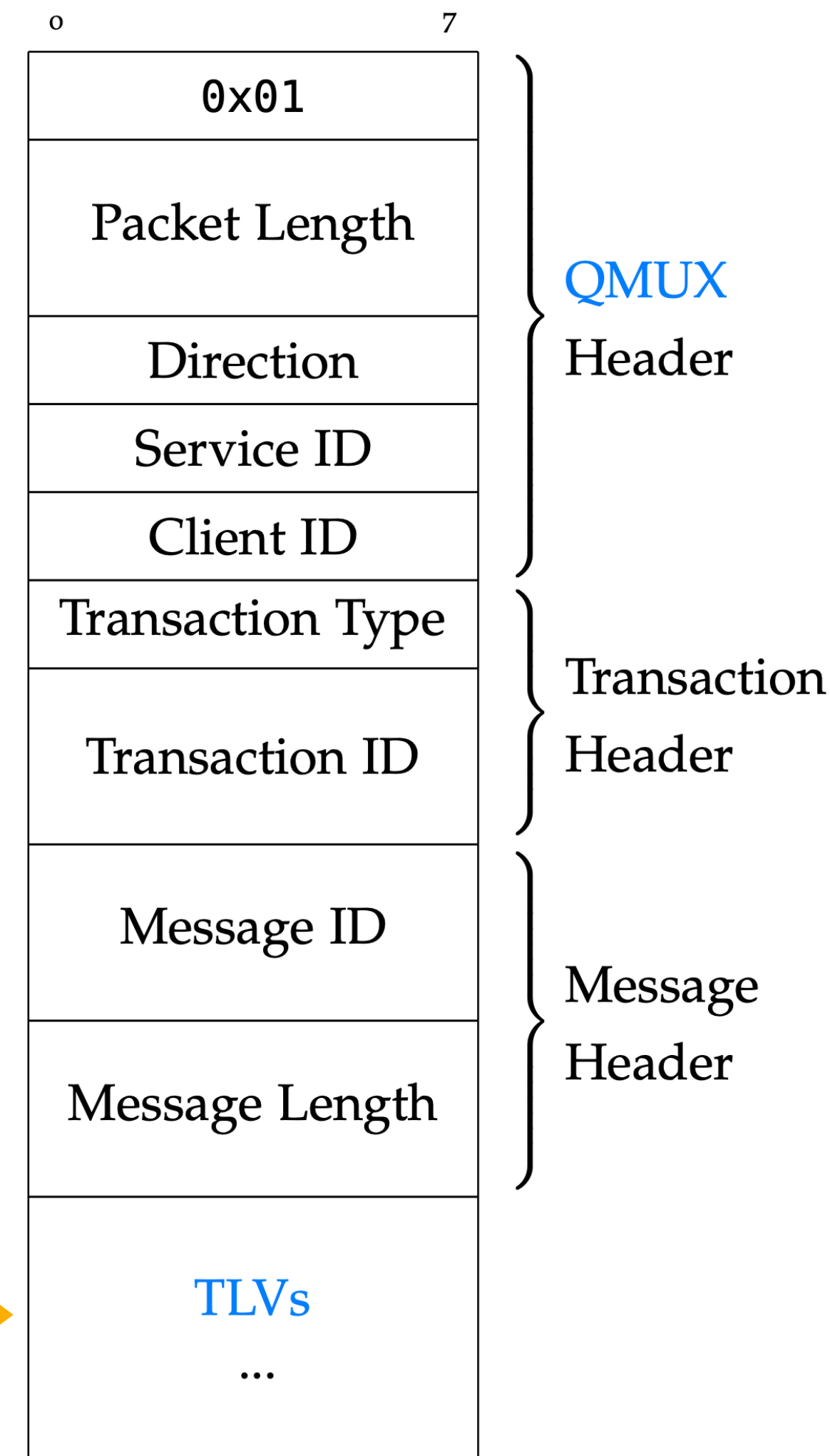
Symbolicate Packets

Simple in theory, hard to automate 🤖

- Services – `as{Short, Long}String`
- Messages – `MessageBase::{MessageBase, validateMsgId}`
- TLV Structures – `tlv::{parseV<T>, writeV<T>}`
- TLV IDs - `MutableMessageBase::getTLV` & `MessageBase::{findTlvValue, findNextTlv, applyTlv}`



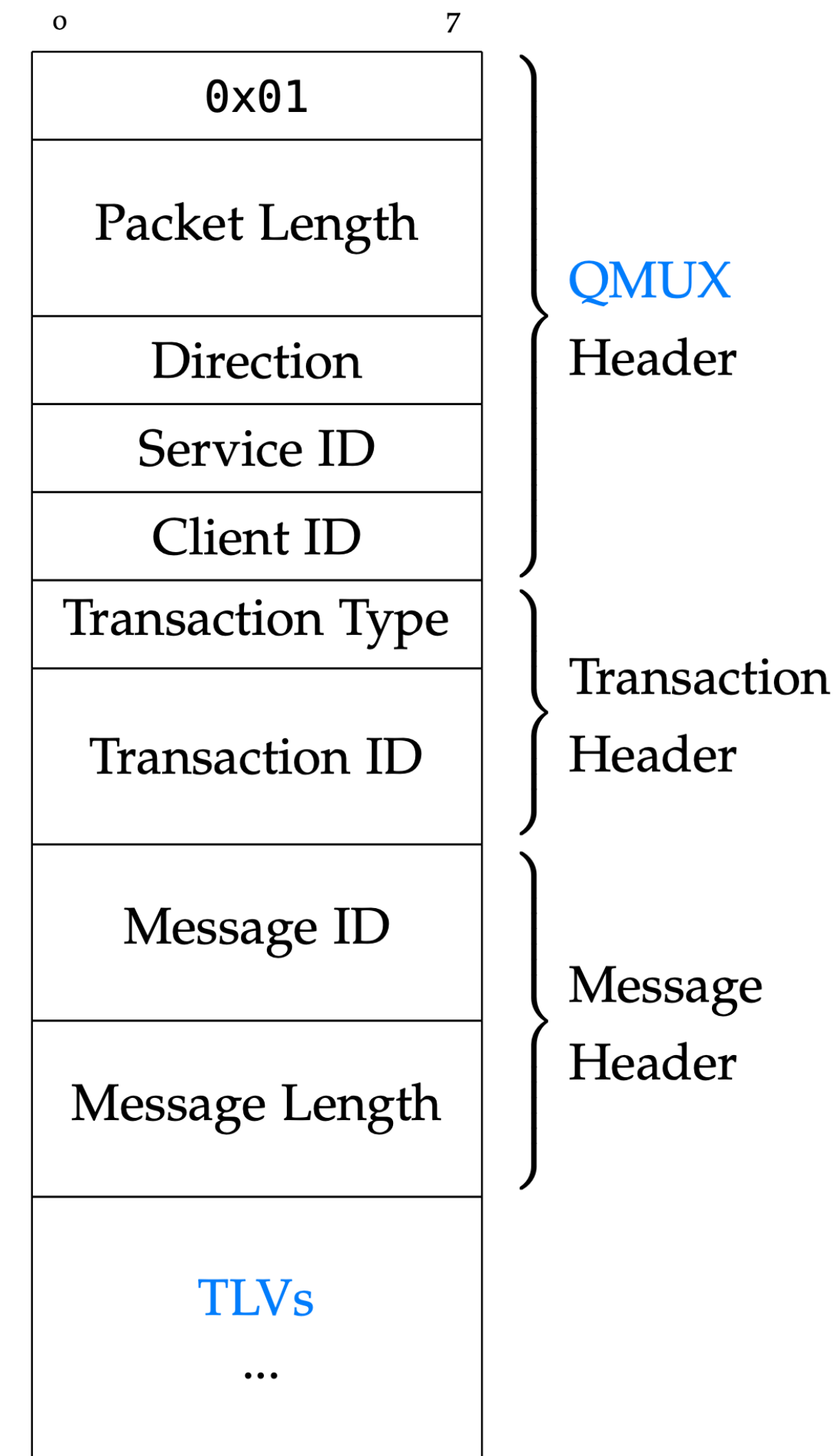
Loca...	Name	IDs	Capabilities	Size	Structure
2031c7370	nas::tlv::EmergencyType		ws	7	4
2031c73e8	nas::tlv::lccid		rws	4 + ?	[1 ? ?]
2031c74c8	nas::tlv::ScanCompletionStatus		r	1	1
2031c7520	nas::tlv::ScanStatus		r	2	2
2031c75dc	nas::tlv::IncrementalNetworkInfo		r		[2 ([2 2 1 [1] ..
2031c7648	nas::tlv::RegisterAction		ws	4	1
2031c7664	nas::tlv::ManualNetworkRegisterInfor...		ws	8	2 2 1
2031c7694	nas::tlv::MNCDigitIncludeStatus		ws	4	1
2031c76b0	nas::tlv::BlindTransfer		ws	7	2 2



Symbolicate Packets

Simple in theory, hard to automate 🤖

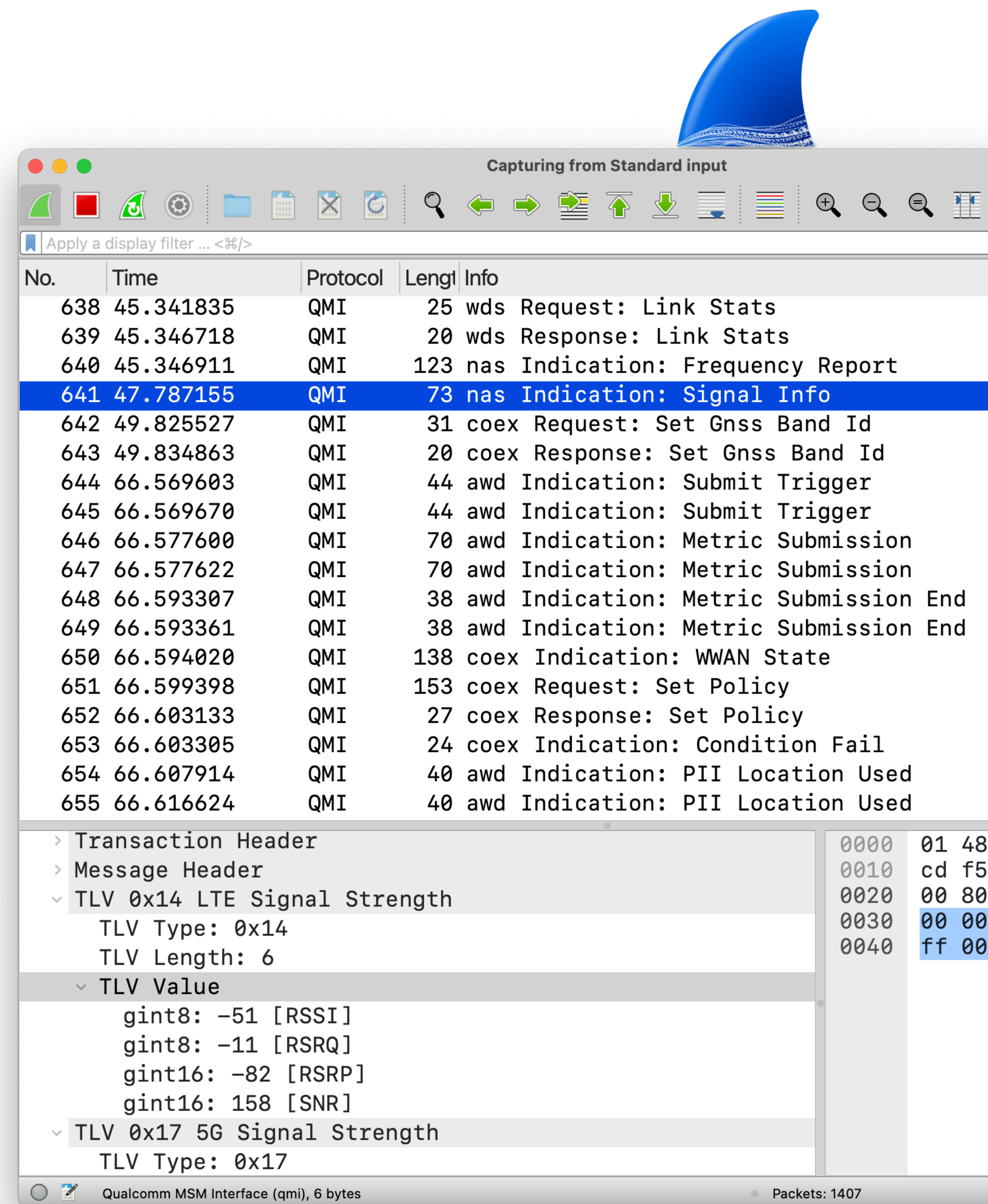
- Services – `as{Short, Long}String`
- Messages – `MessageBase::{MessageBase, validateMsgId}`
- TLV Structures – `tlv::{parseV<T>, writeV<T>}`
- TLV IDs - `MutableMessageBase::getTLV` & `MessageBase::{findTlvValue, findNextTlv, applyTlv}`
- Challenges: Missing symbols, many indirections, and historical grown mess 🦖



Packet Dissection

Our Wireshark dissector for QMI

- Uses existing libqmi packet definitions, completed with our own 📖
- Support for (provided) TLV structures
- Collect packets anywhere without jailbreak for later analysis 🔬
- Helpful in understanding iPhone's satellite communication —> [TROOPERS24](#) and [#OBTS v6.0](#) (Jiska 🦄 & Alex ❄️)







BaseTrace Wireless Diagnostics

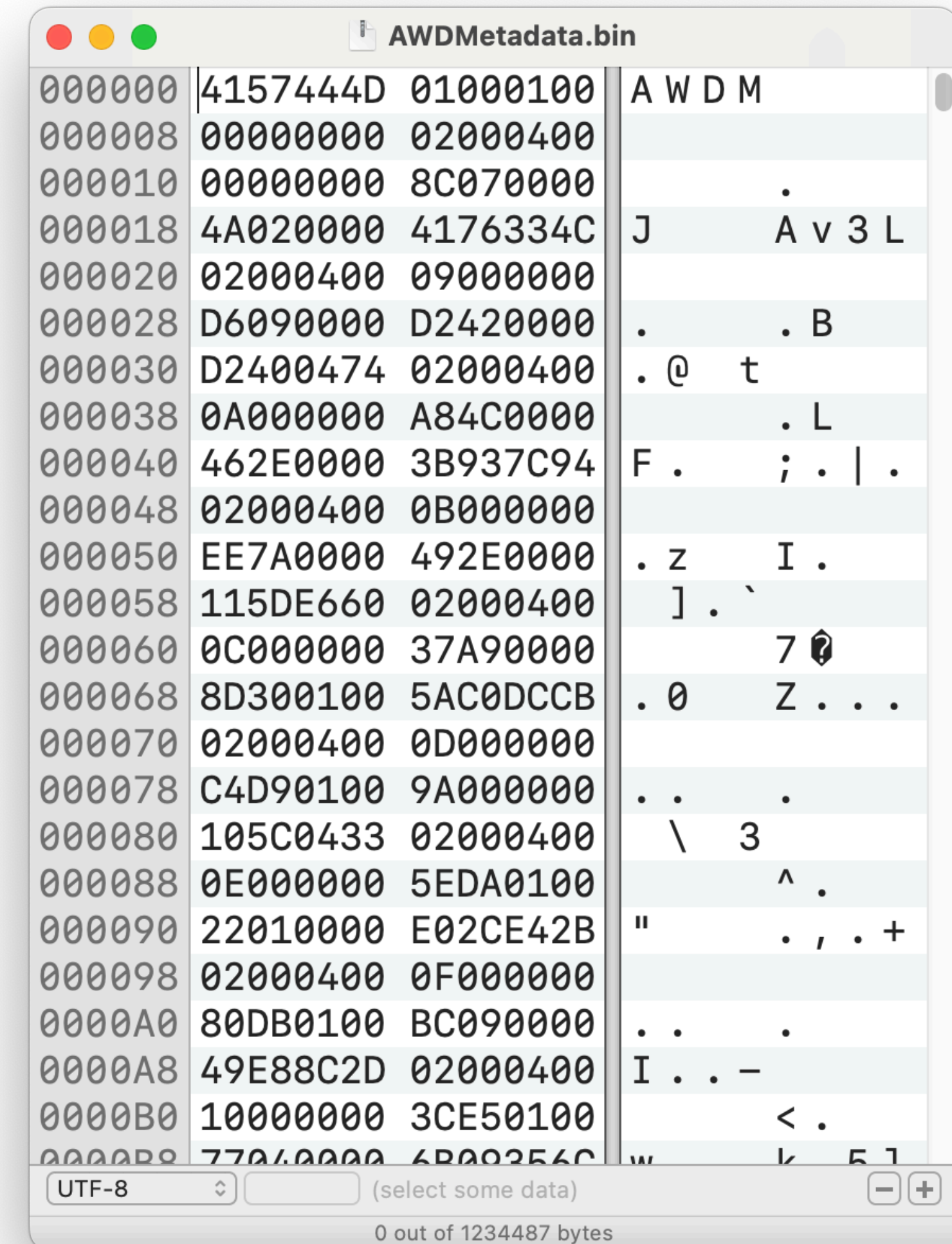


FRIDA

Apple Wireless Diagnostics

Monitoring performance of iOS components

- Collect binary logs of various (wireless) coprocessors 
- Diagnostics metadata is encoded in manifest files shipped with iOS 
 - Root Manifest – /System/Library/PrivateFrameworks/WirelessDiagnostics.framework/Support/AWDMetadata.bin
 - Extension Manifests – /System/Library/AWD/Metadata/
- Metadata has custom binary format, decoded by [@rickmark](#) [@nicolas17](#)

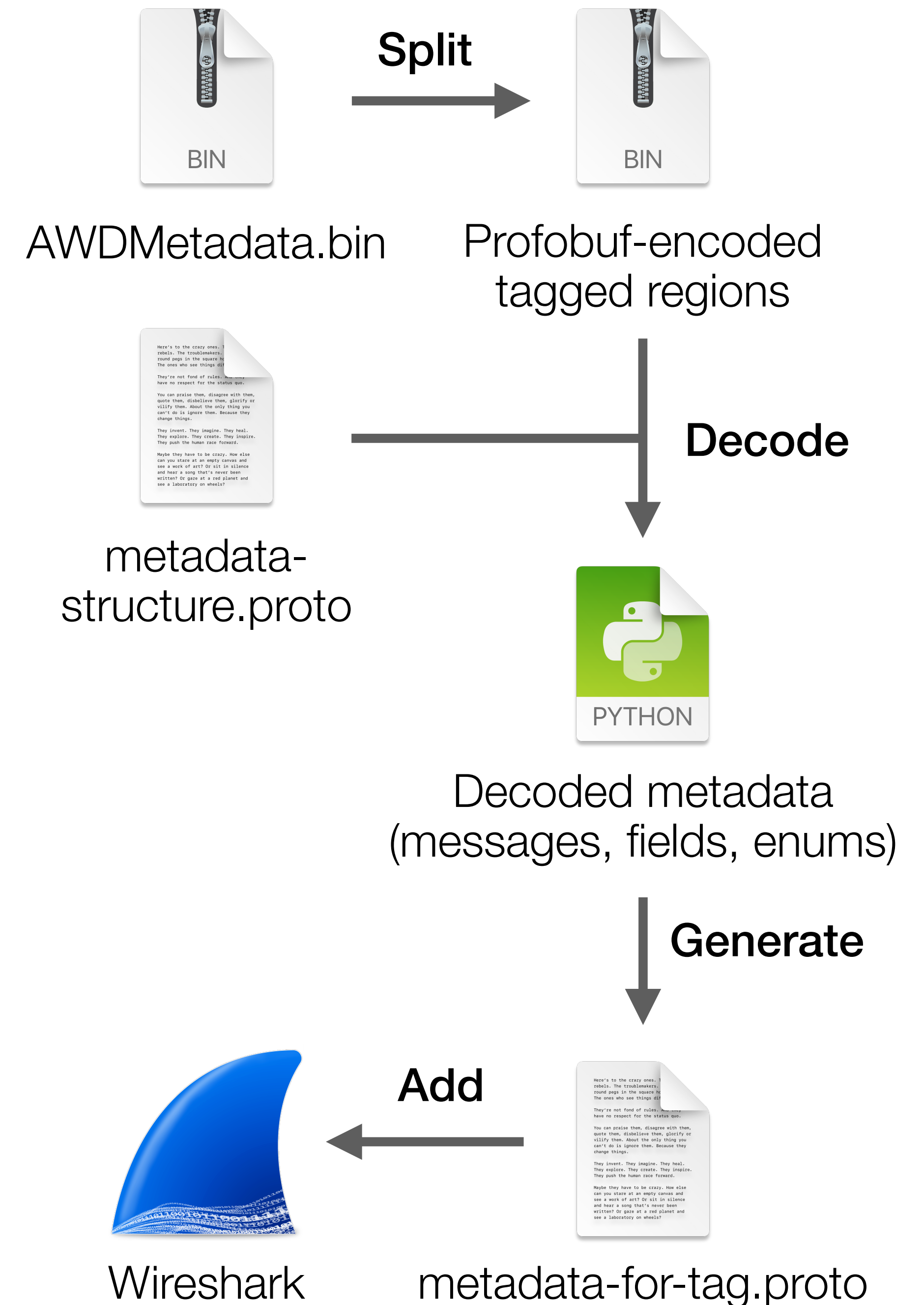


```
AWDMetadata.bin
000000 4157444D 01000100 A W D M
000008 00000000 02000400
000010 00000000 8C070000
000018 4A020000 4176334C J A v 3 L
000020 02000400 09000000
000028 D6090000 D2420000 . B
000030 D2400474 02000400 . @ t
000038 0A000000 A84C0000 . L
000040 462E0000 3B937C94 F . ; . | .
000048 02000400 0B000000
000050 EE7A0000 492E0000 . z I .
000058 115DE660 02000400 ]. `
000060 0C000000 37A90000 7
000068 8D300100 5AC0DCCB . 0 Z . . .
000070 02000400 0D000000
000078 C4D90100 9A000000 . . .
000080 105C0433 02000400 \ 3
000088 0E000000 5EDA0100 ^ .
000090 22010000 E02CE42B " . , . +
000098 02000400 0F000000
0000A0 80DB0100 BC090000 . . .
0000A8 49E88C2D 02000400 I . . -
0000B0 10000000 3CE50100 < .
0000B8 77040000 6B002560 w k 5 1
UTF-8 (select some data)
0 out of 1234487 bytes
```

Decoding AWD Metadata

It's all about Protocol Buffers

- Metadata files consists of multiple Protobuf-decodable tag regions
- Protocol Buffers (Protobuf) is Google's data interchange format
- Regenerate Protobuf code from decoded metadata with a simple „compiler“ 🧑‍🔧
- Region 2 includes field & enum names 🎉



Decoding AWD in Wireshark

Feeding the shark 🦈

- AWD QMI service generates ~1/3 of traffic
- Wireshark dissector uses generated Protobuf to decode submitted metrics from baseband
- AWD tag 12 is relevant for cellular, 68 tags in total for various iOS features
 - Wi-Fi, Bluetooth, VPN, FaceTime, ...

The screenshot shows a Wireshark capture window titled "Capturing from Standard input". The packet list pane shows several QMI packets. Packet 55 is selected, showing a QMI packet of length 66 bytes with the info "awd Indication: Metric Submission". The packet details pane shows the following structure:

```
Message Length: 53
TLV 0x52 awd::tlv::abm::MetricDataExt
TLV Type: 0x52
TLV Length: 50
  TLV Value
    guint32: 1 [app_id]
    guint32: 29546 [component_id]
    guint32: 680004 [trigger_id]
    guint32: 816687 [profile_id]
    guint32: 816687 [metric_id]
    guint32: 12 [submission_id]
    guint16: 32768 [other_id]
    guint16: 22 [Size Prefix] [payload]
  Protocol Buffers: awdd.AWDMetadatabin_region_3_tag_12.message_0
    Message: awdd.AWDMetadatabin_region_3_tag_12.message_0
      Field(816687): cellularPowerLogLTE_RRCStateChange (message)
        Message: awdd.AWDMetadatabin_region_3_tag_12.CellularPowerLogLTE_
          Field(1): timestamp = 1732380487585 (uint64)
          Field(2): state = RRC_339_CONNECTED(4) (enum)
          Field(3): prev_state = RRC_339_CONNECTING(3) (enum)
          Field(4): prev_state_dur_ms = 70 (uint32)
          Field(5): subs_id = 0 (uint32)
```

The status bar at the bottom indicates "Qualcomm MSM Interface (qmi), 53 bytes", "Packets: 440", and "Profile: Default".



BaseTrace Qualcomm DIAG

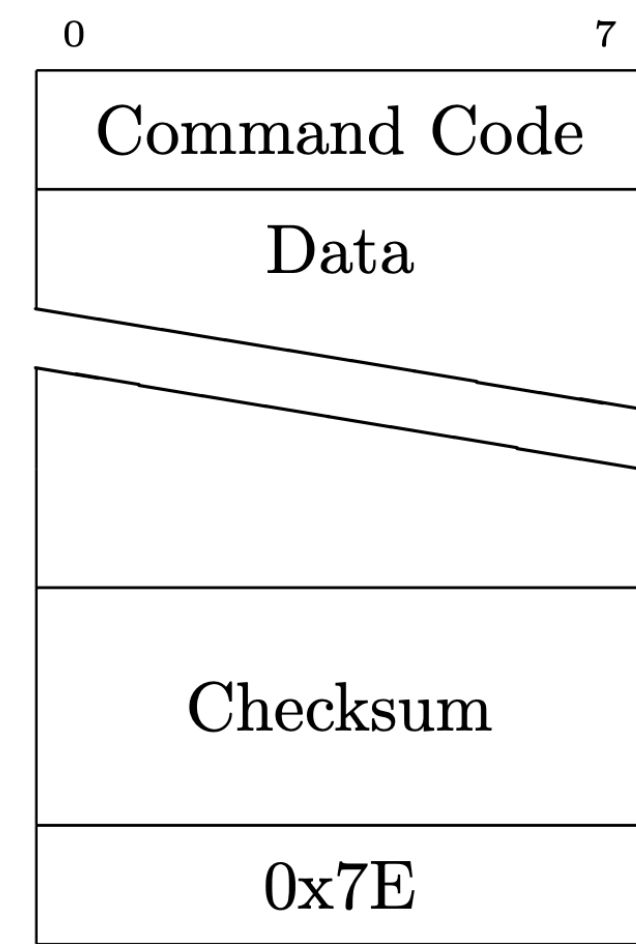


FRIDA

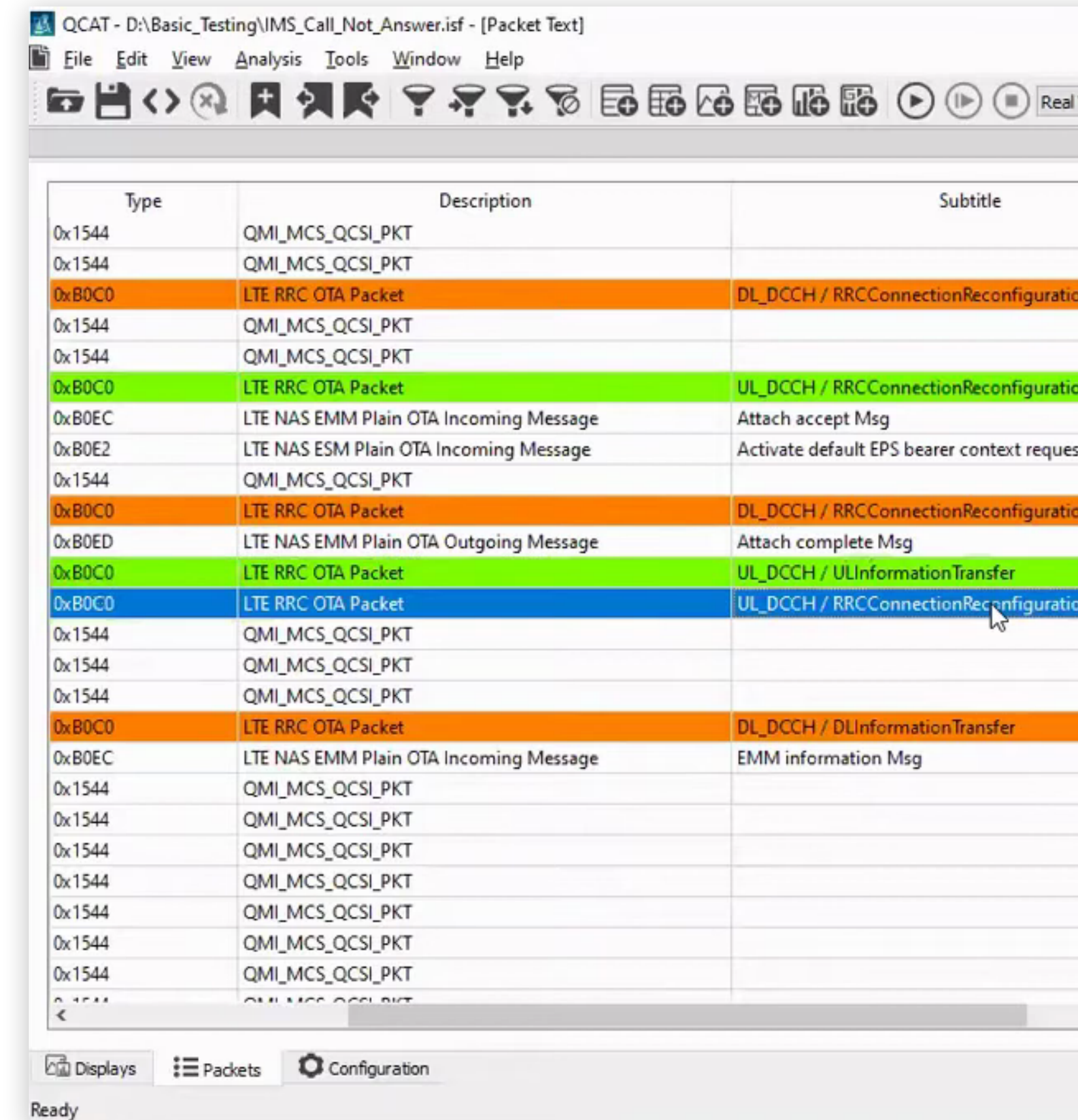
Qualcomm DIAG

Yet another proprietary protocol

- Diagnostics protocol for basebands from Qualcomm
 - NR, LTE, UMTS, ... over-the-air packets 🙄🙄
- Intended to be used with Qualcomm® eXtensible Diagnostic Monitor (QXDM) – Expensive software hidden behind NDAs 😞
- Many open-source DIAG implementations 🥳
 - P1Sec/QCSuper
 - fgsect/scat



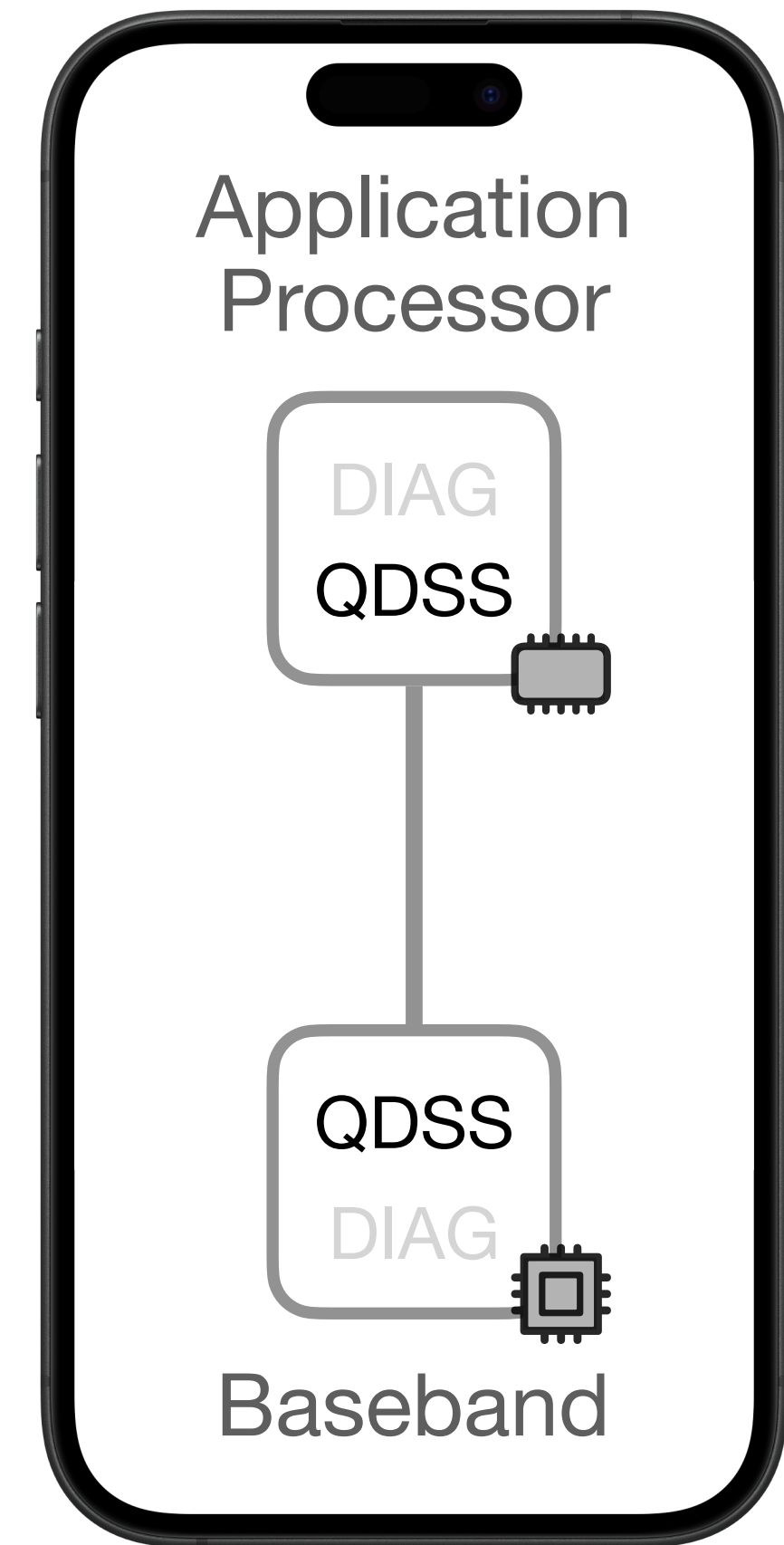
Source: HowITworks / YT



Qualcomm DIAG on iPhones?

There's QDSS and we don't like it

- iPhones use Qualcomm BBs, so they should also support DIAG
 - Open-source tools only support Android or USB modems
 - Sysdiagnoses have log-bb-[date]-qdss directory with BB data
 - Tools can't process it, encoded using newer **QDSS** protocol
- Can we convince the iPhone to „downgrade“ to DIAG?
 - abm-helper is responsible for data collection

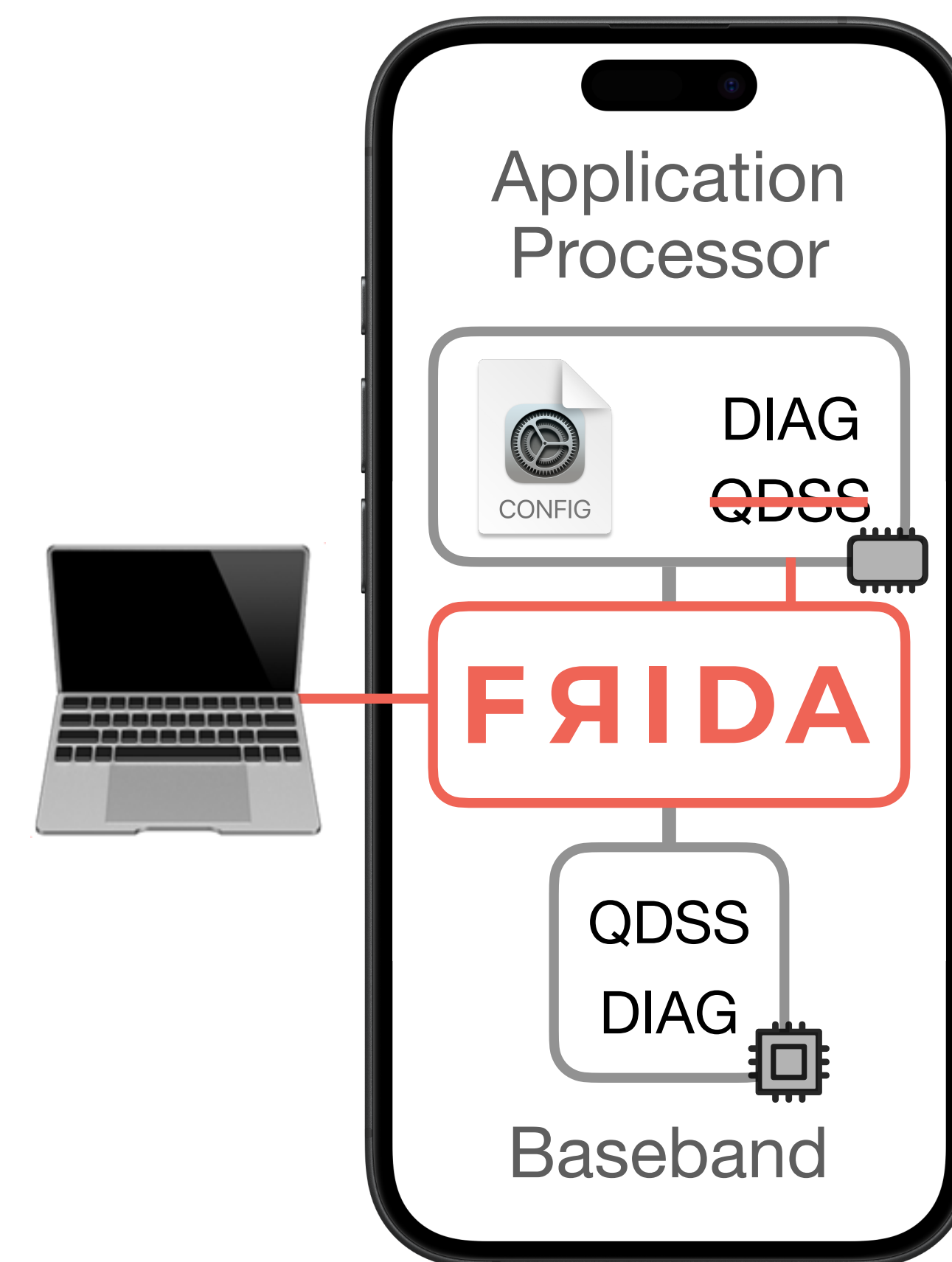


Qualcomm Debug Subsystem
– modern version of DIAG

Qualcomm DIAG on iPhones!

Reject modernity, return to DIAG

- **abm-helper** only active if baseband debug profile installed
- Checks different telephony capabilities upon startup: ADPL, QDSS, QShrink4
- Hook them with Frida and lie that we don't support them
- **abm-helper** falls back to DIAG we can intercept



abm-helper is part of
Apple Baseband Manager

Qualcomm DIAG on iPhones!

Reject modernity, return to DIAG

- **abm-helper** only active if baseband debug is enabled
- Checks different telephony capabilities using ADPL, QDSS, QShrink4
- Hook them with Frida and lie that we don't have these capabilities
- **abm-helper** falls back to DIAG we can intercept
- Integrate with QCSuper's framework

No.	Time	Protocol	Length	Info
136	11.551860	LTE RRC UL_DCCH	66	MeasurementReport
138	12.126338	LTE RRC DL_DCCH	68	RRCConnectionReconfiguration
140	12.137729	LTE RRC UL_DCCH	46	RRCConnectionReconfigurationComplete
141	12.206036	LTE RRC DL_DCCH	84	RRCConnectionReconfiguration
143	12.263694	LTE RRC UL_DCCH	46	RRCConnectionReconfigurationComplete
148	14.045046	LTE RRC DL_DCCH	53	RRCConnectionReconfiguration
150	14.057687	LTE RRC UL_DCCH	46	RRCConnectionReconfigurationComplete
158	16.671369	LTE RRC UL_DCCH	66	MeasurementReport
166	18.330934	LTE RRC DL_DCCH	107	RRCConnectionRelease [cause=other]
221	20.018764	LTE RRC DL_SCH	72	SystemInformationBlockType1
222	20.028762	LTE RRC DL_SCH	91	SystemInformation [SIB2 SIB3]
258	20.069966	LTE RRC DL_SCH	91	SystemInformation [SIB2 SIB3]
259	20.071854	LTE RRC DL_SCH	72	SystemInformationBlockType1
260	20.073397	LTE RRC DL_SCH	66	SystemInformation [SIB4]
262	20.076797	LTE RRC DL_SCH	91	SystemInformation [SIB5]
264	20.082204	LTE RRC DL_SCH	99	SystemInformation [SIB7]
276	27.569849	GSMTAP/NAS-EPS	48	Service request
277	27.569942	LTE RRC UL_CCCH	50	RRCConnectionRequest
288	27.673243	LTE RRC DL_CCCH	68	RRCConnectionSetup
289	27.678398	LTE RRC UL_DCCH/N...	53	RRCConnectionSetupComplete, Service request
295	27.712439	LTE RRC DL_DCCH	47	SecurityModeCommand
296	27.712822	LTE RRC UL_DCCH	46	SecurityModeComplete
297	27.714332	LTE RRC DL_DCCH	158	RRCConnectionReconfiguration
304	27.739379	LTE RRC DL_DCCH	168	RRCConnectionReconfiguration
309	27.778939	LTE RRC DL_DCCH	50	UECapabilityEnquiry
310	27.782639	LTE RRC UL_DCCH	144	UECapabilityInformation
331	32.562454	LTE RRC PCCH	51	Paging (1 PagingRecord)
343	40.957007	LTE RRC DL_DCCH	54	RRCConnectionReconfiguration
345	40.960563	LTE RRC UL_DCCH	46	RRCConnectionReconfigurationComplete



BaseTrace Baseband Firmware

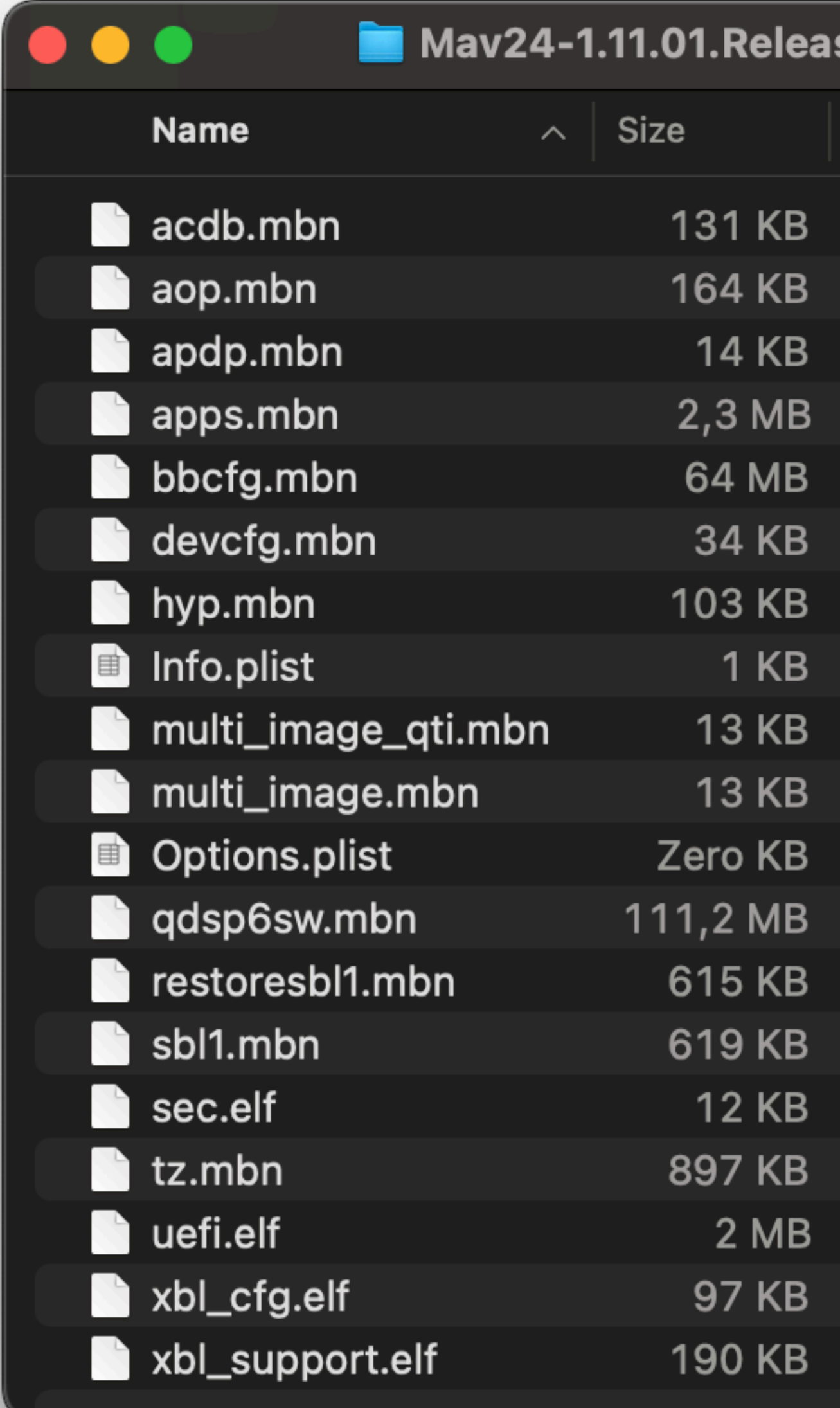


FRIDA

Baseband Firmware

Welcome to hell

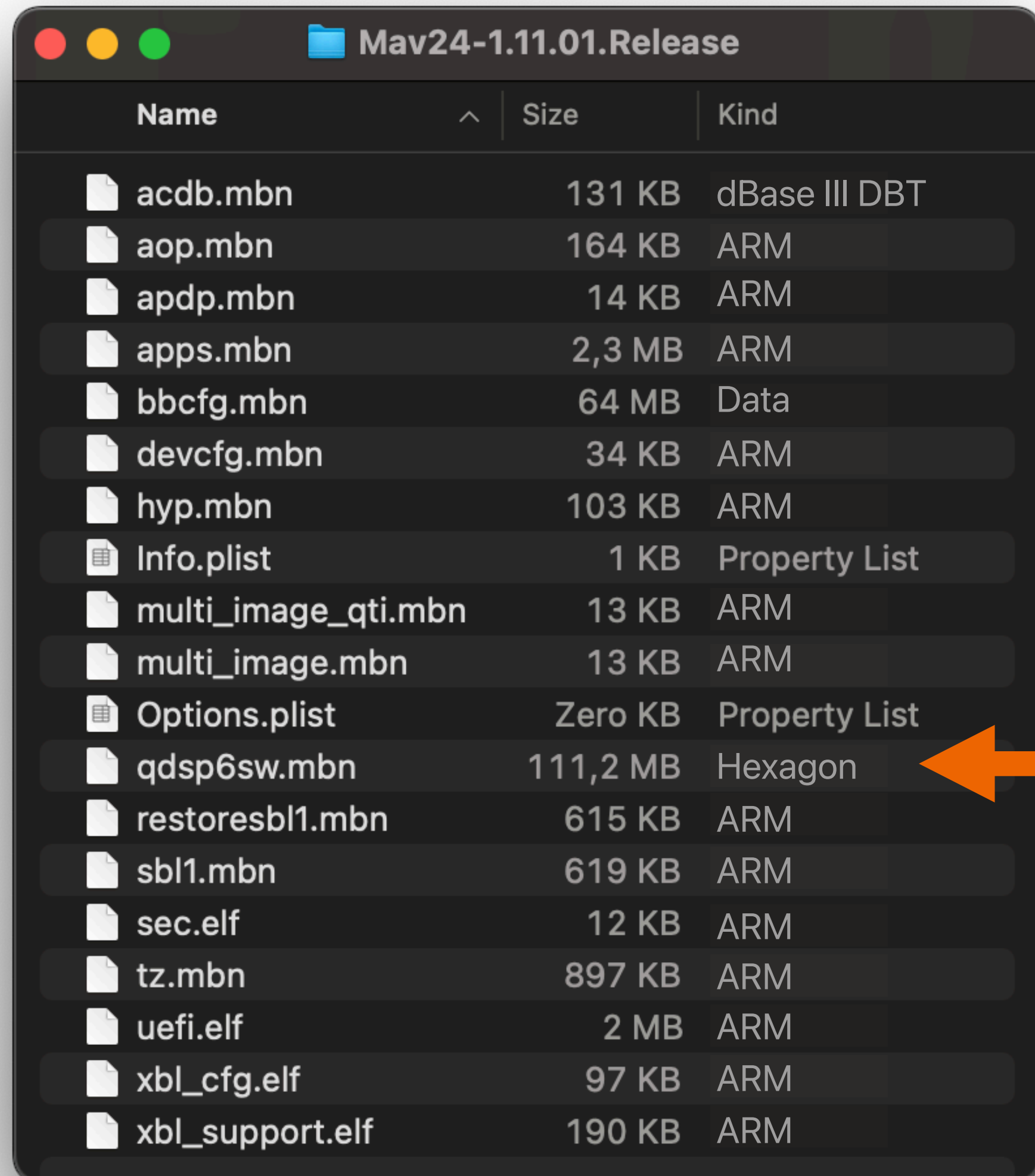
- Compiled by Apple itself with some heavy modifications
- Updates delivered as part of firmware image
 - `/Firmware/ManufacturerVersionGenerationMav24-1.11.01.Release.bbfw`
- MAV – Apple’s firmware for Qualcomm basebands
 - Maverick – Features implemented by Apple
 - Eureka – Features implemented by Qualcomm
- ICE – (Apple’s) Firmware for Intel basebands



Name	Size
acdb.mbn	131 KB
aop.mbn	164 KB
apdp.mbn	14 KB
apps.mbn	2,3 MB
bbcfig.mbn	64 MB
devcfg.mbn	34 KB
hyp.mbn	103 KB
Info.plist	1 KB
multi_image_qti.mbn	13 KB
multi_image.mbn	13 KB
Options.plist	Zero KB
qdsp6sw.mbn	111,2 MB
restoresbl1.mbn	615 KB
sbl1.mbn	619 KB
sec.elf	12 KB
tz.mbn	897 KB
uefi.elf	2 MB
xbl_cfg.elf	97 KB
xbl_support.elf	190 KB

Mav24 v1.11.01

Qualcomm Snapdragon X71 5G Modem in iPhone 16



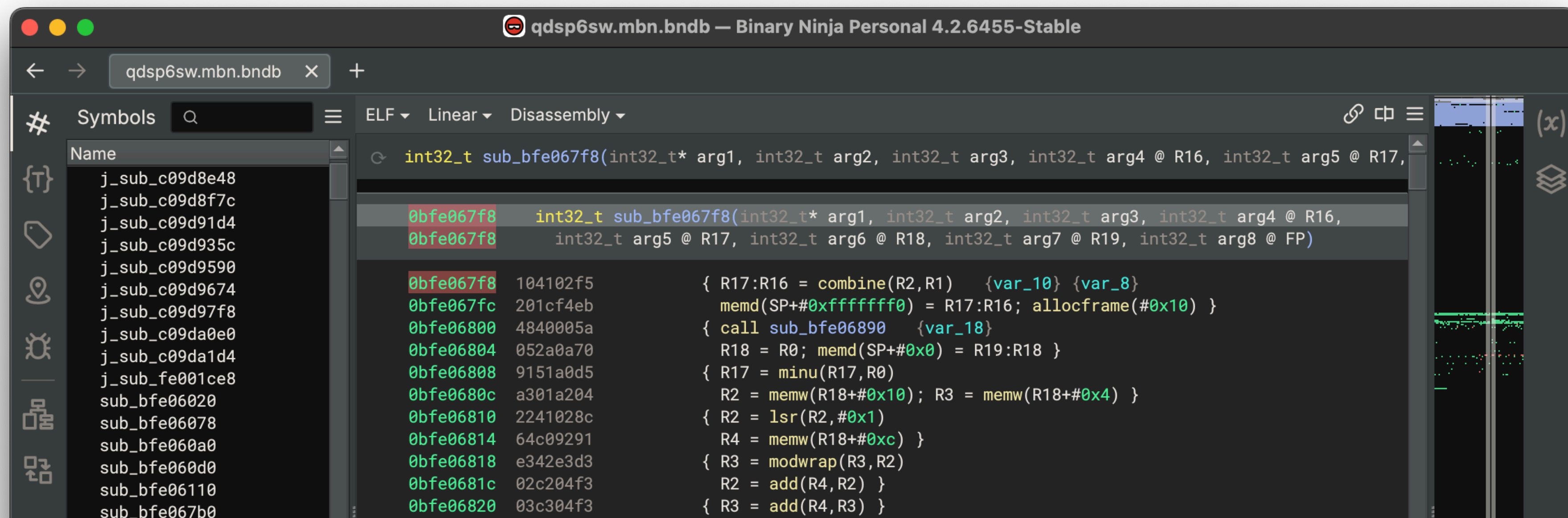
Name	Size	Kind
acdb.mbn	131 KB	dBase III DBT
aop.mbn	164 KB	ARM
apdp.mbn	14 KB	ARM
apps.mbn	2,3 MB	ARM
bbcfig.mbn	64 MB	Data
devcfg.mbn	34 KB	ARM
hyp.mbn	103 KB	ARM
Info.plist	1 KB	Property List
multi_image_qti.mbn	13 KB	ARM
multi_image.mbn	13 KB	ARM
Options.plist	Zero KB	Property List
qdsp6sw.mbn	111,2 MB	Hexagon
restoresbl1.mbn	615 KB	ARM
sbl1.mbn	619 KB	ARM
sec.elf	12 KB	ARM
tz.mbn	897 KB	ARM
uefi.elf	2 MB	ARM
xbl_cfg.elf	97 KB	ARM
xbl_support.elf	190 KB	ARM

- Communication with iOS over PCIe
- Parts of baseband filesystem mounted at `/private/var/wireless/baseband_data`
- Analysis of some files in [@Rick Mark's blog](#)
- Baseband has ARM and Hexagon cores
- Most baseband functionality is implemented in `qdsp6sw.mbn` which uses Hexagon 🤔

Quick Look into qdsp6sw.mbn

Reversing Qualcomm Hexagon is not fun

- Hexagon is a proprietary ISA for Qualcomm's Digital Signal Processor
- DSP uses RTOS with fewer security mitigations compared to GPOS
- Some public documentation and many strings, but limited RE tooling support



Further Hexagon Research

Dive even deeper if you want 🐟



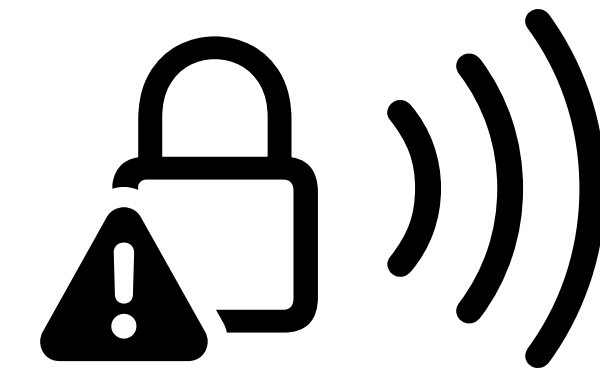
- [A Journey into Hexagon: Dissecting Qualcomm Basebands, 2018, Seamus Burke](#)
- Exploring Qualcomm Baseband via ModKit, 2018, Tencent Blade Team
- [Attacking Hexagon: Security Analysis of Qualcomm's aDSP, 2019, Dimitrios Tatsis](#)
- [Advanced Hexagon DIAG and getting started with baseband vulnerability research, 2020, Alisa Esage](#)
- [In-Depth Analyzing and Fuzzing for Qualcomm Hexagon Processor, 2021, Xilig Gong & Bo Zhang](#)



Leave the rest to experts
and do some fun stuff 😊



CellGuard
Cellular Security



Cellular Security

Attack Goals



Personal Information &
Location Tracking



Traffic Interception &
Manipulation



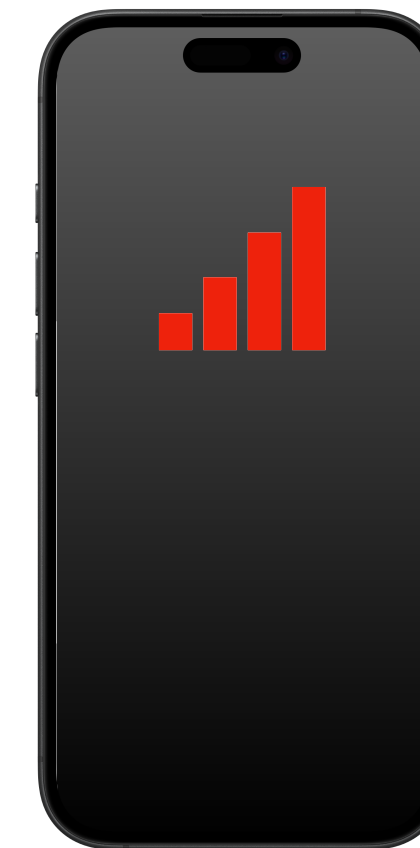
Baseband Vulnerability
Exploitation

Attack Model

Adversaries can **block**,
intercept, and **modify**
over-the-air signals



Rouge Base Station



Genuine Base Station

Cellular Security

Attack Vectors

2G

Downgrade attacks

Missing mutual authentication

5G

Improves security

Targeted information leakage

3G & 4G

Missing integrity protection

Identity information leakage

General

Roaming abuse

Baseband exploits

Protect yourself by disabling 2G

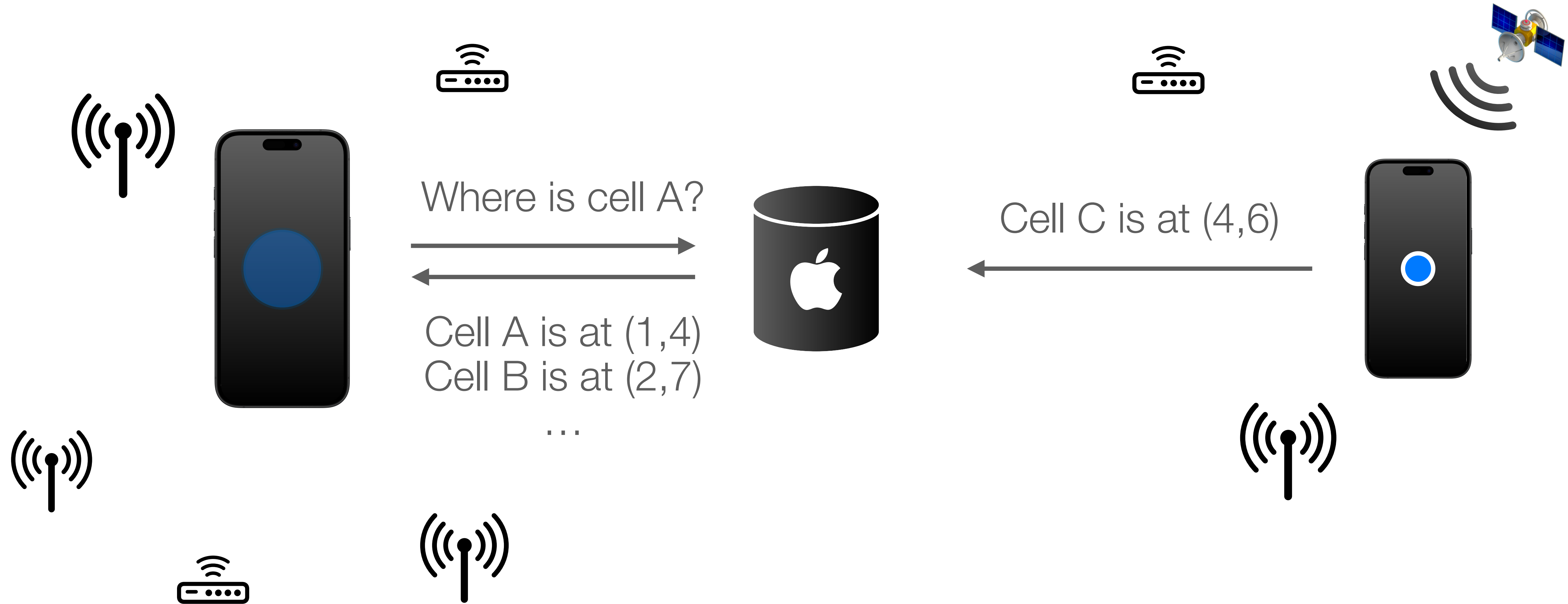


CellGuard
Apple Location Services



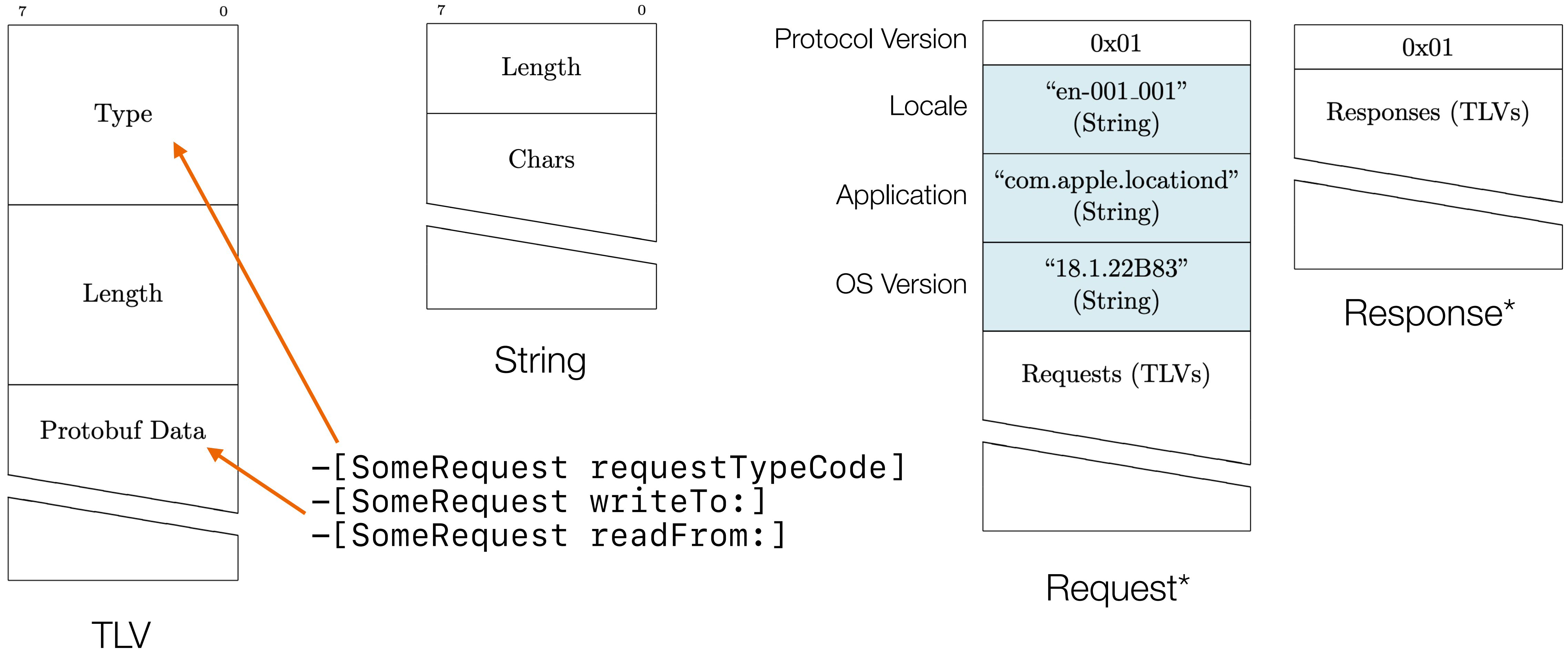
Apple Location Services

Is Apple's Closed-Source Location Database



ProtocolBuffer.framework

Apple's private framework for Protobuf

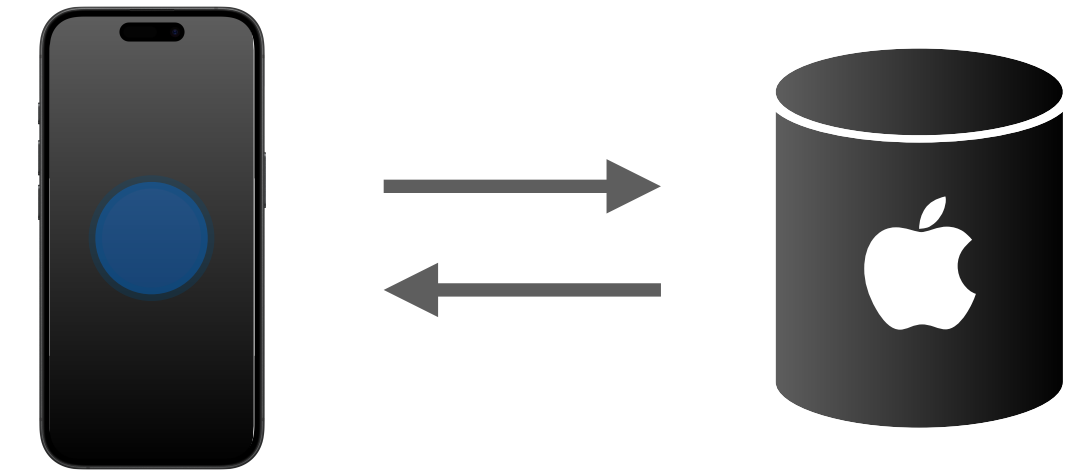


* Byte representation of figures not to scale

Data Request

<https://gs-loc.apple.com/clls/wloc>

- Retrieve approximate locations for WiFi, GSM, CDMA, SCDMA, LTE, NR
- Cached locally at `/var/root/Library/Caches/locationd/cache_encryptedB.db`
- Processing: `ALS*` in `/usr/sbin/locationd`
- Abusable for tracking of devices that provide Wi-Fi hotspots (Erik Rye & Dave Levin)
- More practical research by @acheong08



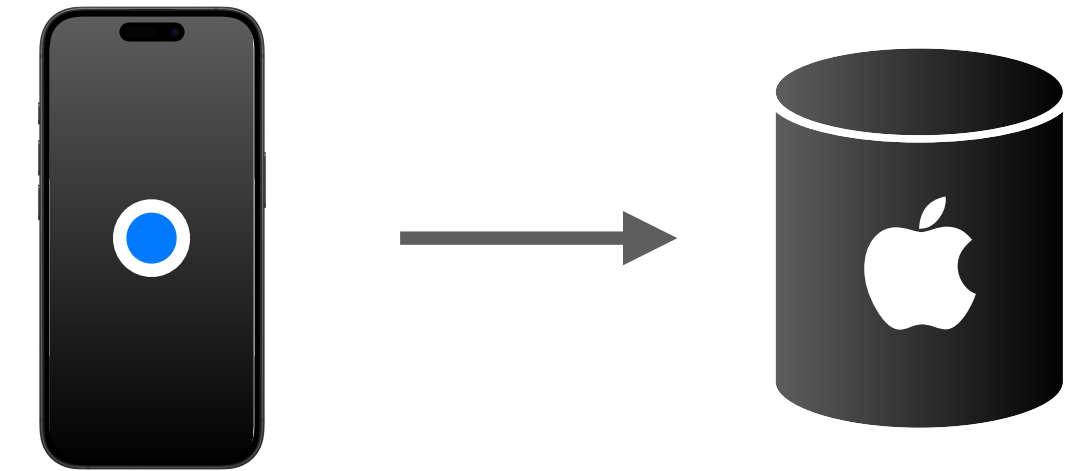
ALSLocationRequest **Type: 0x01**
→ Observed WiFis & cells
← Locations of nearby WiFis & cells

ALSNearbyRequest **Type: 0x03**
→ Coordinates
← Locations of nearby WiFis & cells



Data Injection

<https://gsp10-ssl.apple.com/hcy/pbcwloc>



- How does Location Services learn about new WiFis & cells?
- iPhones continuously harvest data in the background and share it with Apple
 - `/var/root/Library/Caches/locationd/harvest/CLSubHarvesterCell-main`
 - Processing: CLP* classes in `CoreLocationProtobuf.framework`
- Other harvesters: Indoor, Points of Interest, Pressure, Altimeter, Ionosphere

CellWifiCollectionRequest **Type: 0x64**
→ Observed Wi-Fi's, cells, and locations
← Successful injection?

```
lteCellTowerLocations {
  mcc: 262
  mnc: 2
  tac: 45711
  ci: 26396672
  uarfcn: 6300
  pid: 33
  bandInfo: 20
  location {
    latitude: 49.873538
    longitude: 8.655094
    horizontalAccuracy: 4.79252434
    altitude: 327.501434
    verticalAccuracy: 3.29488206
    timestamp: 752791442.00046265
    provider: 1
    motionVehicleConnectedStateChanged: false
  }
}
```


Geo Configuration



<https://configuration.ls.apple.com/config/defaults>

- iPhones fetch country specific configuration for Location Services
- Request parameters: OS, OS Version, Hardware Version
- Response is plist file and contains many feature flags for countries
- iPhones use special servers for Location Services in China

The screenshot shows a plist file named 'config-defaults-iphone14.plist' with the following structure:

Key	Type	Value
Root	Dictionary	(1 item)
com.apple.GEO	Dictionary	(102 items)
AllowGTSuggestionForGTHandles	Boolean	YES
AllowResumingIncompleteResourceDownloads	Boolean	NO
AllowWalkingRouteSuggestionForGTHandles	Boolean	YES
BellflowerPlaceCard	Boolean	YES
BrooklynEnableSharingForAllGuides	Boolean	YES
CLAppleCollectionServer_Altimeter	String	https://gsp10-ssl.ls.apple.com/hvr/alt
CLRealTimeBudgetBytesDefault	Number	838.860
CLTrafficCrowdsourcingRules	Dictionary	(1 item)
CarPlayShouldUseCPMetadataSPIs	Boolean	YES
CellSaverRegionDownloadEnabled	Boolean	NO
CellSaverStaleThresholdInSeconds	Number	864.000
CellSaverTileTypeEnabled.13	Boolean	NO
CellSaverTileTypeEnabled.37	Boolean	NO
CellSaverTileTypeEnabled.53	Boolean	YES
CellSaverTileTypeEnabled.54	Boolean	YES
CohortReportingThresholds	Dictionary	(2 items)
CongestionZonesEnabled	Boolean	YES
CountryProviders	Dictionary	(124 items)
FVRoutingNameForBundleIDKey	String	app_bundle_id

LASD Data

<https://cl1.apple.com/1/v3/lasd.plist>



- IPSWs contain coarse ALS copy allowing for quick network connections 🚑
- SQLite3 DBs with RTree for GSM, CDMA, UMTS, SCDMA, LTE, NR
- Files consumed by CommCenter
- CommCenterMobileHelper regularly fetches updates

```
# IPSW
/System/Library/Frameworks/CoreTelephony.framework/Support/
# Updated databases
/var/mobile/Library/LASD
```




CellGuard Detection Algorithm



Rouge Base Station Detection

With Apple Location Services (ALS)

1. Confirm existence of cell with ALS (20P)



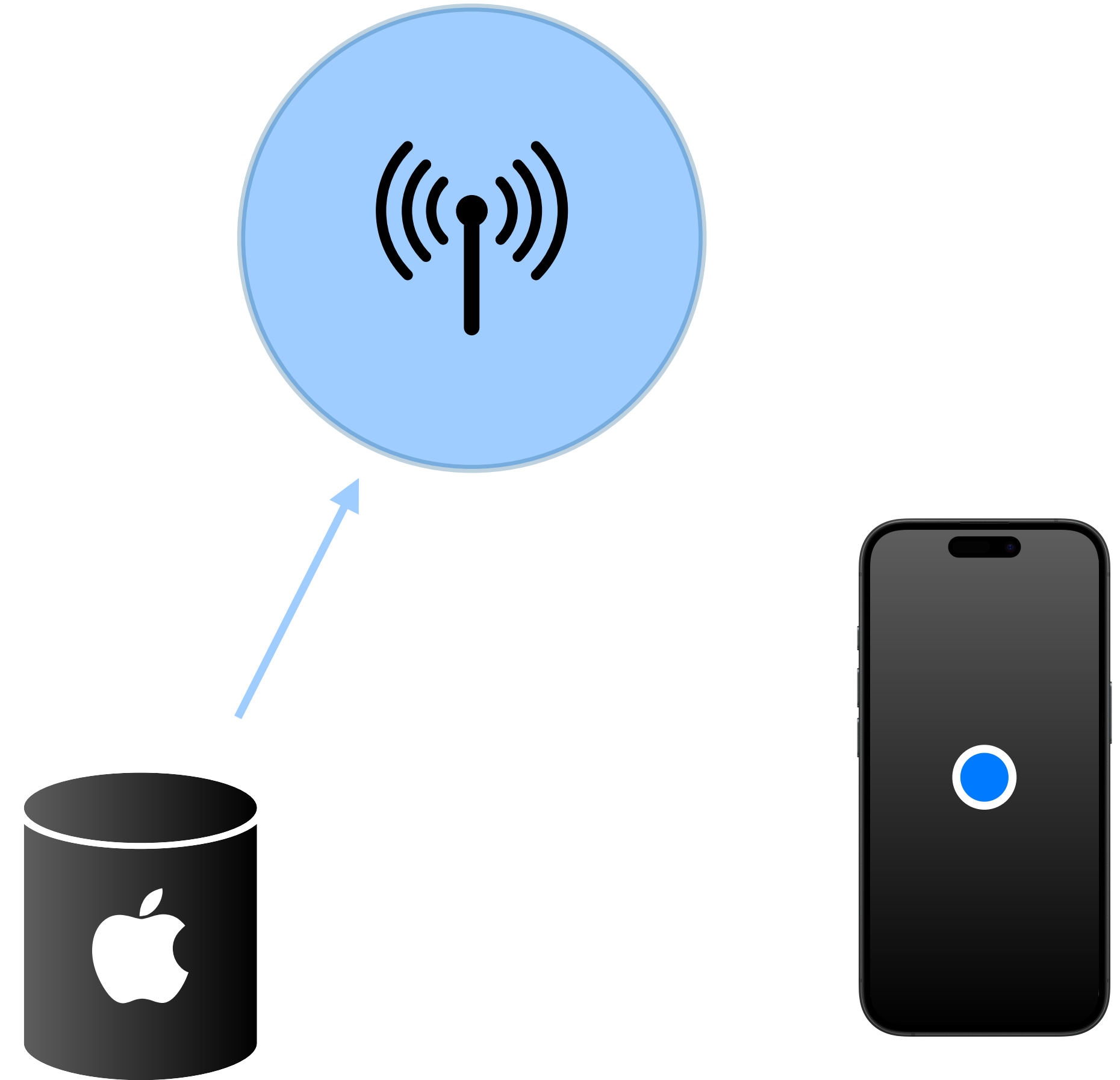
Cell exists?

A horizontal double-headed arrow pointing both left and right, positioned below the text 'Cell exists?'.

Rouge Base Station Detection

With Apple Location Services (ALS)

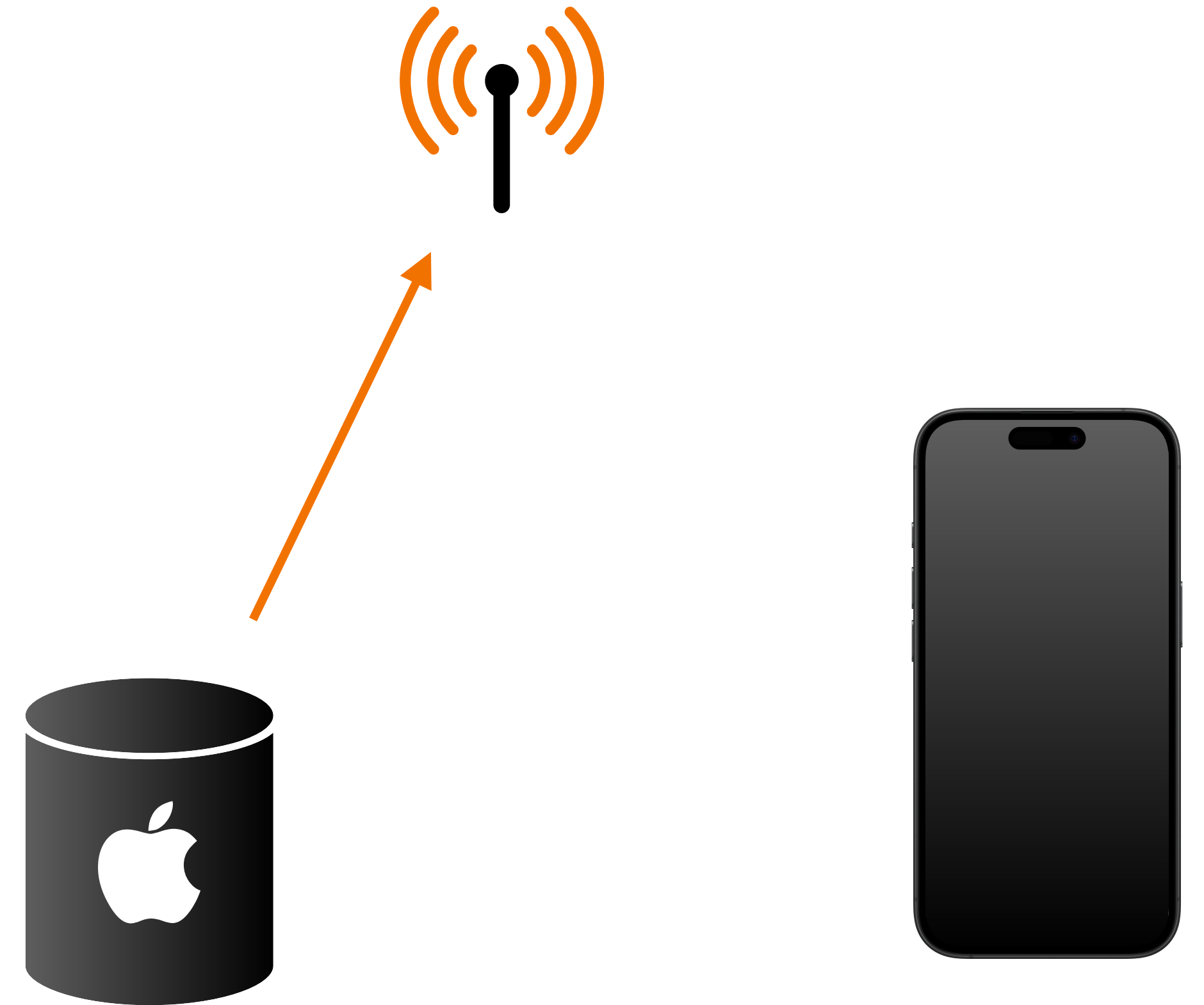
1. Confirm existence of cell with ALS (20P)
2. Calculate distance between recorded and ALS location (20P)



Rouge Base Station Detection

With Apple Location Services (ALS)

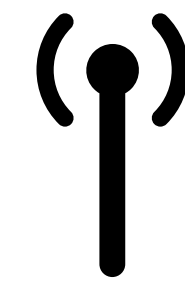
1. Confirm existence of cell with ALS (20P)
2. Calculate distance between recorded and ALS location (20P)
3. Check if frequency and physical cell identity match ALS (8P)



Rouge Base Station Detection

With Apple Location Services (ALS)

1. Confirm existence of cell with ALS (20P)
2. Calculate distance between recorded and ALS location (20P)
3. Check if frequency and physical cell identity match ALS (8P)
4. Low Bandwidth (2P)



Rouge Base Station Detection

With Apple Location Services (ALS)

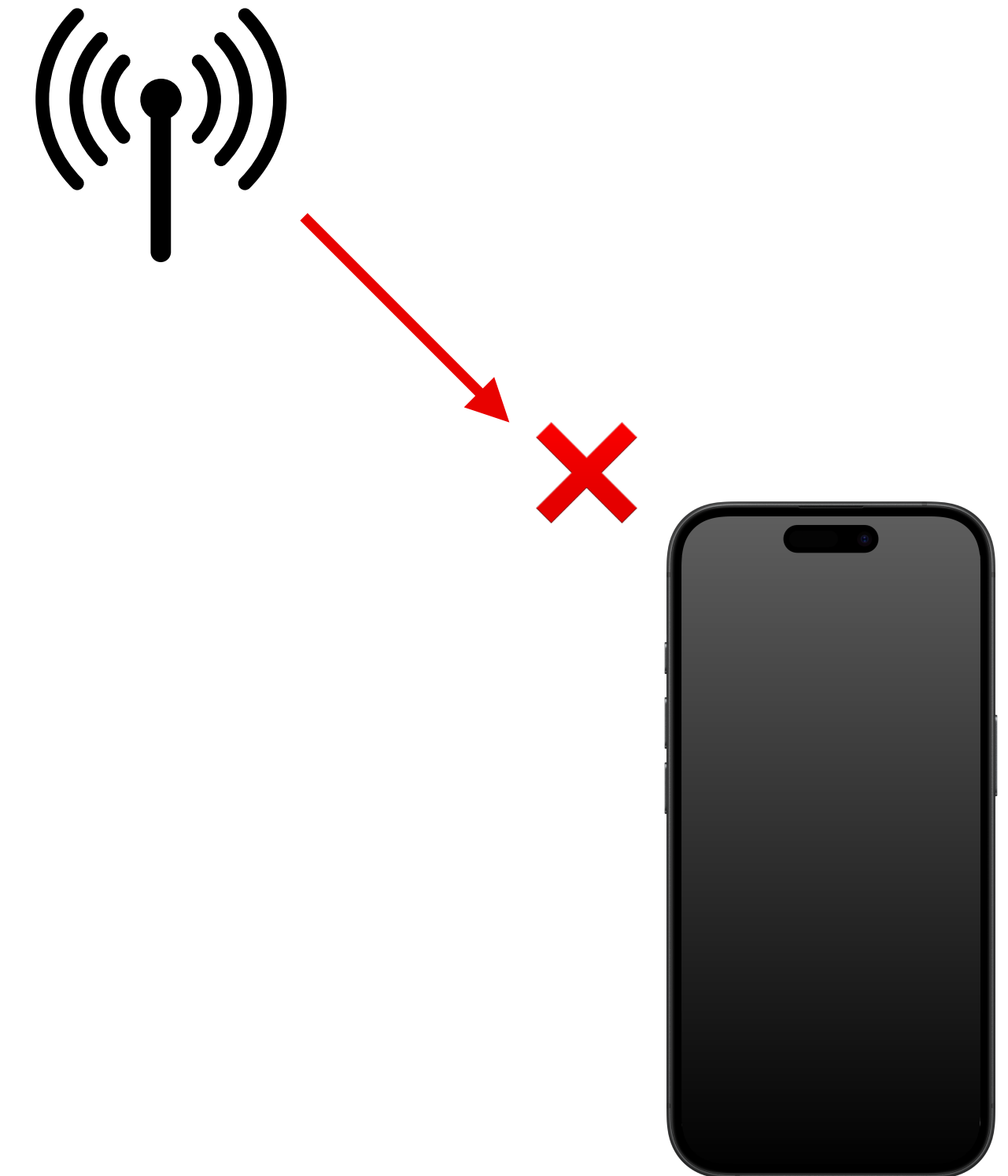
1. Confirm existence of cell with ALS (20P)
2. Calculate distance between recorded and ALS location (20P)
3. Check if frequency and physical cell identity match ALS (8P)
4. Low Bandwidth (2P)
5. High Signal Strength (30P)



Rouge Base Station Detection

With Apple Location Services (ALS)

1. Confirm existence of cell with ALS (20P)
2. Calculate distance between recorded and ALS location (20P)
3. Check if frequency and physical cell identity match ALS (8P)
4. Low Bandwidth (2P)
5. High Signal Strength (30P)
6. Unexpected Network Reject (30P)



Rouge Base Station Detection

With Apple Location Services (ALS)

1. Confirm existence of cell with ALS (20P)
2. Calculate distance between recorded and ALS location (20P)
3. Check if frequency and physical cell identity match ALS (8P)
4. Low Bandwidth (2P)
5. High Signal Strength (30P)
6. Unexpected Network Reject (30P)



Verdict

Trusted (100P - 95P)

Anomalous (50P - 94P)

Suspicious (45P - 0P)



CellGuard
Portable Cellular Analysis

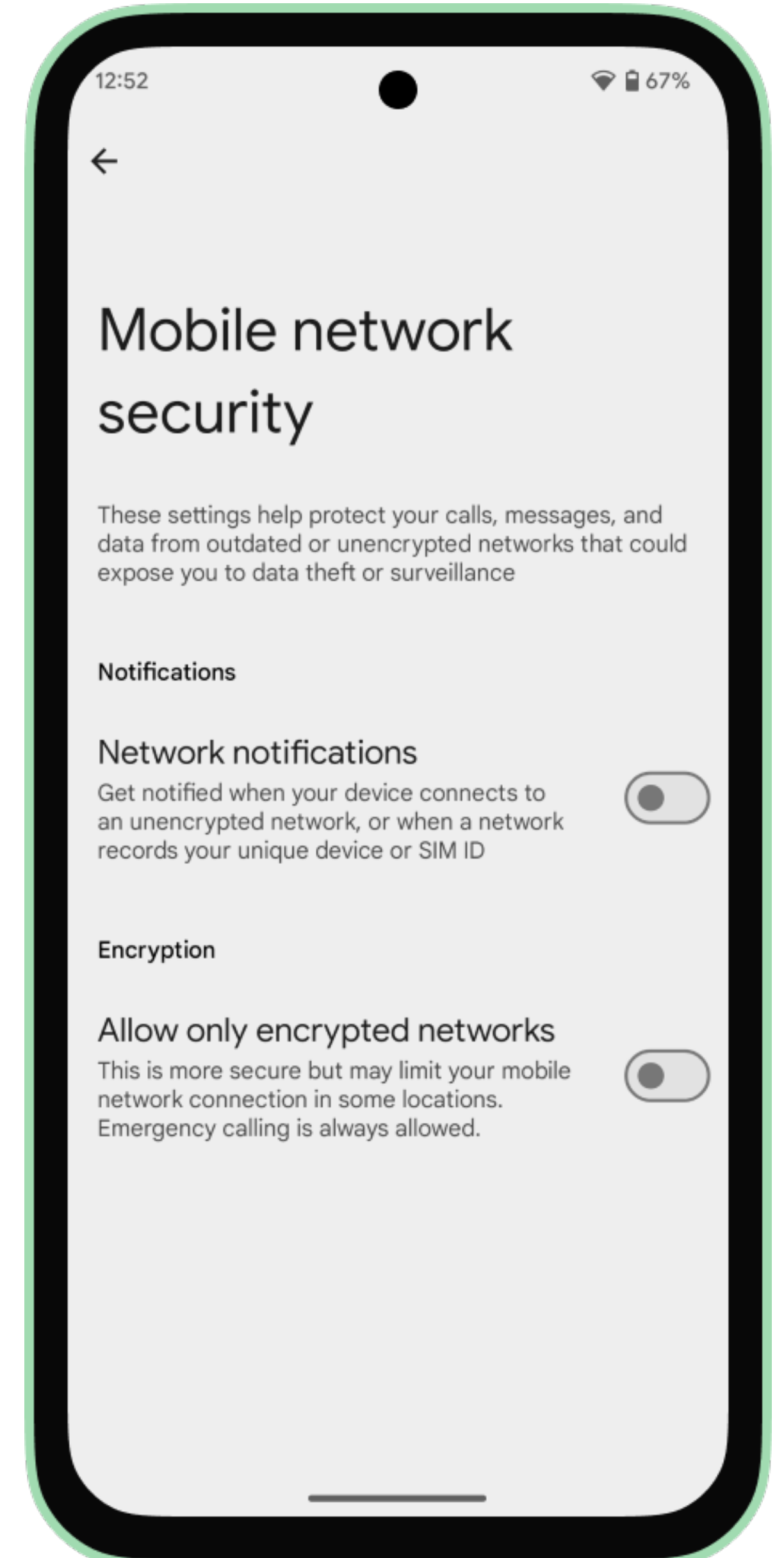


Mobile Network Security

New Security Features in Android 15

- Requires hardware support, so not yet deployed
- Deep integration with baseband and OS
- Can we also do it on iOS?

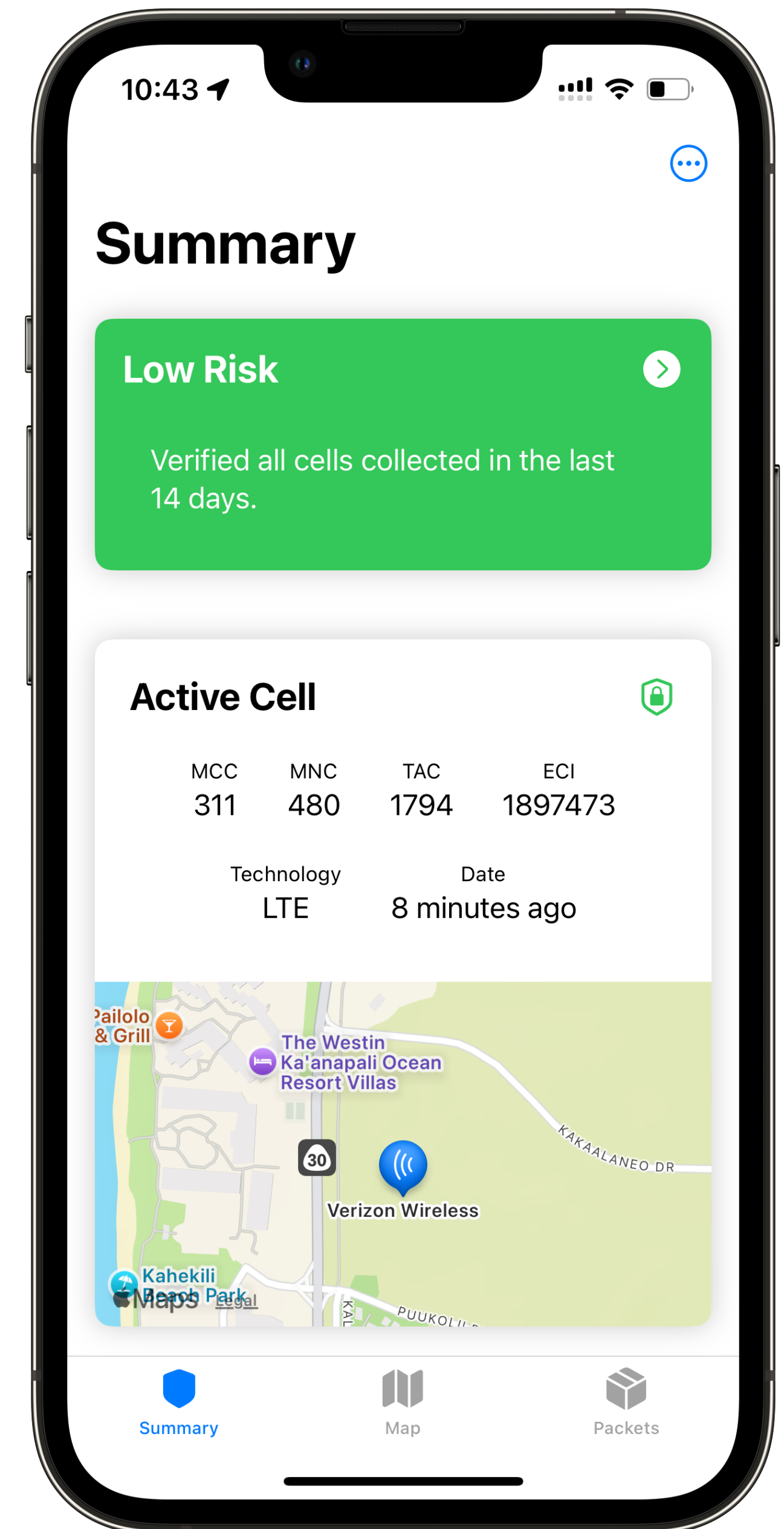
Android 



CellGuard for iOS

Mobile Network Security for iPhones

- RBS detection using baseband packets (QMI / ARI), locations, and Apple Location Services
- Our approach is limited compared to native integration but packets contains useful data
- Experimentation with different detection criteria
- Processing of all data on-device and deletion after few days

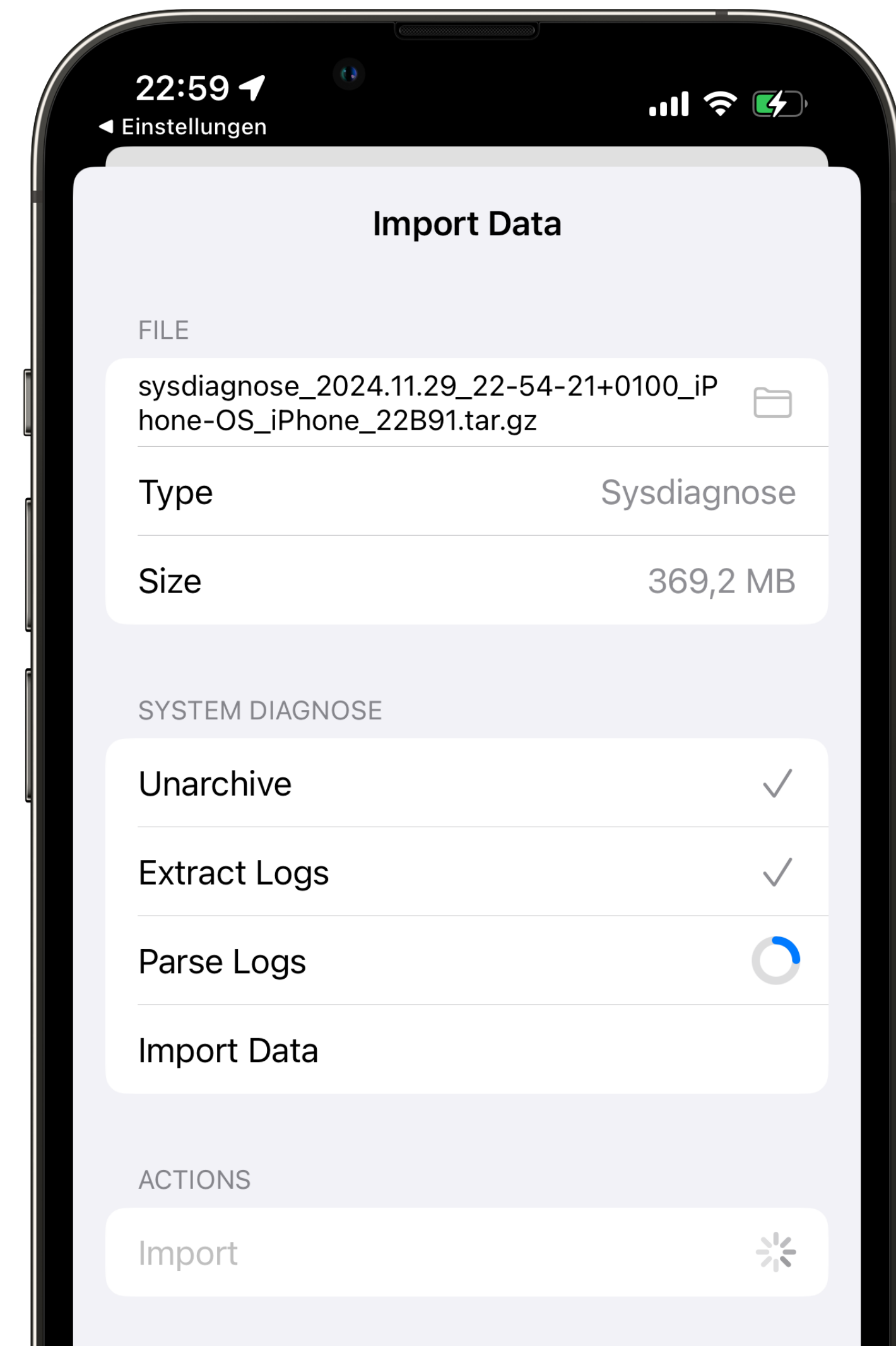


Data Extraction

On non-jailbroken iPhones



- Recommended for primary device with lockdown mode
- Users install baseband debug profile, capture sysdiagnose, and share it with CellGuard
- CellGuard extracts `system_logs.logarchive` and parses logs with macOS-unifiedlogs
- Extract baseband packets and connected cells from log
- Data of last 1h - 12h (depending on device activity)

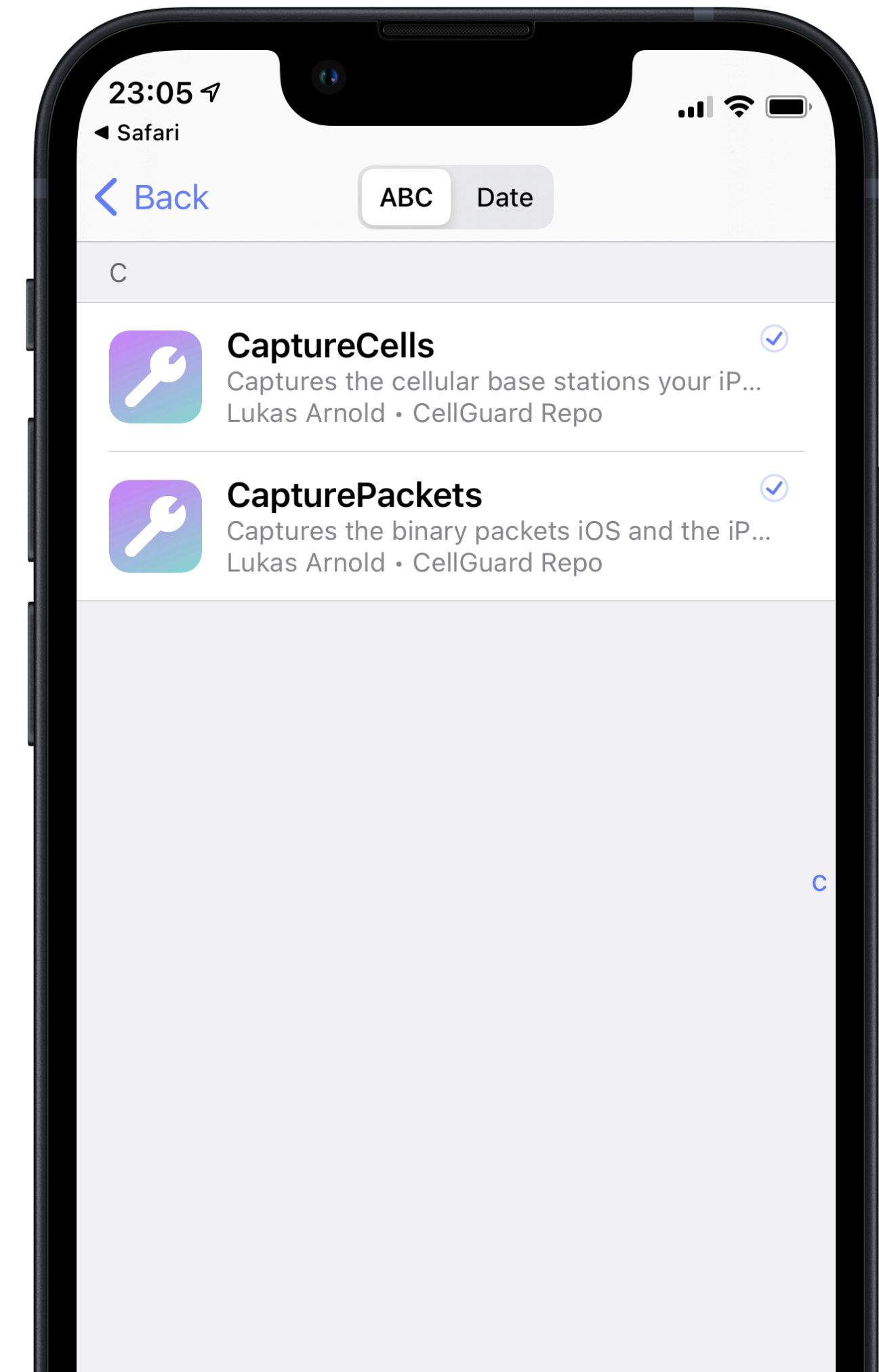


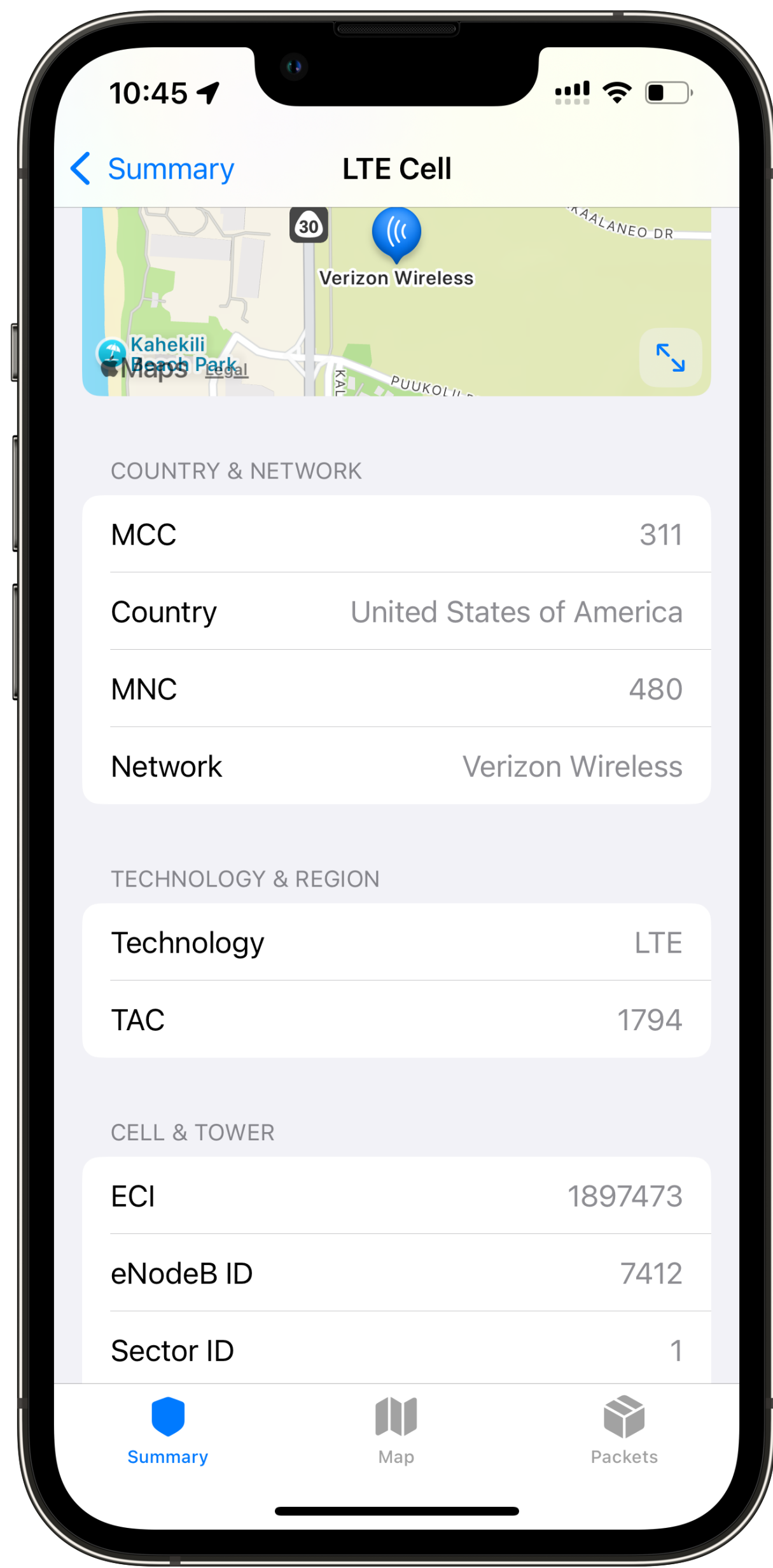
Data Extraction

On jailbroken iPhones

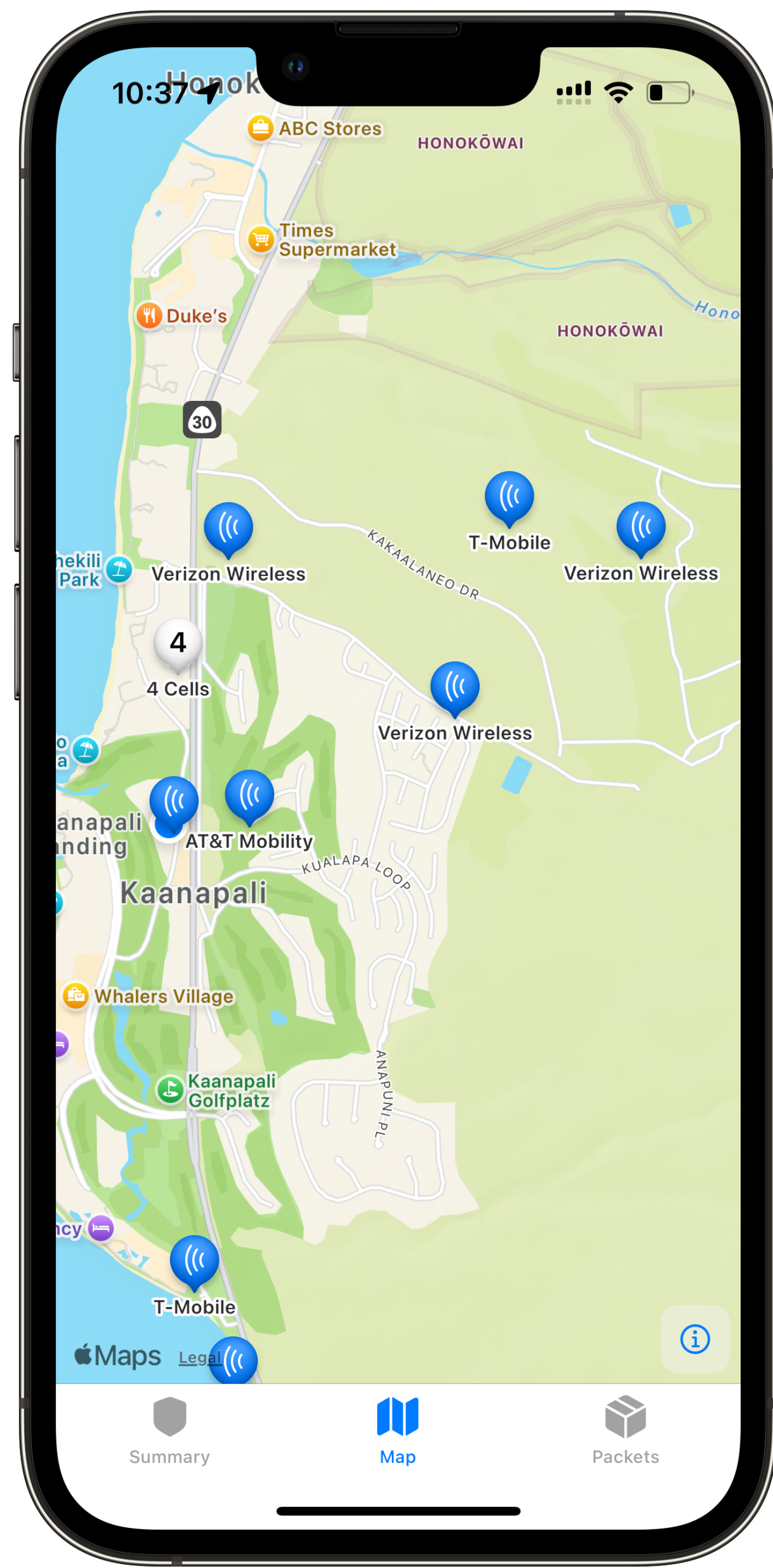


- Recommended on secondary device used for monitoring
- Automatic data collection and evaluation using tweaks
- Exchange data over TCP with CellGuard app

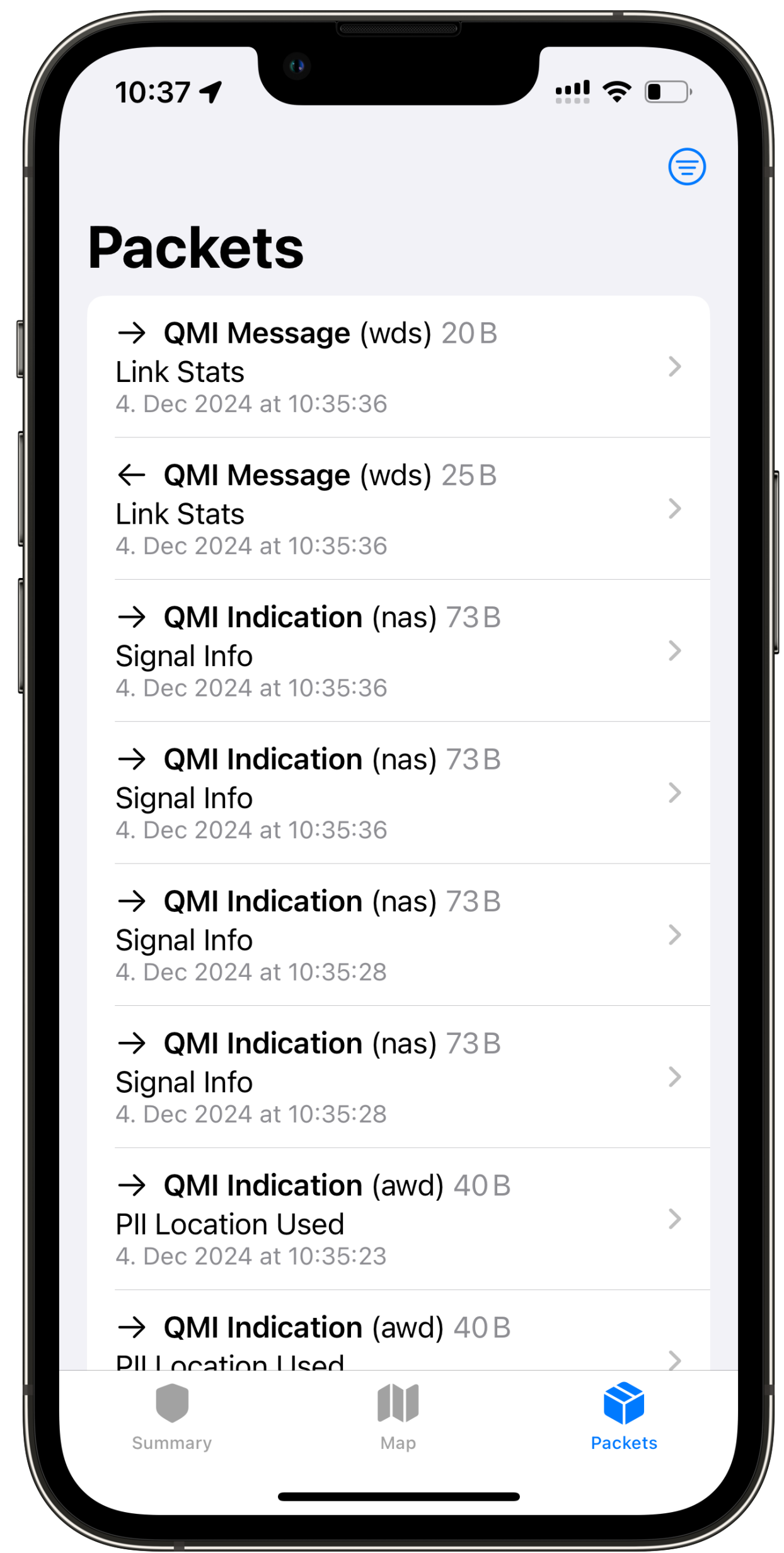




Dive into Details



Explore Nearby Cells



Dissect Packets

Evaluation of CellGuard

In our lab and in the wild

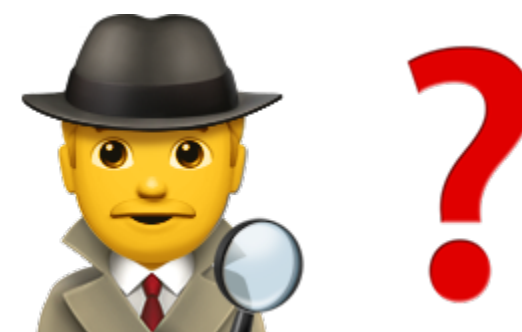


Datasets from across Europe collected over multiple months

1.6% anomalous
0.0% suspicious



Excellent coverage of Apple Location Services



Detection of anomalous activity but confirmation difficult



Lab setup with evil twin rogue base stations

CellGuard is Public

Join the beta and contribute to our large-scale study



Continuous development
of BaseTrace & CellGuard



Download CellGuard at
cellguard.seemoo.de



View the app's code
[seemoo-lab/CellGuard](https://github.com/seemoo-lab/CellGuard)

Thanks 🎉

Time for your questions

Contact

✉️ larnold@seemoo.tu-darmstadt.de

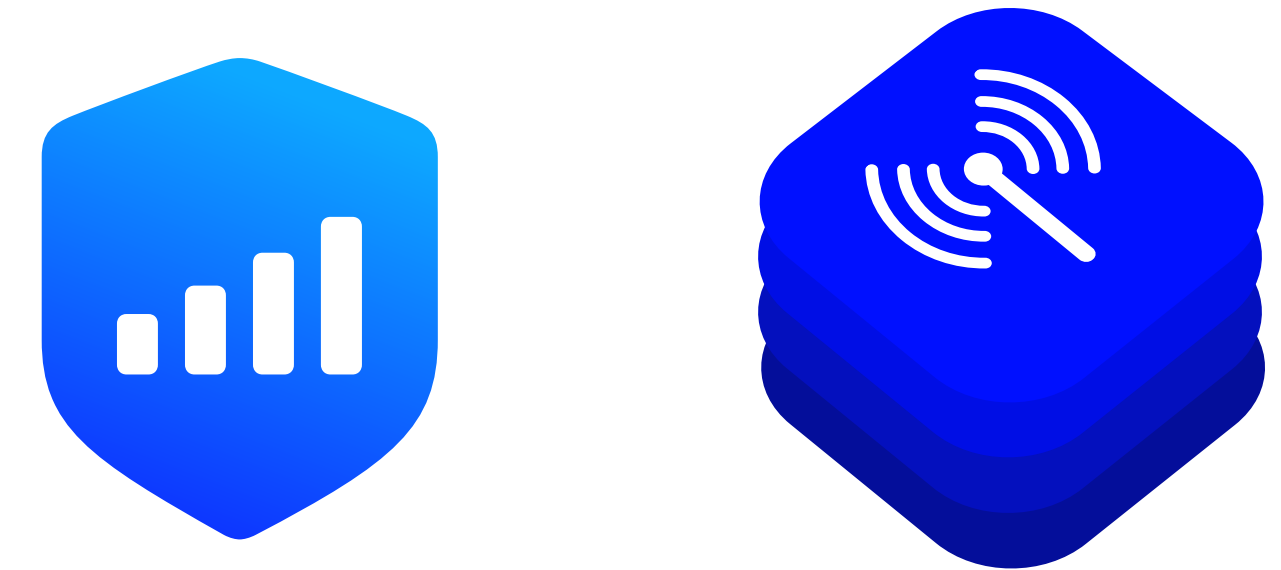
🐘 [@lukasarnold@mastodon.social](https://mstdn.social/@lukasarnold)

🦋 [@lukasarnold.bsky.social](https://bsky.social/@lukasarnold)

🐣 [@lukasarnold](https://twitter.com/lukasarnold)

Our Paper

Lukas Arnold, Jiska Classen, Matthias Hollick
Catch You Cause I Can: Busting Rogue Base Stations
using CellGuard and the Apple Cell Location Database



Download CellGuard