# Starshields for iOS: Navigating the Security Cosmos in Satellite Communication

Jiska Classen[*]
Hasso Plattner Institute
University of Potsdam
jiska.classen@hpi.de

Alexander Heinrich[*]
Secure Mobile Networking Lab
TU Darmstadt
aheinrich@seemoo.de

Fabian Portner
Secure Mobile Networking Lab
TU Darmstadt
fportner@seemoo.de

Felix Rohrbach
Cryptoplexity
TU Darmstadt
felix.rohrbach@tu-darmstadt.de

Matthias Hollick
Secure Mobile Networking Lab
TU Darmstadt
mhollick@seemoo.de

***Abstract*—Apple has integrated satellite communication into their latest iPhones, enabling emergency communication, roadside assistance, location sharing with friends, iMessage, and SMS. This technology allows communication when other wireless services are unavailable. However, the use of satellites poses restrictions on bandwidth and delay, making it difficult to use modern communication protocols with their security and privacy guarantees. To overcome these challenges, Apple designed and implemented a proprietary satellite communication protocol. We are the first to successfully reverse-engineer this protocol and analyze its security and privacy properties. In addition, we develop a simulation-based testbed for testing emergency services without causing emergency calls. Our tests reveal protocol and infrastructure design issues. For example, compact protocol messages come at the cost of missing integrity protection and require an internet-based setup phase. We further demonstrate various restriction bypasses, such as misusing location sharing to send arbitrary text messages on old iOS versions and sending iMessages over satellite from region-locked countries. These bypasses allow us to overcome censorship and operator control of text messaging services.**

## I. Introduction

Apple introduced satellite communication support with the iPhone 14 [5], allowing users to request assistance during emergencies in areas without cellular coverage. Further features include sharing the current location with friends over Find My [10] and requesting roadside assistance [9]. iMessage and SMS text messaging were added in iOS 18 [14].

Apple relies on the satellite network provided by Globalstar, a company operating satellites in Low Earth Orbits (LEOs) at around $1\,414\,\mathrm{km}$ height, orbiting the earth multiple times a day. We found that the oldest satellite used by Apple was launched in 2007, the same year the first iPhone was released [51]. Operating a 17-year-old infrastructure implicates technological challenges by itself [84], but Apple must also deal with satellites' fast movement during transmission, high delays, and generally low bandwidth requirements. At the same time, users need to be able to operate satellite communication intuitively under high stress in emergencies. This fact necessitates fast transmissions, ideally within a minute, and high satellite network availability. Apple's emergency SOS via satellite proved instrumental in saving lives during wildfires and hurricanes, when cellular network communication was no longer possible [52], [81].

Globalstar offers satellite network subscriptions directly to customers for location, text, data, and voice services [34]. In 2015, researchers revealed severe vulnerabilities in Globalstar's satellite communication protocol and released tools for decoding it [55]. We confirm that these attacks still work as of 2024 to intercept Globalstar's services. With this troubled security history, Apple invented a novel, proprietary protocol and only shares the Globalstar infrastructure. Apple's satellite protocol aims to protect highly privacy-sensitive data: A location shared via satellite should only be accessible to the designated recipients. More sensitive data leaves the phone during Emergency SOS via satellite: Health information from the user's medical ID [15], text messages, location information, and accurate information about the current emergency.

This work represents the first comprehensive investigation into Apple's satellite services' security and privacy. We reverse-engineer previously undocumented internals to answer the following research questions: ***RQ1:*** *How are security and privacy features implemented in this resource-constrained satellite communication environment?* ***RQ2:*** *Can users bypass service restrictions imposed by Apple?*

**Contributions.** At the time of writing, only limited related work on the security of end-device to satellite communication exists [55], [46], [84], [86]. We are the first to analyze satellite communication implemented in the iPhone. Our main contributions are:

[*]Both authors contributed equally to this research.

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240124
www.ndss-symposium.org

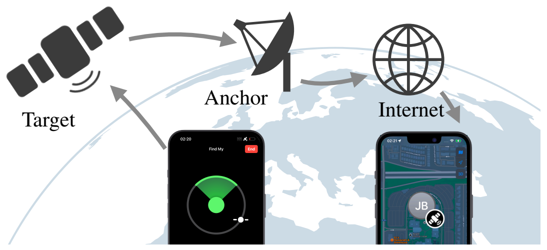


Fig. 1: Communication principle for Find My location sharing.

- We reverse-engineer the protocols for Emergency Texting, Roadside Assistance, Find My, iMessage, and SMS over satellite.
- We show how data exchanges are minimized and compressed to meet technological limitations, including a complex scheme to set up various encryption parameters when the user has a regular internet connection.
- We analyze the cryptographic properties of the transmission of privacy-sensitive data.
- We create an app to send and receive arbitrary, unlimited SMS-like messages over Find My location sharing (Demo: https://youtu.be/3qYby6CeBxs).
- We demonstrate that various restrictions are only implemented on the client side, allowing adversaries to bypass regional restrictions and enable satellite services in countries where they are not supported.
- We build a simulation-based testbed to analyze satellite features in iOS without communicating with actual satellites (Demo: https://youtu.be/igaYCdnqdjE).
- We propose mitigations to design secure satellite communication systems.

**Responsible Disclosure:** We responsibly disclosed our findings to Apple, allowing for improvements in their satellite communication system. As of July 2024, Apple imposed sharper restrictions on text messaging over Find My, but their system architecture prevents a complete resolution.

**Ethical Considerations:** Emergency SOS over satellite allows a user to contact first responders without cellular coverage. During our tests, we never alarmed any first responders. Instead, we built a simulation-based testbed to safely study satellite features without resulting in real-world emergency calls. Furthermore, we never bypassed regional restrictions in countries where satellite communication is illegal.

**Availability:** Our Artifact Appendix includes an app allowing text messaging via satellite, the Wireshark dissector for the satellite protocol, and scripts to setup our test environment. For ethical reasons, we do not publish code for the client-side communication restriction bypasses we demonstrate.

## II. Background

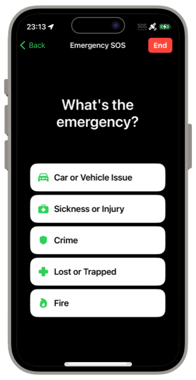
This section describes satellite communication basics with a focus on Globalstar's setup. This setup differs from other popular satellite networks, thereby creating novel challenges to be solved by Apple's protocol stack. Then, we dive into Apple's satellite features from a user's perspective.

### A. Satellite Communication

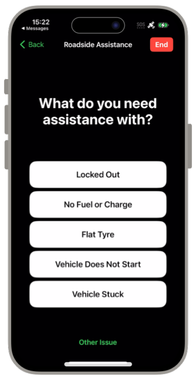
**Satellite Networks for Smartphones:** In 2022, Apple was the first to launch a high-end smartphone with satellite functionality covering the U.S. , Canada, and Europe [5], [8]. Around the same time, Huawei launched the Mate 50 with satellite SMS support in China [80], followed a year later by the Mate 60 Pro, which introduced satellite phone calls [43]. In 2024, the Google Pixel 9 received an US-only emergency SOS feature via the Skylo satellite network [74]. AT&T partnered with AST SpaceMobile to create a new satellite that can send 4G and 5G signals to Earth, reaching download speeds of up to 14 Mbit/s [71]. T-Mobile, together with Starlink, also announced satellite support for 5G smartphones in 2022 [79], with the first satellite launched in early 2024 [25]. These projects share a fast-paced development of new features combined with infrastructure updates. Their technology stacks fundamentally differ: Apple uses legacy satellites but updated the iPhone's cellular modem and communication stack; SpaceX updated its satellites to enable off-the-shelf phones to communicate. Apple's approach to using legacy satellites is arguably the most challenging from a technological perspective, with tight constraints that leave little room for security and privacy. Apple partnered with Qualcomm for a custom modem supporting satellite communication [59], developed a customized antenna to boost outgoing signals, and added custom hardware to ground stations to handle incoming data directly. In this work, we focus on Apple's implementation.

**Globalstar Satellite Communication Principle:** Globalstar satellites employ a bent-pipe architecture [34], acting as mirrors in space. They forward an iPhone's signal to a ground station and vice versa (see Figure 1). A satellite requires the iPhone and the ground station to be within range to establish a communication channel. Ground stations are connected to the internet, allowing interaction with services provided by Apple. In Apple's terminology, satellites are *targets* the user points their iPhone at, and ground stations are *anchors*. Globalstar's constellation consists of 48 satellites, with an additional four spare satellites [58]. The satellites orbit the earth in LEO at an altitude of 1 414 km, which equates to an orbital period of about 2 h or 12 turns around earth per day [33]. According to our analysis, as of July 2024, Apple uses 28 Globalstar satellites. The oldest satellite in use is *M069*, launched on May 29, 2007, and the newest one is *M097*, launched on February 6, 2013. Globalstar launched a newer satellite, *M087*, on June 19, 2022, which Apple is yet to use. Globalstar operates satellites from two generations: the first launched before October 2010 with a life expectancy of 7½ years, and the second launched with a life expectancy of 15 years [28]. Consequently, some satellites are already past their expected lifetimes, increasing technological challenges in working with legacy hardware. Interestingly, Apple uses satellites from both generations. To future-proof the constellation, Globalstar ordered 17 new satellites, which are scheduled to launch in 2025 [68].

**Satellite Orbits:** The position of objects orbiting the Earth, such as satellites or space debris, is commonly specified by a



(a) Emergency SOS



(b) Roadside Assistance

Fig. 2: Satellite emergency communication questionnaires.

set of parameters called orbital elements. In 1980, researchers published the Two-Line Element (TLE) orbital element format, together with a series of orbit models known as simplified perturbation models [42]. A TLE consists of two lines with 69 characters each for punch card compatibility and precisely encodes an object's orbit and position therein at a specific time, known as the epoch. Using a simplified perturbation model, with the de-facto standard being SGP4 [83], one can propagate a TLE to any point in time in the past or future. However, predictions gradually diverge from reality [27]. The TLEs must thus be updated regularly to ensure accurate predictions.

### B. *iPhone Satellite Features*

The iPhone 14 was the first device with satellite connectivity. Since then, Apple has added more features with software updates. All satellite features mandate that Wi-Fi and cellular connectivity are unavailable.

**Find My Friends:** Find My Friends allows people to share their location with friends and family [10]. The user has to set up location sharing while they have an active internet connection. Next to sharing the location via internet, Apple added the option to share a location via satellite. Users can initiate location through the Find My app and their friends can view the shared location only when they have an internet connection. When the iPhone has transmitted a location update via satellite, it prohibits the user from sharing another location for a cooldown period of $15\,\mathrm{min}$.

**Emergency SOS:** In the case of an emergency, users can contact emergency services via satellite [17]. Emergency SOS starts with a questionnaire (see Figure 2a). These questions help describe an emergency, ensuring no essential details are overlooked and reducing follow-up questions. People can continue to communicate with first responders using text messages. The maximum size for one text message is $160\,\mathrm{byte}$, and the input is limited to text only. If the iPhone detects that the user fell or might have had a car accident, it automatically initiates an emergency call [16], [18]. In case of no cellular coverage, the iPhone falls back to SOS over satellite and offers the possibility of follow-up emergency texting. Falls or crashes might leave users unable to point their iPhones toward a satellite. Still, their orbiting nature increases the chance that the Emergency SOS is sent eventually.

**Roadside Assistance:** In 2023, with iOS 17, Apple added roadside assistance for U.S. customers [9]. When typing *Roadside Assistance* into the recipient field of the *Messages* app, the user is asked to select one of the assistance providers. Then they go on and specify the issue they are facing, as shown in Figure 2b. Optionally, the user can add a phone number. Then, they are again presented with the interface to help them establish a satellite connection and send text messages.

**Messaging:** iOS 18, released in fall 2024, supports iMessage and SMS over satellite [14]. Like roadside assistance, messaging is only available in the U.S. SMS messages are restricted: Users can only send and receive messages if they have already messaged the recipient in the past or they are an emergency contact or family member.

## III. Related Work

More than 100 satellite hacking incidents have occurred in the last 60 years [64]. The security research community has contributed to uncovering more security problems and proposed solutions in recent years, summarized in the following.

The low-latency promises of LEO satellite constellations were vulnerable to Denial of Service (DoS) attacks [31]. After classifying further attacks, several solutions were proposed for LEO satellite services [88]. Many satellite internet services are based on the DVB-S protocol, originally designed for video broadcasting. Most services did not use encryption at all or their encryption was weak, allowing for eavesdropping attacks [65], [66], [87], [26]. Furthermore, Jedermann et al. [46] showed that location privacy in LEO satellite communications can be compromised by observing satellite transition events, i.e., when the satellite a user is communicating with changes due to its orbit. Since their attack requires several hours of satellite communication and the iPhone only uses short-lived communication, it is not applicable.

Keeping up with the latest security developments is a significant challenge for satellite hardware. While expected lifetimes are on the order of 10 to 20 years, this hardware typically operates well beyond that timeframe. Previous research has uncovered serious security flaws in satellite software [84]. Satellite terminals operating on the ground also had major security issues. In 2022, an attack on Viasat satellite terminals, likely aimed at Ukraine's satellite internet communications, also impacted and disabled large parts of Germany's wind turbine monitoring and control system [36]. The control infrastructure of the satellite terminals had multiple vulnerabilities that allowed adversaries to compromise the satellite modems, disable recovery mechanisms, and perform DoS attacks [85]. Yu et al. [87] created an automated test setup, evaluated nine terminals, and found 18 novel attacks. Modern terminals for Starlink have additional security mitigations but remain vulnerable to fault injection attacks [86].

The first sniffing and injection attacks on the Globalstar network were presented in 2015 [55]. However, we found that Apple's communication protocol is fundamentally different at all layers compared to Globalstar's legacy protocol and other existing satellite protocols: It uses a different modulation scheme, adds encryption, and includes application-specific compression. As a result, we are the first to reverse-engineer the satellite communication protocol used in iPhones and identify a number of novel attacks that allow adversaries to bypass restrictions and abuse the satellite services provided by Apple and Globalstar.

Apple's Find My network implementation over Bluetooth Low Energy (BLE) has been studied extensively [38], [23], [39], [72]. Nevertheless, neither Find My location sharing over the internet nor its satellite version have been analyzed before. We find that both versions of Find My location sharing are similar, but the location data is shortened to transmit only the essential information. Similar to the BLE-based Find My network, elevation, speed, and other details are redacted [39].

## IV. Methodology

### A. Reverse Engineering

Reverse engineering Apple internals is challenging, even with access to a jailbroken iPhone or a Security Research Device (SRD). While Apple releases source code for some of their subsystems [6], highly proprietary components, such as satellite communication, remain a secret until security researchers analyze them. We describe our analysis approach, which allows the reproduction of our results and protocol insights. We generally use similar tools as previous work on reverse engineering the Apple ecosystem [40], [41], [49], [75], [70], [24]. Oftentimes, reverse engineering is a manual task, which cannot be automated, if one wants to uncover the full protocol and all system components involved. Automated testing and fuzzing approaches can be applied in some cases, as done for satellite terminals [87], but are not applicable for our work.

**System Components:** Find My over Satellite is implemented within multiple components of iOS. Identifying system components and their interfaces is crucial to understanding proprietary features. Public and private frameworks abstract functionality executed by user-space daemons or dispatched to hardware via the kernel. To unveil which system components are involved, we investigate logs. The iPhone under analysis does not need to be jailbroken for this. Debug profiles increase log verbosity [7]. By observing logs, we find that the daemons responsible for the core functionality are *CommCenter*, and in the case of Find My, also *searchpartyd*.

The *CommCenter* daemon instructs the baseband chip to communicate over satellite. It is the core of the satellite communication system, holding all states about currently ongoing emergency conversations, taking care of regional restrictions, providing the baseband chip with all necessary information, and parsing the baseband's responses. The *SOSBuddy* app is the user interface for satellite communication. It instructs users where to point their phones and leads them through the emergency questionnaire. *SOSBuddy* is launched by *CommCenter* and primarily displays its internal state for end users. The Demo mode in the *Settings* and *FindMy* send a request to the satellite subsystem in *CommCenter*, which in turn opens *SOSBuddy*. The *searchpartyd* daemon handles necessary tasks for Find My in satellite communication. E.g., it creates and encrypts location data before sending it via satellite. The *identityservices* daemon coordinates keys for sending iMessages and SMS over satellite.

**Communication Protocol Analysis:** Data sent over Qualcomm MSM Interface (QMI) contains privacy-sensitive information. Thus, by default, QMI log messages are replaced by the tag `<private>`. Apple provides a baseband debug profile, which overcomes this limitation [7]. After installing the profile on a non-jailbroken iPhone, a byte-wise representation of QMI messages is displayed in the logs. We then observe the logs with the macOS *Console* app or perform a system diagnosis on the iPhone [19]. The latter allows for testing satellite communication in the field when no laptop is connected and for capturing logs for later analysis.

To inspect the QMI messages, we adapt a *libqmi*-based Wireshark dissector [20], [56]. It already includes most message names, which are auto-generated based on iOS' shared libraries. With further reverse engineering, we add field names and reconstruct the protocol flow. Specifically, protocol fields are located in the DYLD shared cache library `libCommCenterMCommandDrivers.dylib`. All fields follow the naming scheme `tlv::sft::*`. This approach provides us with further insights about the communication.

**iPhone 13 Satellite Simulation:** We create a safe satellite simulation environment to enable using an iPhone 13 that does not include satellite hardware and cannot alarm first responders. As iPhones using the same iOS version share most parts of the code, the simulation environment is an accurate representation of the satellite communication happening on the device itself. We use Frida [60] scripts to hook relevant parts within *CommCenter*, which allows mimicking communication with a satellite. This setup requires a jailbroken iPhone or an SRD. In the following, we detail the required modifications.

Upon initialization, *CommCenter* determines its so-called radio personality, which configures baseband features. Satellite communication requires two properties: `MobileGestalt` has a device-specific hardware configuration number, which must be set to `0x34`, and the system `FeatureFlags` must support *Bifrost*, which is a code name for the iOS satellite communication subsystem. With these two properties set, the baseband is initialized with the so-called *Stewie* service responsible for satellite transmissions. Furthermore, other services regularly query *CommCenter* to check whether *Stewie* is enabled. They send a Cross-Process Communication (XPC) message requesting the property `getStewieSupport`, which we overwrite in our scripts always to return `true`. When initiating satellite communication, *CommCenter* checks if a service is allowed, given the current preconditions. E.g., even if there is no cellular coverage, Emergency SOS is not allowed until a cellular emergency call fails. *CommCenter* keeps track of currently allowed services in an internal state struct, represented as a service bitmask (see Appendix B). We overwrite the service mask every time it is checked against currently allowed services, effectively enabling us to test all services regardless of any preconditions. With these patches in place, *CommCenter* attempts satellite communication, even on a device that does not support it. We further modify the functions `pci::transport::th::writeAsync`, which is responsible for writing QMI packets to the baseband, and `QMux::State::handleReadData`, which parses packets originating from the baseband. These two functions are the core of simulating the full satellite functionality on the iPhone 13. We re-implement the protocol logic described in Section V-C, such that the iPhone 13 can complete authentication and registration and receive faked message transmission confirmations. Implementing the logic based on Find My transmission logs enables the emergency texting protocol stack and allows us to test without alarming first responders.

We publish our scripts as part of our artifacts [37]. The scripts enable other researchers to use the same safe satellite simulation environment. Our main script allows for debugging and change of various properties for testing. This includes usage of encryption keys, efficiency of text compression, and entering Emergency SOS without dialing an emergency phone number.

### B. *Experimental Setup*

After the iPhone 14 release, it took 1½ years until jailbreaks became available [30]. Thus, we used different research options and built custom tools to analyze Apple's satellite communication subsystems. We create a step-by-step guide to recreate our experimental setup.

**1) Sending Satellite Messages:** Satellite communication is enabled while the iPhone is within a supported country. Satellite services become available when no other cellular or Wi-Fi connection is possible. Deactivating Wi-Fi and using an expired SIM card enables this required state. For our tests, we primarily use Find My location sharing and observe the behavior of the device using the following steps.

Due to ethical considerations, we did not initiate emergency calls but instead relied on different reverse engineering approaches, such as our simulation environment (see Section IV-A). Sending iMessage and SMS over satellite was not available when writing this paper. While announced for iOS 18 [14], this feature is not supported outside the U.S. We find bypasses for regional restrictions (Section VI-E) and a service bitmask to enable disabled services. With these, we can also test the remaining services, including iMessage over satellite.

**2) Recording Signals with Software-defined Radios:** Software-defined Radios (SDRs) can observe wireless transmissions and send arbitrary signals in the frequency bands the Globalstar satellite network uses. Sniffing radio signals shows messages sent to or received from a satellite. Recording the satellite communication requires a clear view of the sky. We use the BladeRF [61] for a portable, USB-powered setup. The uplink channel operates in a licensed frequency band virtually exclusive to Globalstar, ensuring minimal interference with other radio signals. The uplink signal is strong; we can easily capture it with regular Wi-Fi antennas.

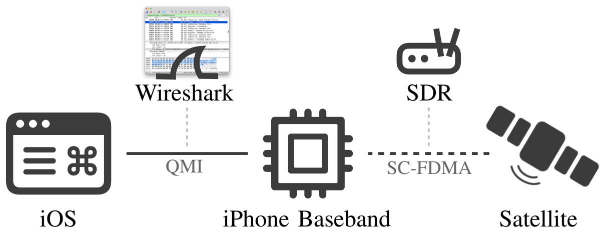


Fig. 3: Intercepting satellite communication.

**3) Intercepting On-device Baseband Communication:** The Qualcomm baseband chip on the iPhone 14 and newer supports satellite communication. As shown in Figure 3, when iOS sends messages to a satellite, these go to the baseband chip first, which takes care of over-the-air transmission. The interface between iOS and the baseband chip is called QMI. We use a reverse engineering setup that allows us to observe QMI protocol messages as they happen, even on non-jailbroken phones (see Section IV-A).

**4) Security Research Device:** Apple has a SRD program [11], providing devices for research even without public jailbreaks. However, only previous-generation iPhones are distributed as SRD, initially limiting us to an iPhone 13 mini without a satellite-compatible baseband. The later provided iPhone 14 SRD is a South Korean model, which disables Find My Friends due to legal restrictions. Nonetheless, Apple's SRD program enables us to modify the software state on iPhones. We modify the iPhone 13 to simulate satellite behavior (see Section IV-A). Moreover, we bypass the model-specific South Korean restrictions on the iPhone 14 (see Section VI-F). Although the phones were South Korean models, we never used them in South Korea and ensured that we followed all local regulations. Both setups are helpful for security research on the satellite protocol. The satellite simulation on the iPhone 13 allows us to test emergency communication safely without alarming first responders. In contrast, the iPhone 14 setup enables testing with physical satellite infrastructure.

## V. iOS Satellite Communication

Apple gives away no information on the protocol used for satellite communication and the security properties of it. We reverse-engineered the entire protocol flow on the iPhone and present it in this section.

### A. *Initial Configuration*

The iPhone requires an unobstructed line of sight to the satellite and down from the satellite to the ground station. The iPhone calculates the positions of suitable satellites (see Section II-A) to guide the user in pointing their phone in the right direction. The required information for these calculations is downloaded and updated by the *Trial* service, requiring an active internet connection.

Two trial configuration files are present for satellite communication: An XML property list and a Distinguished Encoding Rules (DER) encoded version. Both files contain the same content, such as ground station locations, countries where satellite communication is enabled, communication features, and frequency bands. Information is provided per country, allowing to geofence satellite services in software as required by Federal Communications Commission (FCC) [29], and limiting certain satellite services only to selected countries. The DER-encoded file is required to share the same configuration with the baseband chip over QMI.

Based on the configuration update from May 3, 2024, Figure 4 shows countries in which satellite satellite services are enabled, along with ground station locations. Within the 16 allowed countries, there are radio exclusion zones, where transmissions are forbidden even if there is satellite coverage. Radio exclusion zones are near astronomy sites where transmissions could disturb observations [29], some islands, and national border areas. 19 ground stations are active, with ground station indices indicating that there might be 63 ground stations in total. Ground stations are located across the globe, even in regions without satellite support, including Mexico, Brazil, Singapore, and Estonia. All countries use the same 7 channel numbers for communication: 262 336, 262 338, 262 340, 262 342, 262 344, 262 346, 262 348. The signal uses a bandwidth of 200 kHz. Only even channel numbers are specified, as each number represents 100 kHz.

The iPhone maintains a list of TLEs from which it can propagate the position of satellites. TLE data is maintained by the iPhone in a target property list containing a total of 15 entries. Each entry is valid for two days and contains TLEs for all 28 satellites currently used by Apple. These files allow the iPhone to calculate the satellite trajectories 30 days in advance. Nevertheless, targets are updated daily if an internet connection is available. When offline for more than 30 days, an iPhone loses satellite connectivity because it can no longer calculate satellite trajectories. When preparing a satellite transmission, iOS transforms the currently valid targets list entry into a QMI message for the baseband chip.

Globalstar satellite data, including identifiers, launch date, and TLEs, is publicly available [47], [58]. The known satellites follow a naming scheme with the prefix *M*, followed by a three-digit satellite number. We confirm that the satellite numbers used in the Targets list match this scheme with the experiment described in Appendix A.

## B. *Key Material Setup*

An iPhone must register a set of keys with Apple before performing satellite communication. Internally, these keys have multiple terms: Link Layer Communication (LLC), transport, or session keys. We refer to them as LLC keys. This section covers the key generation to the final key exchange steps. Figure 5 shows a simplified sequence diagram of this procedure.

**Key Generation:** ① The iPhone generates 30 private-public key pairs on the `NIST-P256` curve. The keys are generated inside the Secure Enclave Processor (SEP), a co-processor similar to a Trusted Execution Environment (TEE) on Android [4]. Processes running on the main processor can only indirectly access keys on the SEP through predefined functions, such as copying the public key and elliptic-curve Diffie-Hellmann (ECDH) key exchange. For each key, the iPhone performs a SHA256 hash of the public key and uses the first eight bytes as a key identifier termed Ephemeral Public Key Identifier (EPKI). After initial generation, all new 30 keys are marked as *proposed* and cannot yet be used for satellite communication.

**Key Synchronization:** ② After local key generation, the keys must be synchronized with Apple's servers, when the iPhone has an active internet connection. The iPhone sends its public keys to Apple over a REST API as an HTTPS request. This request contains a list of all LLC public keys and a list of allowed services for which these keys can be used. On iOS 16, these services are `networking.st.text-911` and `networking.st.find-my`. iOS 17 adds the `networking.st.roadside` service, and iOS 18 adds the `networking.st.imessage-lite` and `networking.st.sms` services. In addition, the request contains information about the current device and the user's Apple ID. The server links the keys to the

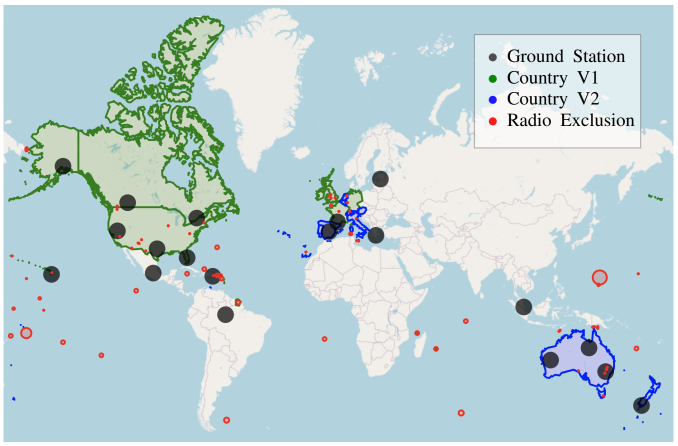
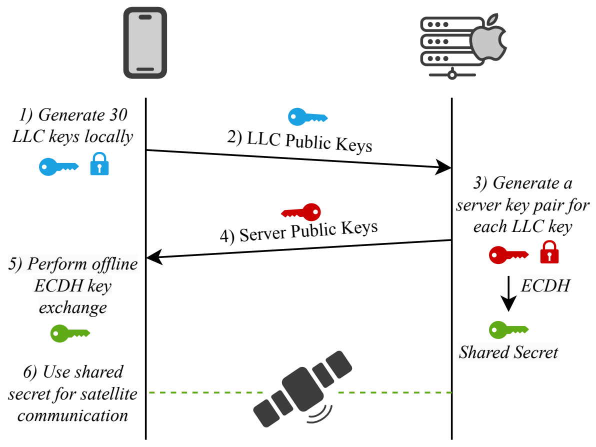


Fig. 4: Ground station and country configurations.


1) Generate 30 LLC keys locally
2) LLC Public Keys
3) Generate a server key pair for each LLC key
ECDH
Shared Secret
4) Server Public Keys
5) Perform offline ECDH key exchange
6) Use shared secret for satellite communication
Fig. 5: Key synchronization procedure with Apple before an iPhone can perform satellite communication.

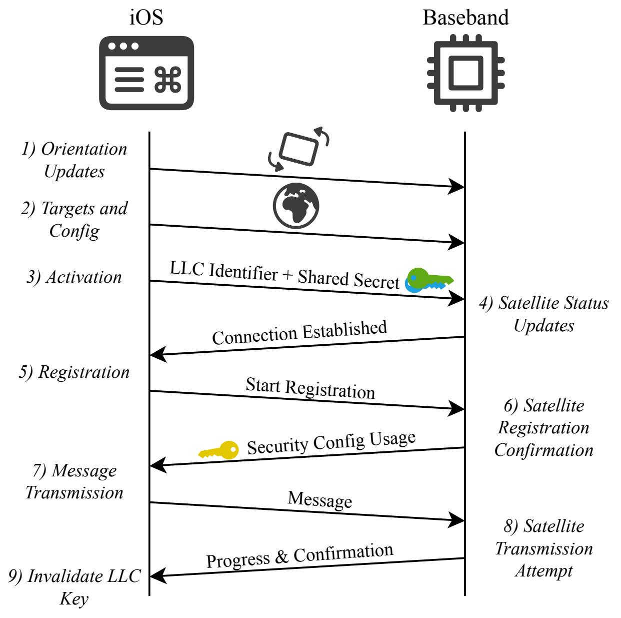


Fig. 6: QMI messages exchanged with the baseband chip during a satellite transmission.

user's Apple ID, which allows for the addition of subscription-based services in the future. One key is marked as the *last resort* key, allowing the iPhone to continue using satellite communication if all prior keys were already used.

(3) Apple's server generates a new server key pair for each LLC public key received. (4) Apple responds with all *server public keys*. The iPhone stores keys in the system's keychain together with the LLC key identifiers under the term *STKData*, likely an abbreviation for Satellite Transmission Key. Afterward, the iPhone marks the LLC keys as *settled*.

**Key Exchange and Transport Encryption:** (5) When communicating via satellite, the iPhone performs an offline ECDH key exchange using one LLC key and the matching server public key. This is a standard key exchange over the `NIST-P256` curve as specified by NIST [21]. Note that the authentication of the keys is implicit, as they are shared over a secure, authenticated channel. The resulting shared secret can now be used for symmetric encryption, authentication, and integrity protection.

(6) The shared secret and the EPKI for the LLC key are transferred to the iPhone's baseband chip using a QMI message. When the baseband successfully connects with the satellite, the associated keys are marked as *invalidated* and deleted afterward. Each LLC key and the corresponding shared secret are only valid for one communication session. When using Find My over Satellite, a communication session equals a single message, while multiple messages are exchanged for text messaging services. A new LLC key for each *invalidated* one is generated when the iPhone has an internet connection.

### C. Satellite Communication Protocol

The satellite transmission consists of multiple protocol steps. iOS does not directly communicate with satellites but uses the Qualcomm baseband chip for this task, which adds a layer of abstraction. iOS sends QMI requests to the baseband chip to start certain actions, which the baseband responds to. In addition, the baseband can send asynchronous events using QMI indications [35]. Satellite communication is an Apple-specific QMI extension, which we observe with our reverse engineering setup (see Section IV-A). Figure 6 contains the underlying protocol steps.

**Setup Phase:** (1) Successful transmissions require the user to point their iPhone towards a satellite. The baseband does not have a gyroscope, so iOS regularly informs the baseband of its current orientation. This information is used when satellite communication is initiated, with further orientation updates supplied when needed. (2) The baseband requires additional information to calculate satellite positions and direct its signal. iOS sends the TLE-encoded target list and the DER-encoded config to the baseband, enabling the baseband to perform calculations.

**Security Configuration:** (3) Satellite communication requires a valid LLC key provisioned by Apple. In the activation step, iOS sends the 8 byte EPKI, uniquely identifying the key on the iPhone's local database as well as on Apple's servers, and the shared secret, derived using ECDH from the LLC key, which is a 256 bit symmetric key. This is configured along with further information, such as the user's country and time zone. The following communication between the baseband and the satellite might be encrypted based on the shared secret. Apple confirmed this encryption layer to us in a discussion on reported issues (see Section VI-B).

(4) The baseband indicates regular updates about the optimal satellite identifier, signal strength, and transmission progress. Status updates tell iOS when it can proceed with the next protocol steps. (5) Once a satellite is in range, the protocol continues with a registration based on the activation information. A ground station answers the registration, which can take a while, depending on transmission conditions. (6) The *Security Config Usage* indication confirms a successful registration. This confirmation contains the same EPKI as in the activation step. Moreover, it contains a fresh symmetric 256 bit key, internally called *GeneratedAppKey*. This key is generated for all communication modes and used as *Master Session Key* for emergency texting. When the iPhone runs out of LLC keys, it reuses the *last resort* key in the activation step. We observed that the resulting *Master Session Key* differs for every session. Thus, we conclude that it is generated by Apple and sent over satellite.

If Apple's servers decide to invalidate the key material, e.g., the Apple ID configuration or the allowed services changed, communication cannot be established and the baseband indicates *Security Config Update Needed*. Further communication is only possible with a valid configuration.

**Message Transmission:** (7) iOS proceeds with sending application-specific content, such as Find My or text messages. The basic message format has stayed the same since the introduction of satellite communication. However, later iOS versions added new message formats to support novel features.

We publish a documentation of message types and formats in our artifacts' repository [37]. All message contents we observed are encrypted, adding a second encryption layer. Section V-D explains the Find My Friends encryption; Section V-E explains text encryption for emergency messages and roadside assistance. (8) Once a ground station confirms message reception, the baseband sends an indication to iOS to confirm the message transmission. If messages are not acknowledged, iOS schedules retransmissions. This process requires the user to collaborate, as they must continue pointing their iPhone towards the satellite until the transmission succeeds.

**Teardown:** (9) When the user terminates the communication, the baseband deactivates the service to save power. In the case of Find My, this happens automatically after the message transmission finishes. For other applications, the user must actively end the satellite transmission.

## D. Find My Friends

Find My location sharing normally requires an active internet connection. Location sharing over satellite enables similar functionality when no internet connection is available. As the satellite functionality is closely related to normal Find My Friends, we describe both versions in this section.

**Find My Friends over Internet:** Find My Friends uses an internet connection to send and receive location updates by default. While the Bluetooth-based Find My network has been studied in the past [39], [23], [38], we are the first to reverse-engineer the internet-based protocol. To start, users can share their location using the Find My app. They can add a new person in the *People* tab. The targeted recipient of location updates must have an Apple account and an iOS or macOS device to accept the request and view the shared location. Whenever a person wants to share their location with someone, the iPhone sends a request to Apple containing the recipient's Apple ID. The recipient receives a push notification and a prompt whether they want to share their location back to the sender. Each person sharing their location gets assigned a Find My *location id*, which is used to retrieve location updates from Apple's servers.

Find My Friends uses end-to-end encryption, preventing anyone from querying Apple's database and users' locations. End-to-end encryption is based on the Elliptic Curve Integrated Encryption Scheme (ECIES) with a `NIST-P256` elliptic curve key exchange (internally called *keyForSharingLocationToFriends*) and AES-GCM symmetric encryption. When sharing location with a new person, the device creates a direct connection based on Apple's push notification service to transfer the private key to the newly added friend. It is unclear if an Apple-internal attacker could access the keys during the transfer. We consider this scenario out of scope, as iPhone users generally trust various services Apple provides. We compared the keys on both devices to verify that the private key was exchanged. Note that a legacy version of Find My location sharing does not encrypt the location and is still available as a fallback. This legacy variant is incompatible with location sharing via satellite.

When the user opens the Find My app in the *People* tab, the app will request a location update. It sends a request with the friend's *location id* to Apple's servers, which notifies the friend's device via push notifications. The device will fetch a current Global Navigation Satellite System (GNSS) location, encrypt it, and share it back to Apple. Apple's servers forward the location update to the user using a silent push notification. Users can also request live location updates, delivering constant location updates via push notifications.

**Find My Friends over Satellite:** The user must have friends on Find My before they initiate location sharing via satellite. Once started, the iPhone requests a GNSS signal to generate a *litelocation* from it. A *litelocation* is a shortened location format that Apple uses to transfer location information efficiently. The process transforms latitude and longitude double values into a 32 bit fixed-point integer representation by multiplying them with 10 000 000. The bytes of both integers are concatenated and a 1 byte integer, reflecting the horizontal accuracy, is appended. This results in a 9 byte representation of the user's location. Find My uses a similar format when an iPhone finds an AirTag and reports its location to Apple [39]. Then, Find My applies the same end-to-end encryption to the *litelocation*. The ECIES encryption adds 73 byte for the key material and the authentication tag, resulting in an 82-byte long message.

The message neither contains the linked Apple ID nor the associated *location id*, which are required for friends to receive the encrypted location information. Therefore, we conclude that Apple's servers store a link between each user's Apple ID, their Find My *location id*, and the associated LLC keys.

## E. Emergency SOS

This section describes how emergency communication is implemented, detailing the applied text compression and used encryption. We present the message format in Appendix C.

**Initiate Emergency Texting:** The user indicates their emergency by dialing an emergency number on the iPhone. Unlike other phone calls and data transmission, emergency calls are allowed without authentication to cellular network providers [3]. Emergency calls only fail if there is no cellular coverage at all. In this situation, prior to iOS 18, the *Phone* app shows a button for emergency texting over satellite. As this functionality was hidden, Apple updated this part of the user interface in iOS 18 [14], allowing users to see all potentially available satellite services.

**Efficient First Message:** Shorter messages are transmitted faster and more likely under bad conditions. The emergency questionnaire covers the most essential information without any text. After the user has responded to the questionnaire, we found that their responses, current location, and battery level form the *Emergency Start* message sent to the satellite. Each field has a bitwise representation: The full questionnaire only requires 3 byte of data, four battery levels represent the current charge with 2 bit, and the location has another unique format that only requires 8 byte.

**Conversations:** Writing with first responders is based on an interface similar to the Messages app. A message can be up to 160 byte, like SMS. The user interface shows if the message length is exceeded but allows users to type longer texts, which are split into multiple messages and transmitted separately. The internal message representation contains a conversation identifier and a message counter. This way, the receiver knows if messages are pending transmission and assembles messages in the correct order. A conversation ends if the user is inactive for more than 120 min. The message counter uses only 13 bit, allowing for another 2 bit of the message counter to be used for the current battery level, letting first responders know if the user might stop texting due to an empty battery.

**Location Updates:** When the user continues a conversation after a pause or the location changes, their current location information is retransmitted. The location messages have an individual counter of only 8 bit.

**Language Compression Codecs:** Language-specific compression codecs reduce data usage. The emergency text interface limits inputs to UTF-8 characters without emojis or images. Then, the codec compresses the text. Depending on the languages configured in the user's system, multiple compression codecs are applied, and the most efficient one is selected. If the compressed message is not shorter than the uncompressed one, the uncompressed text will be sent. We find that text compression is highly efficient. Using ChatGPT [62], we generate 100 exemplary emergency messages and send them through our simulated environment. The average message length is 34.83 characters, which are compressed to 12.35 characters, resembling a compression ratio of 2.82.

**Message Encryption:** Emergency text messages are shared with first responder coordination centers. In contrast to friends in Find My, their identity is unknown during the online setup phase. Apple solves this challenge using the 256 bit Master Session Key, retrieved during the authentication and registration phase (see Section V-C). We assume that this Master Session Key is known to Apple, enabling them to decrypt and forward emergency messages as needed. Using an HMAC-based Key Derivation Function (HKDF) [48] with SHA-256 as a hash function, iOS derives two separate 256 bit keys, used for transmitting respectively receiving encrypted messages. The initial message, text messages, and location updates are encrypted in AES256 in counter (CTR) mode without padding. The Initialization Vector (IV) is created from the incrementing message counter and session-specific conversation ID. An analysis of the security properties of this encryption scheme follows in Section VI-B.

### *F. Roadside Assistance*

iOS 17 introduced roadside assistance, allowing one to request help in case of a car breakdown [9]. The communication is similar to emergency texting, starting with a questionnaire and continuing with a conversation and location updates. The questionnaire asks for different information, including the user's phone number in free text form. Roadside assistance uses the same encryption scheme as emergency texting and uses almost the same message format.

### *G. SMS and iMessage*

iOS 18 added text messaging features in the U.S. [13]. At the time of writing, these features were restricted to users who installed a developer beta. Since the features were not final and subject to changes, we did not perform an in-depth analysis. Sending SMS is implemented similarly to emergency texting. Apple receives text messages transmitted by the user over satellite and forwards them to the sender's network provider, who forwards them to the receiver. Thus, users can only send SMS over satellite if their network provider collaborates with Apple. For iMessage over satellite, iOS exchanges satellite-specific encryption keys with other iMessage users for each conversation while online. The key exchange only completes if the other user is also on iOS 18.

### *H. Physical Layer Signal*

The iPhone transmits satellite signals over the L-Band (1 610 MHz to 1 626.5 MHz) and receives signals over the S-Band (2 483.5 MHz to 2 500 MHz) [29]. The FCC report specifies the modulation scheme used in the uplink as Single Physical Resource Block (1-PRB) Single Carrier Frequency Division Multiple Access (SC-FDMA). We observe a channel spacing of 200 kHz, with 180 kHz used for the transmission and 20 kHz as guard space to the neighboring channels. These characteristics are very similar to Narrow-band Internet of Things (NB IoT), which uses a variant of LTE [1].

Apple optimized the standard NB IoT transmission scheme for satellite transmissions. As the user points their iPhone directly to a satellite, multi-path effects on the signal are minimal, and thus, a reduced Cyclic Prefix (CP) is used. The signal start contains a training symbol, which enables synchronization. When sending a location message over Find My, the transmission consists of 7 or more bursts, each lasting

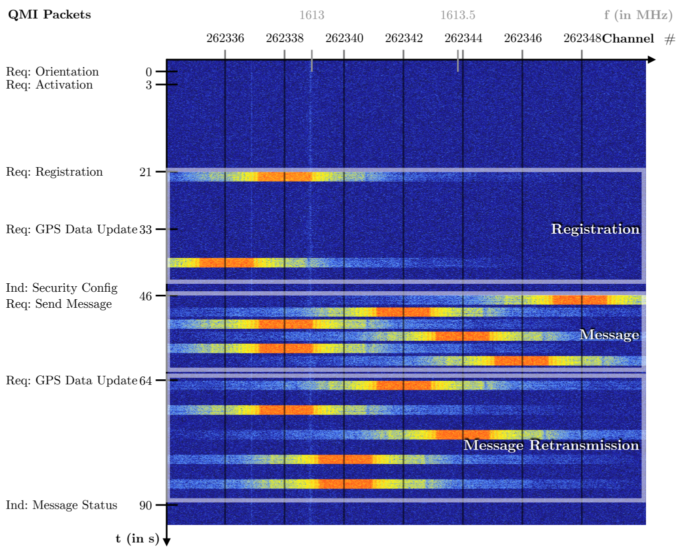


Fig. 7: Spectrogram of a physical-layer uplink signal.

TABLE I: Summary of all attacks we found when analyzing the iPhone satellite communication.

| Attack | Adversary | Privileges | Results |
|---|---|---|---|
| Confidentiality & Integrity (VI-B) | Ⓐ | Receiving satellite signals. | Likely not feasible due to multi-layered encryption. |
| Privacy Limitations (VI-C) | Ⓐ | Receiving satellite signals. | Identifying used satellite communication type. |
| Free SMS-like Messaging Service (VI-D) | Ⓑ | Root-level device access. | Misusing satellite connectivity to send arbitrary messages. |
| Bypassing Service and Location Restrictions (VI-E) | Ⓑ | Root-level device access. | Using restricted services and communicating in restricted regions. |
| Device-specific Restrictions (VI-F) | Ⓑ | Root-level device access. | Bypassing restrictions imposed by local laws. |
| Safety-critical Locations (VI-G) | Ⓒ | System logs with debug profile. | Access to critical locations of ground stations. |

for $1.867\,\mathrm{s}$. Each burst contains a different data type. The supported types are `ack`, `llc_control`, `registration`, and `ucast`. If the satellite connection is interrupted, the bursts are repeated with a fixed backoff timer per type. Each transmission can be on a different channel, and the order of channels changes between transmission attempts.

Figure 7 shows a recording captured with a BladeRF SDR. General information sent over QMI, such as activation information and orientation updates, does not result in wireless transmissions. Once a satellite is within reach, the iPhone proceeds with a registration, which is the first message sent over the air. Due to non-optimal transmission conditions, the registration is retransmitted once. After being confirmed by the satellite, the iPhone applies the security configuration with the shared key and immediately sends the encrypted message. In this particular case, it is a Find My message. Usually, a message transmission only consists of six bursts immediately following each other. Here, the main burst of the communication, containing the actual message, is retransmitted five more times with longer pauses until it is finally confirmed to be received by the satellite. The overall transmission in this example took approximately $90\,\mathrm{s}$. Under optimal conditions, the total transmission time is about $20\,\mathrm{s}$.

## VI. Security, Privacy, and Safety Analysis

We perform an in-depth security and privacy analysis based on the results of our reverse-engineering effort. We define three different attacker models. The first two attackers align with our research questions *RQ1* and *RQ2*, while the third attacker focuses on the safety of critical infrastructures.

Table I summarizes our findings. We find that message-layer encryption does not protect message integrity, however, this is protected on the transport layer (see Section VI-B). Apple has not accommodated adversaries with root-level access to the device: On a jailbroken iPhone, an adversary can bypass any regional restrictions and modify outgoing satellite messages (see Section VI-E). We develop an open-source application, to demonstrate how location sharing can be used to send arbitrary messages of up to 83 bytes (see Section VI-D). In Section VII we give recommendations on designing a secure satellite system that introduces mitigations for most issues presented in this section.

### *A. Attacker Model*

Ⓐ **Passive and Active Network Adversaries:** We consider passive adversaries that record communication, aiming to extract sensitive information, such as medical information or location data. We also consider active adversaries that modify the transmitted data. The active adversary may change parts of the contents, e.g., to modify coordinates such that help for victims will be misdirected to the wrong location. We do not consider adversaries with quantum capabilities—ECDH, as used by Apple, is not post-quantum secure. These adversaries are found in related work as eavesdroppers receiving, decoding and decrypting satellite communication [65], [66], [87], [85], [55]. In most cases, satellite communication was not encrypted.

Ⓑ **Individuals Bypassing Communication Restrictions:** We consider adversaries with jailbroken iPhones who aim to bypass restrictions imposed by Apple. Such jailbroken iPhones are available to the public [54], supporting the iPhone 14 up to iOS 16.5.1 at the time of writing. The attacker can use the jailbreak to modify configuration files and the behavior of binaries executed on the device. The motivation of the adversaries is twofold: The adversaries want to use satellite communication in regions that are not supported or even declare the possession of a satellite phone as a crime [32]. Furthermore, they want to extend the available features to send messages without paying for them and without restrictions by possible censorship.

Ⓒ **State-sponsored Actors and Terrorists:** Satellite communication is a backup channel in case other communication systems collapse. Thus, satellite infrastructure, including ground stations, must be well-protected. We consider an attacker who aims to disrupt satellite communication, such as state-sponsored actors or terrorists. During the Russian war on Ukraine, cellular and satellite services were intentionally blocked, underscoring the relevance of this threat in current times [44], [67]. We assume these attackers have access to the same information as individuals and possess weapons to attack satellite communication infrastructure.

### *B. Confidentiality & Integrity*

Previous Globalstar Spot devices were shown to expose sensitive user location data due to the absence of any encryption during transport [55]. We analyze the applied encryption methods for satellite communication in this section.

**Emergency SOS & Roadside Assistance:** In this section, we present multiple issues with the byte-efficient encryption used for emergency texting and roadside assistance. The encryption algorithm used is AES256 in CTR mode with a deterministic IV based on the conversation ID and the message counter. While this approach minimizes communication size, which is essential in an emergency case over a slow satellite connection, it could have weaknesses exploitable by passive or active machine-in-the-middle attacks. Because AES in CTR mode is a stream cipher, reuse of a nonce for the same key leads to a simple known-plaintext attack. Furthermore, as AES-CTR is not an authenticated encryption scheme, the ciphertext is malleable, i.e., it can be modified to change the plaintext without the receiver realizing it. Every bit flip in the ciphertext corresponds to a bit flip in the plaintext at the same position. An adversary might use this to change answers in the questionnaire, which is transmitted as a bit array, or to change the location, potentially leading emergency responders to the wrong location.

**Find My Location Sharing:** As detailed in Section V-D, all location updates are end-to-end encrypted using the well-established ECIES standard with the `NIST-P256` curve and AES in Galois/Counter Mode (GCM). Adversaries who are able to record or intercept an outgoing location update cannot decrypt the actual location, nor can they encrypt a fake location update.

**iMessage and SMS:** All messages sent using iMessage over satellite are end-to-end encrypted with a key that is used exclusively for encrypting satellite messages between the users. We could not test SMS via satellite, as our carriers do not support it.

**Multi-layer Encryption:** Emergency texting uses AES in CTR mode to encrypt messages sent via satellite to first responders. As the encryption mode does not provide authentication and no further integrity protection is used, we reported this to Apple's product security team. They reviewed the issue and described a multi-layer encryption approach that they use for all satellite communication (see Appendix D). The outer layer is encrypted and authenticated during transport, i.e., while the message travels through air and space to a ground station. When the message is received, the *Apple Satellite Service* decrypts the payload and retrieves metadata to obtain routing information. The inner payload, which is still encrypted depending on the message sent, is then routed to the correct service. E.g., for an emergency message, Apple's emergency service receives the payload, decrypts the inner payload, and displays it to first responders.

Apple did not specify the exact algorithms in use, but they usually adhere to well-tested encryption standards. We did not see any signs of this encryption in iOS, and it is likely that the outer layer encryption is implemented in the baseband. Since the baseband has access to the shared secret derived from the LLC key, it could encrypt and authenticate all messages.

### C. *Privacy Limitations*

**Offline Location Sharing:** The list of friends can only be modified when the user is online. If the user shares a location with their friends while offline, they cannot change who is seeing this location update. All friends of the user share the same key for decrypting the location update, making it impossible to exclude one of them without an internet connection. As a result, one might involuntarily share private location data.

**Unencrypted Message Types:** Although we found that the transmitted private data is protected against external access, an eavesdropper can identify the type of communication a user performs. By default, the message type is not encrypted when the message is sent to the baseband. Even if the message is fully encrypted, the transmission scheme allows the eavesdropper to identify which kind of satellite communication is happening, e.g., location sharing always creates a pattern of seven bursts. Furthermore, text lengths become apparent, as there is no padding to enable faster transmissions.

The second limitation can be mitigated using end-to-end transport encryption based on the LLC keys and padding to ensure all messages are the same length. However, previous research has shown that even encrypted traffic can be analyzed using machine learning [73].

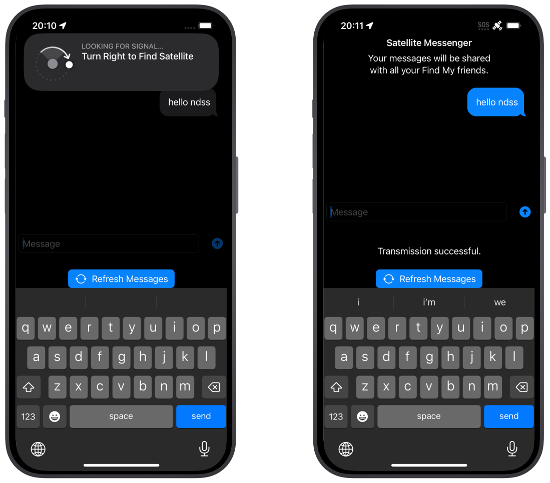
### D. *Free SMS-like Messaging Service*

Find My Friends is based on end-to-end encryption between the sender and the group of friends with whom the sender shares their location. Attacker Ⓑ wants to utilize this feature and modify the location update, allowing them to use this technology as a free messaging service.

**Sending Messages:** To send custom messages over satellite, the attacker modifies the encrypted location update before it is forwarded to the baseband and sent to the satellite. The attacker can manipulate the processes involved at runtime and inject any custom message before the iPhone sends it to the satellite. The only requirement for this is a computer with Frida or similar tooling and a jailbroken iPhone.

Any message forwarded to the baseband is sent to the satellite as a Find My Friends location update. Although the legitimate location update uses only 83 byte, attackers can send messages of up to 160 byte. If the attacker tries to send messages with more than 160 byte, the transmission is not acknowledged. The same maximum message size is used for emergency text messages and roadside assistance. Apple has likely imposed this size as an upper bound on the MAC layer.

**Receiving Messages:** The receiver must be a friend of the attacker in the Find My app, allowing them to receive location updates from the sender. The receiver can fetch the message in two ways: ① By opening the Find My app and requesting the attacker's location. However, the location update cannot be decoded because the attacker modified the contents. Therefore, the app cannot display the message but will log the HTTP response containing the entire message. Observing logs does not require root access for the receiver. ② The receiver can send the HTTP request for the location update directly to



20:10

LOOKING FOR SIGNAL...

Turn Right to Find Satellite

hello ndss

20:11

Satellite Messenger

Your messages will be shared with all your Find My friends.

hello ndss

Transmission successful.

Fig. 8: Messenger app using satellite location sharing to send arbitrary messages (Demo: https://youtu.be/3qYby6CeBxs).

Apple's servers, authenticated with anisette data [39] and an access token. Access to these parameters requires privileged access to the system, like a jailbreak on iOS or disabling system integrity protection on a Mac [39]. The server response contains the Base64-encoded message.

Listing 1 shows an HTTP response containing a modified message sent using a satellite location update. The *fmt* set to *1* indicates that the location update was shared via satellite. The *location* field contains our Base64-encoded message consisting of the repeating letter 'A'.

**Message Throttling Bypass:** Sharing a location over Find My is limited to one location every 15 min. Apple's servers or ground stations do not enforce this limit. Instead, iOS throttles message transmission locally using a `CFPreference` called `lastLiteLocationPublish`, which stores the last date of a successful location share. By overwriting this setting, we can send multiple locations within the 15 min time window.

**Proof of Concept App:** We create an iOS app that works for the iPhone 14 on iOS 16.1 or higher using the *Dopamine 2* jailbreak [30]. We use the jailbreak to access private frameworks and to add a tweak that overwrites Find My messages before they are sent. We do not integrate further bypasses for legal reasons. Figure 8 shows the proof of concept, which we also made available open source [37].

```
{ "locationPayload": [
    { "locationInfo": [ {                AAA... in Base64
    "locationTs": 1707137144362,
    "location": "QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUF",
    "fmt": 1 } ],
    "id": "fq0UsX8ZEUUrVJHvHRMc/9qxFm5vsCIPZWSW5kdCK6I="}
],
"configVersion": 1,
"statusCode": "200" }
```

Listing 1: JSON response with custom message inside the location information.

**Security Implications:** Misuse can affect service stability since satellite bandwidth is limited, reducing user experience for others. Additionally, Apple might monetize text messaging services through satellite connections with iOS 18. Similar services already exist by Huawei and Motorola [22], [57]. Without added protection mechanisms, users can send text messages for free without restrictions on the number of messages. While Apple cannot fix this issue without significant updates to the Find My protocol, they started enforcing the exact message size of 83 byte after our responsible disclosure. As they can map LLC keys to Apple IDs, they could take further actions in the future, such as blocking users abusing the service.

## *E. Radio Exclusion Zones & Regional Restrictions Bypass*

Satellite transmission devices must comply with radio exclusion zones. iPhones apply these restrictions in software, which complies with the FCC report [29]. The Trial-based configuration allows Apple to update all of these settings at any time, e.g., if geopolitical events require adding new exclusion zones. Even users who do not update their iOS version will receive these changes. Staying offline to avoid new rules does not work, as the target configuration will expire, meaning that the iPhone can no longer calculate satellite orbits.

However, on jailbroken devices, attackers can modify the radio exclusion zones and other restrictions by modifying the Trial configuration. The configuration is stored in a human-readable XML format in the file `/private/var/mobile/Library/Trial/Treatments/BIFROST_PROD_2/.../Config.plist`.

According to the Globalstar coverage map [33], their satellite services are available and legal to use across Europe. As of July 2024, Apple restricts satellite services to selected European countries. We test the effectiveness of modifying the Trial configuration in the following experiments.

**Adding Support for a Country:** We add Poland to the Trial configuration and attempt satellite communication close to the Polish–German border. During the *Activation* phase, the iPhone indicates its location as Poland, as expected. Despite Poland not being officially supported, the *Registration* is confirmed, and we can send Find My locations over satellite.

**Roadside Assistance in Non-U.S. Countries:** Another restriction is that Roadside Assistance is not available in the EU. We manipulate the configuration files to add the feature to an EU country. We successfully initiate communication but abort it before the message is sent to prevent false alarms.

**iMessage and SMS texting outside of the U.S.:** We manipulate the configuration files to add iMessage and text messaging via satellite to an EU country. As SMS requires collaboration with the provider, we were only able to test iMessage. We were able to communicate as intended with another person residing in the U.S.

**Security Implications:** Our experiments demonstrate that Apple's satellite services can be used regardless of an individual's location. Feature and location availability solely depends on the Target configuration, which is modifiable.

## F. Device-specific Restrictions

Find My Friends is unavailable in South Korea due to legal restrictions [12]. Devices sold in Korea cannot use Find My, even outside Korea. Device model numbers ending with *KH* indicate Korean devices. Existing iOS tweaks for jailbroken devices support modifying the model by changing it in the property list of `Mobile Gestalt` [63]. The property list is obfuscated to prevent jailbreaks from easily figuring out configurations. We change the model number from Korea to Poland by changing these entries:

```
<key>zHeENZu+wbg7PUprwNwBWg</key>
<string>PM/A</string>
<key>h63QSdBCiT/z0WU6rdQv6Q</key>
<string>PM</string>
```

However, this approach did not fully allow Find My Friends on the iPhone 14 SRD linked to Korea. It is further required to add the following entry to allow the Find My Friends feature:

```
<key>FMFAllowed</key>
<real>1</real>
```

These preferences become effective when opening the *Settings* app and selecting *Reset All Settings*. This step ensures that the iPhone reloads all settings from the modified file, preventing them from being cached elsewhere. The user must relogin with their iCloud account, thereby re-fetching account-specific settings. However, this approach only works with iCloud accounts not created on Korean devices.

**Security Implications:** Our experiments show that users are not bound to the legal restrictions of the country where they purchased a device.

## G. Safety-critical Locations

Globalstar and Apple have no openly available documentation about their ground station infrastructure. We assume this is to protect communication sites and hide business secrets.

Successful satellite transmission also requires the targeted satellite to reach a ground station. The iPhone holds a list of ground station locations to calculate an optimal communication path. iOS does not contain a ground station list as part of the system image but downloads it on demand over the Trial service to keep it up-to-date. Trial assets are encrypted and packed in a proprietary format, making them difficult to download and analyze directly.

We found two possibilities for extracting the list of ground stations from an iPhone. ① On jailbroken devices, the Trial configuration file is accessible through SSH. ② On non-jailbroken devices, system logs contain the list of ground stations. The log entries are marked as `<private>` by default. However, after installing a baseband debug profile [7], the log verbosity increases and includes the anchor list. We confirm the ground stations are valid locations by looking them up in Google Map's satellite view. All of them reveal massive antenna installations.

**Safety Implications:** Ground stations for satellite communication are difficult to set up and cannot be moved. Once leaked, knowledge about their precise locations can have severe implications for the physical security of these sites. We reported this issue to Apple, but since this is an essential limitation of their satellite communication algorithm, they do not rate it as a potential security issue.

## H. Responsible Disclosure

We reported all issues to Apple's product security team. We made our first report in March 2023, and also shared the entire paper with Apple before it was peer-reviewed. While we list several security issues in this section, Apple did not share this view with us. None of the reported problems were classified as security issues. Nevertheless, Apple made some changes after our reports: They limited the maximum length of a satellite location report to 82 byte at the ground station. Therefore, messages longer than that may no longer be received when using our satellite messenger (see Section VI-D). Regarding the issue in Section VI-G, Apple classified the ground station leak as expected behavior and stated that the information is publicly available. We confirm this by finding the location of the ground station in Estonia online [53]. While they agreed that their message encryption scheme lacks integrity checks, they provided a detailed answer explaining how they further secure satellite communication on the transport layer. They supported us in including their answer in this paper (see Appendix D).

## VII. Recommendations for Designing Secure Satellite Communication

Previously studied satellite communication systems lacked encryption, leaking personal data, and allowing adversaries to manipulate messages [55]. Partly, these issues arise from the bandwidth constraints and the need to guarantee reliable communication. Apple invested in designing secure and private satellite communication. While this system has many strong points and solves various challenges through an internet-based setup phase, we found several vulnerabilities.

With satellite services also coming to Android devices [80], [43], [71], [79], including rumors for satellite support in mainline Android [50], designing secure satellite communication becomes relevant to a broad target audience. This section shows how to design a secure satellite communication system based on Apple's work. We include mitigations for all discovered vulnerabilities. We generalize our recommendations so that they apply to future satellite communication systems.

## A. Enforcing Usage Restrictions

Satellite services must follow legal restrictions. As described in the FCC document for the iPhone 14, Apple conforms to this by applying such restrictions in software [29]. Software-based restrictions come at the risk of being modified on jailbroken iPhones and rooted Android devices (see Section VI-D and VI-E). While jailbreaks are less common on iOS, current lawsuits and legislation [78], [82] might force Apple to open iOS further.

On the contrary, access to wireless firmware is increasingly restricted to comply with radio emission guidelines [77].

Baseband chips enforce this by only running firmware signed by the vendor. Thus, we consider the baseband chip firmware as an independent root of trust.

**Recommendation:** We propose integrating the baseband chip into the chain of trust for satellite configurations. Configuration settings such as radio exclusion zones, country-based restrictions, and allowed services could be signed by the vendor, with verification performed by the baseband firmware.

However, mobile devices should not be considered trustworthy. Attackers could reconstruct the entire communication stack using SDRs to bypass restrictions. Apple addresses this issue by enforcing legitimate usage of its services and requiring valid LLC keys to participate in satellite communication. These keys are generated within the SEP, enhancing their protection against extraction. Given their ability to identify users based on LLC keys, Apple could block users if used for satellite transmissions within restricted areas or for sending Find My locations more frequently than every $15\,\mathrm{min}$.

**Recommendation:** We suggest checking for malicious client-side behavior on the server side and blocking users who do not conform to the protocol constraints.

### *B. Robust Cryptography*

Cryptography ensures secure communication in wireless systems, including properties like confidentiality, integrity, and authentication. Usually, these properties are ensured by complex protocols like Transport Layer Security (TLS) on the internet or 5G authentication for cellular networks [2], [45]. In satellite communication, any packet transfer or byte added to a packet leads to a longer transmission time in an order of magnitude that is directly noticeable by the end-user. When bringing secure communication to satellites, existing protocols must be modified to meet this constraint.

Apple solves key establishment and authentication through the online pre-registration of LLC keys. This way, they can use the secure ECDH key exchange protocol [21] and directly proceed with authentication and transport layer encryption. Apple then adds a second layer of encryption depending on the service used. For Find My Friends locations are end-to-end encrypted using public key encryption with pre-shared keys and secured against manipulation with an authentication tag. This increases the message size of a 9 byte location by 79 byte, which is costly but acceptable for Apple during non-emergency usage. To encrypt emergency text messages during SOS and roadside assistance, Apple leaves out authentication tags as the transport layer already integrates them. This allows to save bytes when sending time-critical information.

**Recommendation:** For future implementations of satellite communication we recommend a similar approach using strong encryption and authentication modes, which protect confidentiality, authenticity, and integrity. We recommend 32 bit authentication tags as a security and packet size tradeoff. Furthermore, to protect against potential nonce reuse attacks, we recommend using random nonces instead of message counters.

## VIII. Conclusion

Apple's satellite communication system brings a modern protocol to an infrastructure that existed since the first iPhone was released in 2007. We reverse-engineered the details of this protocol and built a safe simulation-based test environment, followed by an in-depth security and privacy analysis. Apple designed a protocol that works offline, based on pre-fetched information. Despite this limitation, each session is secured by a fresh session key pair.

The security and safety issues we discovered are based on design decisions, which become apparent when looking at the protocol in a broader context. Find My location sharing can be abused for SMS-like messaging. While this is a flaw by design, with the ability to modify message content outside of the baseband chip, the maximum message length was not checked on the server end. In addition, we found that all regional restrictions—radio exclusion zones, model-based restrictions, and unsupported countries—are purely configured in software, allowing us to bypass them on jailbroken devices. This implies that Apple's satellite communication system is open for adversarial use with unintended applications. Moreover, we found that the transmission path calculation information leaks safety-critical ground station locations.

Signal modulation and transport layer encryption are handled in the Qualcomm baseband chip, which is an open target for future research. The modularity and extensibility of the overall system will allow for additional use cases in the future. We are excited to see how this will be extended.

## Acknowledgment

This work has been co-funded by Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 – 236615297, by the German Federal Ministry of Education and Research and the Hessian State Ministry for Higher Education, Research, and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE.

## References

[1] 3GPP, "3GPP TS 36.300 V18.0.0," 2023. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/36_series/36.300/36300-i00.zip

[2] 3GPP, "Security architecture and procedures for 5G system," 3GPP, Technical Specification (TS) 33.501, 2023, version 18.4.0.

[3] ——, "3GPP TS 33.102 V17.0.0(2022-03)," 2024. [Online]. Available: https://3gpp.guru/trts/Rel-17/33102-h00.html

[4] Apple, "Apple Platform Security," 2022. [Online]. Available: https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf

[5] ——, "Emergency SOS via satellite available today on the iPhone 14 lineup in the US and Canada," 2022. [Online]. Available: https://www.apple.com/newsroom/2022/11/emergency-sos-via-satellite-available-today-on-iphone-14-lineup/

[6] ——, "Apple-oss-distributions/xnu," 2023. [Online]. Available: https://github.com/apple-oss-distributions/xnu

[7] ——, "Bug Reporting Profiles and Logs," 2023. [Online]. Available: https://developer.apple.com/bug-reporting/profiles-and-logs/

[8] ——, "Notruf SOS über Satellit ab heute für iPhone 14 Modelle in Österreich, Belgien, Italien, Luxemburg, den Niederlanden und Portugal verfügbar," 2023. [Online]. Available: https://www.apple.com/at/newsroom/2023/03/emergency-sos-now-in-austria-belgium-italy-luxembourg-netherlands-portugal/

[9] ——, "Request Roadside Assistance via satellite on your iPhone," 2023. [Online]. Available: https://support.apple.com/guide/iphone/request-roadside-assistance-via-satellite-iph29bea54b5/17.0/ios/17.0

[10] ——, “Send your location via satellite in Find My on iPhone,” 2023. [Online]. Available: https://support.apple.com/guide/iphone/send-your-location-via-satellite-iph2aac8ae20/ios

[11] ——, “Apple Security Research Device Program,” 2024. [Online]. Available: https://security.apple.com/research-device/

[12] ——, “Find friends and share your location with Find My - Apple Support (MT),” 2024. [Online]. Available: https://support.apple.com/en-mt/105122

[13] ——, “iOS 18 makes iPhone more personal, capable, and intelligent than ever - Apple,” 2024. [Online]. Available: https://www.apple.com/newsroom/2024/06/ios-18-makes-iphone-more-personal-capable-and-intelligent-than-ever/

[14] ——, “iOS 18 Preview,” 2024. [Online]. Available: https://www.apple.com/ios/ios-18-preview/

[15] ——, “Set up and view your Medical ID,” 2024. [Online]. Available: https://support.apple.com/en-ca/guide/iphone/iph08022b192/ios

[16] ——, “Use Crash Detection on iPhone or Apple Watch to call for help in an accident,” 2024. [Online]. Available: https://support.apple.com/en-us/104959

[17] ——, “Use Emergency SOS via satellite on your iPhone,” 2024. [Online]. Available: https://support.apple.com/en-us/HT213426

[18] ——, “Use Fall Detection with Apple Watch,” 2024. [Online]. Available: https://support.apple.com/en-us/108896

[19] ——, “Using Sysdiagnose to Troubleshoot iOS or iPadOS,” 2024. [Online]. Available: https://it-training.apple.com/tutorials/support/sup075

[20] L. Arnold, M. Hollick, and J. Classen, “Catch You Cause I Can: Busting Rogue Base Stations using CellGuard and the Apple Cell Location Database,” in *RAID*, 2024.

[21] E. Barker, L. Chen, A. Roginsky, A. Vassilev, and R. Davis, “Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography (rev. 3),” National Institute of Standards and Technology, Tech. Rep., 2018. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-56Ar3

[22] Castro, “HUAWEIHuawei Mate 60 Pro satellite calling costs 100 yuan ($13) for 80 minutes of talktime,” 2023. [Online]. Available: https://consumer.huawei.com/za/community/details/HUAWEIHuawei-Mate-60-Pro-satellite-calling-costs-100-yuan-13-for-80-minutes-of-talktime/topicId_280458/

[23] J. Classen, A. Heinrich, R. Reith, and M. Hollick, “Evil Never Sleeps: When Wireless Malware Stays On after Turning Off iPhones,” in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3507657.3528547

[24] J. Classen and M. Hollick, “Happy MitM: Fun and Toys in Every Bluetooth Device,” in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2021.

[25] CNET, “SpaceX Puts Its First Satellites in Orbit to Connect T-Mobile’s Customers,” 2024. [Online]. Available: https://www.cnet.com/tech/mobile/spacex-launches-first-satellites-to-connect-t-mobiles-customers/

[26] B. Driessen, R. Hund, C. Willems, C. Paar, and T. Holz, “An experimental security analysis of two satphone standards,” *ACM Trans. Inf. Syst. Secur.*, 2013. [Online]. Available: https://dl.acm.org/doi/10.1145/2535522

[27] Eumetsat, “Two Line Elements,” 2023. [Online]. Available: https://service.eumetsat.int/tle/

[28] Federal Communications Commission, “Application for Mobile Satellite Service by GUSA Licensee LLC,” 2009. [Online]. Available: https://fcc.report/IBFS/SES-MFS-20091221-01611

[29] ——, “14040863-S1V5 FCC SAR Report, FCC ID: BCG-E8151A, RF Exposure Info,” 2022. [Online]. Available: https://fcc.report/FCC-ID/bcge8151a/6092766

[30] L. Fröder, “Dopamine Jailbreak,” 2024. [Online]. Available: hhttps://github.com/opa334/Dopamine

[31] G. Giuliari, T. Ciussani, A. Perrig, and A. Singla, “ICARUS: Attacking low Earth orbit satellite networks,” in *2021 USENIX Annual Technical Conference*. USENIX Association, 2021. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/giuliari

[32] Global Rescue, “Where Is Your Satellite Phone Illegal? – Global Rescue,” 2023. [Online]. Available: https://www.globalrescue.com/common/blog/detail/where-satellite-phone-illegal/

[33] Globalstar, “Coverage Maps,” 2024. [Online]. Available: https://www.globalstar.com/en-ap/coverage-maps

[34] ——, “Globalstar Satellite Technology,” 2024. [Online]. Available: https://www.globalstar.com/en-ap/about/our-technology

[35] Google Open Source, “Qualcomm MSM Interface (QMI),” 2019. [Online]. Available: https://android.googlesource.com/kernel/msm/+/android-7.1.0_r0.2/Documentation/arm/msm/msm_qmi.txt

[36] J. A. Guerrero-Saade and M. van Amerongen, “AcidRain | A Modem Wiper Rains Down on Europe,” 2022. [Online]. Available: https://www.sentinelone.com/labs/acidrain-a-modem-wiper-rains-down-on-europe/

[37] A. Heinrich and J. Classen, “Artifacts for “Starshields for iOS: Navigating the Security Cosmos in Satellite Communication”,” Zenodo, 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13863530

[38] A. Heinrich, M. Stute, and M. Hollick, “OpenHaystack: A Framework for Tracking Personal Bluetooth Devices via Apple’s Massive Find My Network,” in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3448300.3468251

[39] A. Heinrich, M. Stute, T. Kornhuber, and M. Hollick, “Who Can Find My Devices? Security and Privacy of Apple’s Crowd-Sourced Bluetooth Location Tracking System,” *Proceedings on Privacy Enhancing Technologies*, 2021. [Online]. Available: https://www.petsymposium.org/2021/files/papers/issue3/popets-2021-0045.pdf

[40] D. Heinze, J. Classen, and M. Hollick, “ToothPicker: Apple Picking in the iOS Bluetooth Stack,” in *14th USENIX Workshop on Offensive Technologies*. USENIX Association, 2020. [Online]. Available: https://www.usenix.org/conference/woot20/presentation/heinze

[41] D. Heinze, J. Classen, and F. Rohrbach, “MagicPairing: Apple’s Take on Securing Bluetooth Peripherals,” in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3395351.3399343

[42] F. R. Hoots and R. L. Roehrich, *Models for propagation of NORAD element sets*. Office of Astrodynamics, 1980. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA093554

[43] Huawei, “Huawei Mate 60 Pro is the world’s first satellite calling phone,” 2023. [Online]. Available: https://consumer.huawei.com/za/community/details/Huawei-Mate-60-Pro-is-the-world-s-first-satellite-calling-phone/topicId_280286/

[44] M. Hunder, J. Landay, and S. Bern, “Ukraine’s top mobile operator hit by biggest cyberattack of war,” *Reuters*, 2023. [Online]. Available: https://www.reuters.com/technology/cybersecurity/ukraines-biggest-mobile-operator-suffers-massive-hacker-attack-statement-2023-12-12/

[45] IETF, “The Transport Layer Security (TLS) Protocol Version 1.3,” IETF, Tech. Rep., 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8446

[46] E. Jedermann, M. Strohmeier, V. Lenders, and J. Schmitt, “RECORD: A RECeption-Only Region Determination Attack on LEO Satellite Users,” in *33th USENIX Security Symposium*. USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/jedermann

[47] D. T. Kelso, “Celestrak,” 1985, accessed: 2024-01-30. [Online]. Available: hhttps://celestrak.org/

[48] H. Krawczyk and P. Eronen, “HMAC-based Extract-and-Expand Key Derivation Function (HKDF),” 2010. [Online]. Available: https://tools.ietf.org/html/5869

[49] T. Kröll, S. Kleber, F. Kargl, M. Hollick, and J. Classen, “ARIstoteles – Dissecting Apple’s Baseband Interface,” in *Computer Security – ESORICS 2021*, E. Bertino, H. Shulman, and M. Waidner, Eds. Cham: Springer International Publishing, 2021.

[50] A. Li, “Android 14 will support satellite connectivity and partners,” 2022. [Online]. Available: https://9to5google.com/2022/09/01/android-14-satellite-support/

[51] Macworld, “Macworld Expo Keynote Live Update: Introducing the iPhone,” 2007. [Online]. Available: https://www.macworld.com/article/183052/liveupdate-15.html

[52] ——, “How an iPhone 14 ‘literally’ saved a family trapped in the Hawaii wildfires. Family rescued after using Emergency SOS via satellite.” 2023. [Online]. Available: https://www.macworld.com/article/2027967/iphone-14-emergency-sos-via-satellite-hawaii-maui-wildfires.html

[53] Merko Group, “Kilingi-Nõmme antenna station,” 2021. [Online]. Available: https://www.group.merko.ee/en/project/kilingi-nomme-antenna-station/

[54] mineek, “Serotonin Jailbreak,” 2024. [Online]. Available: https://github.com/mineek/Serotonin

[55] C. Moore, “Spread Spectrum Satcom Hacking Attacking The Globalstar Simplex Data Service,” 2015. [Online]. Available: https://www.youtube.com/watch?v=1VbmHmzofmc
[56] A. Morgado and D. Williams, “libqmi,” 2021. [Online]. Available: https://www.freedesktop.org/wiki/Software/libqmi/
[57] Motorola, “Motorola Defy Satellite Link,” 2024. [Online]. Available: https://motorolarugged.com/en-us/motorola-defy-satellite-link/
[58] NASA Space Science Data Coordinated Archive (NSSDCA), “NSSDCA Master Catalog,” https://nssdc.gsfc.nasa.gov/nmc/, 2024, accessed: 29.02.2024.
[59] S. Nellis, “New iPhones have Qualcomm satellite modem, new Apple radio chips,” *Reuters*, 2022. [Online]. Available: https://www.reuters.com/technology/new-iphones-have-qualcomm-satellite-modem-new-apple-radio-chips-2022-09-17/
[60] NowSecure, “Frida,” 2024. [Online]. Available: https://frida.re
[61] Nuand, “BladeRF,” 2024. [Online]. Available: https://www.nuand.com/bladerf-1/
[62] OpenAI, “ChatGPT: Large-scale generative model,” 2023, generated by ChatGPT, an AI language model developed by OpenAI. [Online]. Available: https://openai.com/chatgpt
[63] PARKasd, “locchange iOS Tweak,” 2024. [Online]. Available: https://github.com/PARKasd/locchange
[64] J. Pavur and I. Martinovic, “Building a launchpad for satellite cyber-security research: Lessons from 60 years of spaceflight,” *Journal of Cybersecurity*, 2022. [Online]. Available: https://doi.org/10.1093/cybsec/tyac008
[65] J. Pavur, D. Moser, V. Lenders, and I. Martinovic, “Secrets in the sky: On privacy and infrastructure security in DVB-S satellite broadband,” in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3317549.3323418
[66] J. Pavur, D. Moser, M. Strohmeier, V. Lenders, and I. Martinovic, “A Tale of Sea and Sky On the Security of Maritime VSAT Communications,” in *2020 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, 2020.
[67] J. Pearson and J. Pearson, “Russia downed satellite internet in Ukraine -Western officials,” *Reuters*, 2022. [Online]. Available: https://www.reuters.com/world/europe/russia-behind-cyberattack-against-satellite-internet-modems-ukraine-eu-2022-05-10/
[68] J. Rainbow, “Globalstar soars on Apple’s $1.7 billion satellite investment,” 2024. [Online]. Available: https://spacenews.com/globalstar-soars-on-apples-1-5-billion-satellite-investment/
[69] B. Rhodes, “Skyfield: High precision research-grade positions for planets and Earth satellites generator,” Astrophysics Source Code Library, record ascl:1907.024, 2019.
[70] N. Rollshausen, A. Heinrich, M. Hollick, and J. Classen, “WatchWitch: Interoperability, Privacy, and Autonomy for the Apple Watch,” *Proceedings on Privacy Enhancing Technologies*, 2024.
[71] E. Roth, “AT&T helped connect the first satellite 5G phone call,” 2023. [Online]. Available: https://www.theverge.com/2023/9/19/23879527/att-cellular-satellite-ast-spacemobile-5g
[72] T. Roth, F. Freyer, M. Hollick, and J. Classen, “AirTag of the Clones: Shenanigans with Liberated Item Finders,” in *IEEE Security and Privacy Workshops*, 2022.
[73] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, and K. Xu, “Machine Learning-Powered Encrypted Network Traffic Analysis: A Comprehensive Survey,” *IEEE Communications Surveys & Tutorials*, 2023.
[74] Skylo Team, “Skylo Connectivity Enables New Satellite SOS Feature on Google Pixel 9 Series - Newsroom - Skylo,” 2024. [Online]. Available: https://www.skylo.tech//newsroom/skylo-connectivity-enables-new-satellite-sos-feature-on-google-pixel-9-series
[75] M. Stute, A. Heinrich, J. Lorenz, and M. Hollick, “Disrupting Continuity of Apple’s Wireless Ecosystem Security: New Tracking, DoS, and MitM Attacks on iOS and macOS Through Bluetooth Low Energy, AWDL, and Wi-Fi,” in *30th USENIX Security Symposium*. USENIX Association, 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/stute
[76] E. TC-SES, “Satellite Earth Station (SES); Possible European standardisation of certain aspects of satellite Personal Communications Networks (S-PCN),” DTR/SES-05007, September, Tech. Rep., 1993.
[77] The European Parliament and the Council of the European Union, “Directive 2014/35/EU of the European Parliament and of the Council of 26 February 2014 on the harmonisation of the laws of the Member States relating to the making available on the market of electrical equipment designed for use within certain voltage limits (recast) Text with EEA relevance,” 2014. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014L0034
[78] ——, “Regulation (EU) 2022/1925 of the European Parliament and of the Council of 14 September 2022 on contestable and fair markets in the digital sector and amending Directives (EU) 2019/1937 and (EU) 2020/1828 (Digital Markets Act) (Text with EEA relevance),” 2022. [Online]. Available: http://data.europa.eu/eli/reg/2022/1925/oj/eng
[79] The Verge, “T-Mobile and SpaceX Starlink say your 5G phone will connect to satellites next year,” 2022. [Online]. Available: https://www.theverge.com/2022/8/25/23320722/spacex-starlink-t-mobile-satellite-internet-mobile-messaging
[80] ——, “The first phone maker to add satellite texting to its devices is... Huawei,” 2022. [Online]. Available: https://www.theverge.com/2022/9/6/23339717/huawei-mate-50-pro-satellite-text-china-beidou
[81] U. Today, “‘Lifesaver’: How iPhone’s satellite mode helped during Hurricane Helene,” 2024. [Online]. Available: https://eu.usatoday.com/story/tech/2024/10/11/iphone-satellite-mode-att-cell-service/75616990007/
[82] U.S. Department of Justice, Antitrust Division, “Office of Public Affairs | Justice Department Sues Apple for Monopolizing Smartphone Markets | United States Department of Justice,” 2024. [Online]. Available: https://www.justice.gov/opa/pr/justice-department-sues-apple-monopolizing-smartphone-markets
[83] D. Vallado, P. Crawford, R. Hujsak, and T. Kelso, “Revisiting spacetrack report# 3,” in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006.
[84] J. Willbold, M. Schloegel, M. Vögele, M. Gerhardt, T. Holz, and A. Abbasi, “Space Odyssey: An Experimental Software Security Analysis of Satellites,” in *IEEE Symposium on Security and Privacy*. Los Alamitos, CA, USA: IEEE Computer Society, 2023. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00131
[85] J. Willbold, M. Schloegel, R. Bisping, M. Strohmeier, T. Holz, and V. Lenders, “VSAsTer: Uncovering Inherent Security Issues in Current VSAT System Practices,” in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3643833.3656139
[86] L. Wouters, “Glitched on Earth by Humans: A Black-Box Security Evaluation of the SpaceX Starlink User Terminal,” 2022. [Online]. Available: https://i.blackhat.com/USA-22/Wednesday/US-22-Wouters-Glitched-On-Earth.pdf
[87] L. Yu, J. Hao, J. Ma, Y. Sun, Y. Zhao, and B. Luo, “A Comprehensive Analysis of Security Vulnerabilities and Attacks in Satellite Modems,” in *ACM CCS*, Salt Lake City, UT, USA, 2024-10-14/2024-10-18.
[88] P. Yue, J. An, J. Zhang, J. Ye, G. Pan, S. Wang, P. Xiao, and L. Hanzo, “Low Earth Orbit Satellite Security and Reliability: Issues, Solutions, and the Road Ahead,” *IEEE Communications Surveys & Tutorials*, 2023.

## Appendix

### A. List of Satellites

The Targets list contains information about all satellite trajectories, described by TLEs. Every TLE comes with a satellite identifier (69 in this case), followed by a data array:

```
<key>69</key>
<array>
  <string>1 00000U 00000A   24 26.70901532  .00000000
       00000-0  58480-4 0  0003</string>
  <string>2 00000  51.9948 286.8188 0001510  54.0997
       279.4771 12.62265350000009</string>
</array>
```

Listing 2: Example TLE data for satellite *M069*.

We confirm that the identifier in the Targets list matches the satellite number. To this end, we use an open-source SGP4 implementation [69] to calculate the satellites’ positions from Apple’s TLE information. Figure 9 shows the resulting

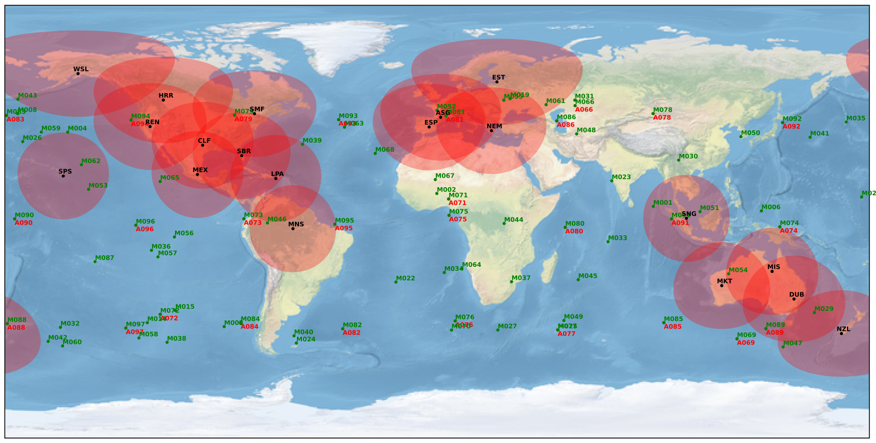
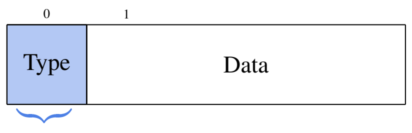


Fig. 9: Map of satellites from iPhone targets list (in red, prefixed with *A*) and publicly available Globalstar satellite data (in green, prefixed with *M*) [47]. Ground stations are shown in black with 2 000 km coverage radius in light red.



| ID | Type |
|---|---|
| 00 | Fake Emergency Start |
| 01 | Emergency Start |
| 02 | Emergency Message |
| 03 | Emergency Location Update |
| 04 | FindMy Message |
| 05 | Fake Roadside Start |
| 06 | Roadside Start |
| 07 | Roadside Message |
| 08 | Roadside Location Update |
| 09 | Carrier Pigeon Message |
| 0a | Carrier Pigeon iMessage LiteMessage |
| 0b | Carrier Pigeon Fetch Message |
| 0c | Satellite SMS Start Message |
| 0d | Satellite SMS Message |

Fig. 10: Basic message format.

map together with all the ground stations. The satellites retrieved from a public database are shown in green, with labels following the public naming scheme *M<xxx>*. Satellites taken from the iPhone targets list are shown in red (mostly overshadowed by the overlapping green ones), and follow a naming scheme *A<identifier>*, as specified in the targets list. Ground stations are shown in black, together with a pessimistic coverage radius of 2 000 km [76] in light red. We can clearly see that where green and red points overlap, satellite numbers and identifiers match.

TABLE II: Satellites in use by Apple as of November 2024.

| Sat Name | NORAD ID | COSPAR ID | Launch Date |
|---|---|---|---|
| M066 | 32265 | 2007-048C | 20.10.2007 |
| M069 | 31573 | 2007-020C | 29.05.2007 |
| M071 | 31576 | 2007-020F | 29.05.2007 |
| M072 | 31574 | 2007-020D | 29.05.2007 |
| M073 | 37193 | 2010-054F | 19.10.2010 |
| M074 | 37189 | 2010-054B | 19.10.2010 |
| M075 | 37192 | 2010-054E | 19.10.2010 |
| M076 | 37190 | 2010-054C | 19.10.2010 |
| M077 | 37191 | 2010-054D | 19.10.2010 |
| M078 | 39076 | 2013-005E | 06.02.2013 |
| M079 | 37188 | 2010-054A | 19.10.2010 |
| M080 | 38041 | 2011-080B | 28.12.2011 |
| M081 | 37743 | 2011-033E | 13.07.2011 |
| M082 | 38042 | 2011-080C | 28.12.2011 |
| M083 | 37739 | 2011-033A | 13.07.2011 |
| M084 | 38040 | 2011-080A | 28.12.2011 |
| M085 | 37742 | 2011-033D | 13.07.2011 |
| M086 | 38045 | 2011-080F | 28.12.2011 |
| M088 | 37740 | 2011-033B | 13.07.2011 |
| M089 | 37744 | 2011-033F | 13.07.2011 |
| M090 | 38044 | 2011-080E | 28.12.2011 |
| M091 | 37741 | 2011-033C | 13.07.2011 |
| M092 | 38043 | 2011-080D | 28.12.2011 |
| M093 | 39073 | 2013-005B | 06.02.2013 |
| M094 | 39074 | 2013-005C | 06.02.2013 |
| M095 | 39077 | 2013-005F | 06.02.2013 |
| M096 | 39075 | 2013-005D | 06.02.2013 |
| M097 | 39072 | 2013-005A | 06.02.2013 |

A list of all satellites currently in use together with their launch dates is shown in Table II. We can see that a range of satellites launched between 29.05.2007 and 06.02.2013 are currently in use, the oldest being *M069*.

### B. Allowed Services Bitmask

*CommCenter* internally holds a structure to track currently allowed services. Whenever the method `-[CTStewieState setAllowedServices:]` is called, we overwrite this with `0xffff`. As Table III shows, this enables all services at once.

### C. Satellite Messages Packet Format

There are various application-specific payloads. As they follow different encryption and compression schemes, their packet formats are highly optimized for the specific use case. Figure 10 lists all supported packet types as of iOS 18. Each of these message types has a different bitwise representation.

Emergency texting and roadside assistance follow the same structure but use separate message types. Figure 12 shows the three messages used for emergency communication over satellite. The communication starts with the message shown in Figure 12a, which compresses the most important information into a few bytes to be sent as fast as possible. The communication is then followed by text messages shown in Figure 12b. Every few minutes, a location update is generated automatically (see Figure 12c). The display in the emergency messaging app indicates these updates.

TABLE III: Service bitmask for allowed satellite services.

| Mask Bit | Meaning |
|---|---|
| `0x8000` | Anywhere |
| `0x4000` | Test |
| `0x0020` | Satellite SMS |
| `0x0010` | iMessage Lite |
| `0x0008` | Roadside |
| `0x0004` | Find My |
| `0x0002` | Emergency Try Out |
| `0x0001` | Emergency |

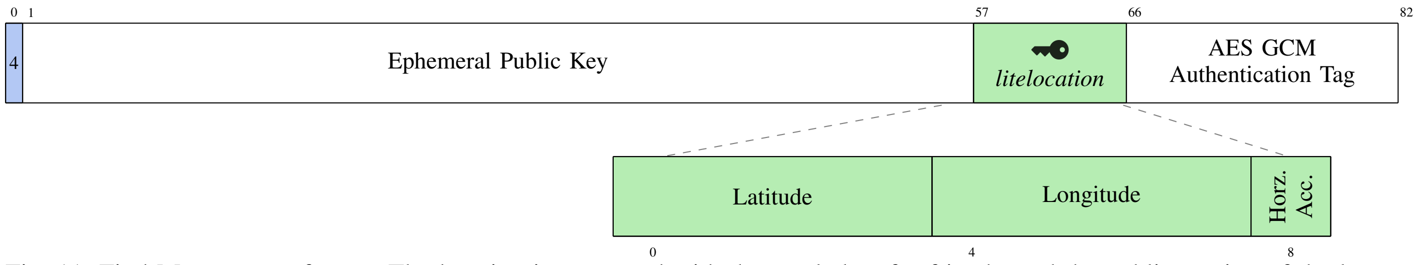
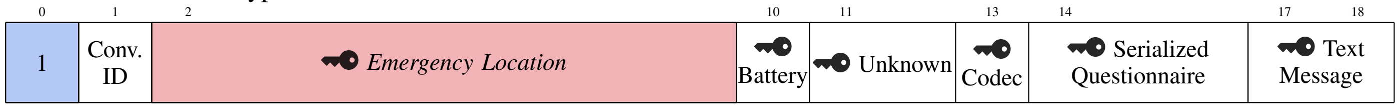
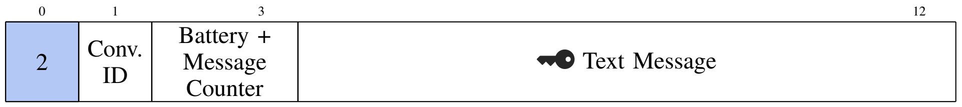


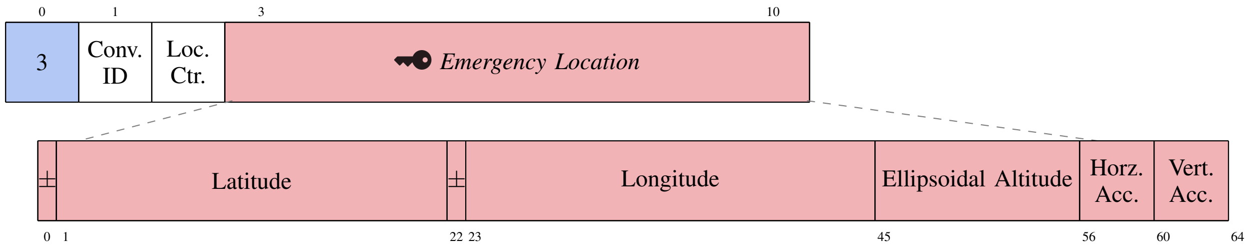
Fig. 11: Find My message format. The location is encrypted with the user's key for friends, and the public portion of the key is added for later decryption.



(a) Emergency start message. It contains all essential information about the emergency, entered by the user during the initial questionnaire, along with further information like the battery level and location. The preferred codec ID indicates the language for the conversation.



(b) Emergency text message sent during an emergency conversation. The maximum text message length is 160 byte.



(c) Emergency message location update. Sent when the user continues a previous conversation after a pause or if the location changed significantly. The location is only 8 byte and uses a bitwise format.

Fig. 12: Emergency SOS messages in the entire protocol flow. First, an emergency start message is sent. The remaining communication consists of text messages, with location updates sent automatically every few minutes.

## D. Responsible Disclosure: Satellite Message Integrity

Apple Product Security, August 2024: *After examining your paper, we determined that Emergency SOS via Satellite & Roadside Assistance via satellite messaging protocol is functioning as expected. When our teams designed this protocol, we had to factor in the extremely limited bandwidth associated with satellite communications. This meant optimizing the protocol and incorporating some regulatory factors as well since these are messages for emergency first responders.*

*First off, the entire Emergency SOS via Satellite message is encrypted while it travels to an Apple data center. To help protect the integrity of the message, the "outer payload" is decrypted by Apple's satellite service server to provide routing information. Once that layer is decrypted, the inner encrypted payload is routed based on the now decrypted metadata to the right service. This protection ensures the confidentiality of sensitive emergency payloads during their transit within Apple's infrastructure.*

*If it's an emergency message, Apple's emergency service server then decrypts that inner payload. Only servers directly interfacing with emergency services are permitted to access the derived HKDF-keys from the Key Management Server. Please keep in mind, all of this processing takes place within Apple data centers too.*

*Now, as for the encryption scheme you're describing, it is a security measure employed by Apple data centers to safeguard plaintext messages from being intercepted and forwarded between servers. This scheme first requires successful decryption of the link layer encryption, which already incorporates a message authentication code, before the payload can be decrypted.*

*Considering that the entire Emergency SOS via Satellite message is encrypted until it reaches an Apple data center, the extremely limited bandwidth constraints of satellite communication, and the time-critical nature of emergency communications, adding a second message authentication code to a subset of the payload that had already undergone authentication does not provide additional benefit to the current model.*

*We hope that this information helps clarify how encryption for Emergency SOS via Satellite works and the intent behind the messaging protocol. Thank you again for submitting your research to us. We appreciate your effort and look forward to seeing more of your work in the future.*

## Artifact Appendix

We make three distinct artifacts available:

1) A Wireshark dissector for QMI satellite messages.
2) A simulation-based testbed for the iPhone 13, allowing research into satellite features like emergency texting without sending actual satellite messages.
3) A satellite messenger, using Find My satellite location sharing as a covert channel to send arbitrary text messages.

We include our artifacts for evaluation to verify our claims and make them available, enabling other researchers to built upon our work.

### A. Description & Requirements

**1) How to Access:** We publish our artifacts in a Zenodo repository at https://zenodo.org/records/13863531 [37].

**2) Hardware Dependencies: Simulation-based Testbed:** A jailbroken or SRD iPhone 13 or 13 mini. While we did not have further devices available for testing, our scripts are also likely to work with the iPhone 12 series.

**Satellite Messenger:** Message transmission requires a jailbroken iPhone that has satellite transmission capabilities. At the time of writing, the only jailbreakable iPhone with a satellite modem is the iPhone 14. Dopamine supports jailbreaking iOS 16.1 to iOS 16.5.1. To install our app, TrollStore is required, which supports these versions and can be downloaded by following this guide.

Message reception is possible with any jailbreakable iPhone. We confirmed compatibility with an iPhone 8 on iOS 16.

**3) Software Dependencies: Wireshark Dissector:** A running instance of Wireshark. No specific version is required.

**Simulation-based Testbed:** The iPhone should run iOS 16.3 or 16.4.1. Frida must be installed and running on the iPhone. The host machine should have Python 3 and the Python packages `frida-tools`, `pwntools`, and `aarch64-elf-binutils` installed.

**Satellite Messenger:** We require Xcode 14 or newer to build the app and the Theos toolchain to compile the tweak. Theos can be downloaded from their website.

### B. Artifact Installation & Configuration

**Wireshark Dissector:** The dissector is written in Lua and named `qmi_dissector_satellite.lua`. For installation, it must be copied to the Wireshark plugion directory, i.e., `~/.local/lib/wireshark/plugins/`. In Wireshark, configure the `DLT_USER` protocol: Open Wireshark preferences → *Protocols* → *DLT_USER* → *Edit encapsulation table*. The final table should be configured as shown in Table IV.

**Simulation-based Testbed:** The simulation-based testbed is enabled through Frida hooks that mimic satellite modem behavior towards *CommCenter* and overwrite flags that satellite communication was supported. Frida scripts containing these hooks are contained in our repository. Attach the iPhone via USB to your host and execute the following Frida commands on the host machine.

First, *CommCenter* must already be hooked early on during its initialization to successfully set up satellite communication capabilities. To this end, stop the *CommCenter* process by executing `frida-kill -U CommCenter`. iOS will automatically restart *CommCenter* within a few seconds. Within this time window, quickly attach the script using the following command: `frida -U -W CommCenter -l commcenter_stewie.js`.

On the SRD, Frida does not support spawn gating required for attaching to a process on startup, and *CommCenter* must be patched to wait on startup. We provide instructions in the example cryptex.

After attaching the script to the newly starting *CommCenter*, wait for the script until you see the following message:

```
Ready to emulate Stewie on SRD!
[iPhone::CommCenter ]-> GsmRadioPersonality
 ::create called with hw_model=0x25,
 replacing with 0x34 for iPhone 14!
```

This confirms that the radio personality was set to a version that has satellite support. Then, in the Frida console, enter the following command: `openURL("x-apple-sosbuddy://request?reason=OfferEmergencyTryOut")`. This should lead you through the demo mode and is required to initialize further satellite modem state.

The *CommCenter* script defines some global variables at the beginning. Use the `alwaysEmergency` and `emergencyType` variables to choose which satellite functionality you want to test. E.g., the type 2 is for emergency texting, while the type 5 enables testing Find My.

While this is sufficient to use some satellite services, the Find My implementation holds further states that prevent simulation of sharing locations over the Find My app. Thus, a second script is required to be attached to *searchpartyd*. The attachment procedure is similar to the previous one: `frida-kill -U searchpartyd; frida -U searchpartyd -l findmy_stewie.js`.

Afterward, restart *locationd*: `frida-kill -U locationd`. This step is only required to reset its state, attaching a script is not required.

**Satellite Messenger:** The satellite messenger can be installed via TrollStore. Install the available `StewieMessenger.tipa` file or build and sign the app binary as described in the `README`. Then, install the tweak, which dynamically replaces method calls, allowing the app to initiate satellite communication. Copy the `.deb` file to the iPhone and install it by executing `dpkg -i tweak.deb` or using Sileo. This iPhone can now send text messages via satellite to anyone it shares its location with on

TABLE IV: Wireshark configuration of the `DLT_USER` encapsulation table to use our QMI dissector.

| DLT | Payload dissector | Header size | Header dissector | Trailer size | Trailer dissector |
|---|---|---|---|---|---|
| User 0 (DLT=147) | qmi | 0 | | 0 | |

Find My. We recommend installing the app on the recipient's jailbroken iPhone to view the text message.

## C. *Major Claims*

Below are our major claims:

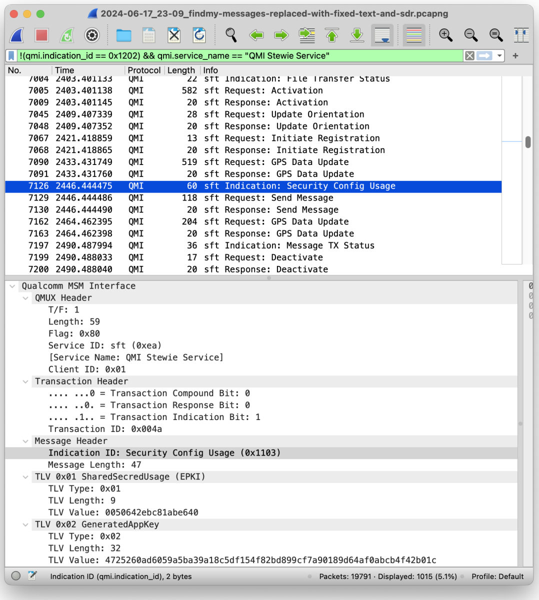
- C1 We document the steps of the satellite communication protocol (see Section V). This can be reproduced by experiment (E1).
- C2 In we claim that we can use iPhone's satellite features in restricted regions (see Section V-E). This can be validated with our QMI log recorded in Poland and experiment (E1).
- C3 We build a satellite simulation-based testbed (see Section IV-A). The testbed can be set up following the steps above, and experiment (E2) demonstrates how it works.
- C4 We claim that we can misuse Apple's satellite channel to send text messages via satellite (see Section VI-D). This can be demonstrated with experiment (E3).

## D. *Evaluation*

This section outlines the necessary experiments to validate if our artifacts are functional and if the results can be reproduced.

**1) Experiment (E1):** Set up the Wireshark dissector, and then open an exemplary `.pcapng` file, which contains satellite communication. Viewing the file verifies that our dissector is functional, and by analyzing the file contents, one can verify the claims (C1) and (C2).

**How To:** Install the Wireshark dissector as described in Section B. Open the provided `.pcapng` file. Wireshark should automatically dissect the QMI messages in this file and present the communication between the baseband and OS. The file can be used to verify claims (C1) and (C2).



```
2024-06-17_23-09_findmy-messages-replaced-with-fixed-text-and-sdr.pcapng
!(qmi.indication_id == 0x1202) && qmi.service_name == "QMI Stewie Service"
No.   Time         Protocol Length Info
7004 2403.401133   QMI        22 sft Indication: File Transfer Status
7005 2403.401138   QMI       582 sft Request: Activation
7009 2403.401145   QMI        20 sft Response: Activation
7045 2409.407339   QMI        28 sft Request: Update Orientation
7048 2409.407352   QMI        20 sft Response: Update Orientation
7067 2421.418859   QMI        13 sft Request: Initiate Registration
7068 2421.418865   QMI        20 sft Response: Initiate Registration
7090 2433.431749   QMI       519 sft Request: GPS Data Update
7091 2433.431760   QMI        20 sft Response: GPS Data Update
7126 2446.444475   QMI        60 sft Indication: Security Config Usage
7129 2446.444486   QMI       118 sft Request: Send Message
7130 2446.444490   QMI        20 sft Response: Send Message
7162 2464.462395   QMI       204 sft Request: GPS Data Update
7163 2464.462398   QMI        20 sft Response: GPS Data Update
7197 2490.487994   QMI        36 sft Indication: Message TX Status
7199 2490.488033   QMI        17 sft Request: Deactivate
7200 2490.488040   QMI        20 sft Response: Deactivate

Qualcomm MSM Interface
  QMUX Header
    T/F: 1
    Length: 59
    Flag: 0x80
    Service ID: sft (0xea)
    [Service Name: QMI Stewie Service]
    Client ID: 0x01
  Transaction Header
    .... ...0 = Transaction Compound Bit: 0
    .... ..0. = Transaction Response Bit: 0
    .... .1.. = Transaction Indication Bit: 1
    Transaction ID: 0x004a
  Message Header
    Indication ID: Security Config Usage (0x1103)
    Message Length: 47
  TLV 0x01 SharedSecredUsage (EPKI)
    TLV Type: 0x01
    TLV Length: 9
    TLV Value: 0050642ebc81abe640
  TLV 0x02 GeneratedAppKey
    TLV Type: 0x02
    TLV Length: 32
    TLV Value: 4725260ad6059a5ba39a18c5df154f82bd899cf7a90189d64af0abcb4f42b01c

Indication ID (qmi.indication_id), 2 bytes    Packets: 19791 · Displayed: 1015 (5.1%)    Profile: Default
```

Fig. 13: Wireshark dissector for the satellite via QMI.

**Execution:** To verify claim (C1), filter for satellite messages using `qmi.service_name == "QMI Stewie Service"`. Then, verify that all steps match those in Figure 6. To verify claim (C2), open message number 2 905 in the example file.

**Results:** (C1): The steps presented in Figure 6 can be viewed in our `.pcap` file. (C2): Message number 2 905 contains location data for Poland (`0x06`). Selecting the relevant bytes will highlight the corresponding hexdump on the right, including the string *POL*—the country identifier for Poland.

Figure 13 shows the Wireshark dissector for QMI messages exchanged between iOS and the baseband chip. The trace is the same as used to annotate the SDR recording in Section V-H. The visible steps include the end of the config file transfer. The trace ends after a successful Find My Friends location update and then deactivation of the satellite connection.

**2) Experiment (E2):** In this experiment, we demonstrate a simulation-based testbed for satellite communication.

**How To:** Extract the CommCenter binary from the iPhone, patch it, and reinstall it on the device. Next, kill the running CommCenter process and attach our Frida scripts.

**Preparation:** Jailbreak the iPhone and install Frida. When using an SRD, install the example cryptex.

**Execution:** To verify claim (C3), follow the steps in the `README`. Then run the commands specified in Section B. When the simulation-based testbed is running, open the *Find My* app, navigate to the *Me* tab, and try to send a location via satellite. The satellite app *SOSBuddy* should open in Emergency SOS mode. No real emergency message will be sent because the iPhone 13 does not support satellite communication.

**Results:** (C3): The satellite capabilities should be visible on the otherwise unsupported iPhone 13. E.g., the *Share location via satellite* button in the *Find My* app. When trying to send the location, the behavior is modified to use the Emergency SOS feature.

**3) Experiment (E3):** Installing the app via TrollStore should be straightforward on a jailbroken iPhone 14. You can then immediately start sending messages via satellite.

**How To:** Install the app and demonstrate sending text messages via satellite.

**Preparation:** Jailbreak the iPhone 14, install TrollStore, and ensure that you are sharing your location via Find My with at least one person.

**Execution:** Install the satellite messenger by sharing the `Satellite.tipa` file with the iPhone and installing it with TrollStore. Then install our tweak `tweak.deb` by sharing it with the iPhone and executing `dpkg tweak.deb`. To verify claim (C4), send a message via satellite.

**Results:** (C4): When sending a message, the user should see the satellite interface, assisting them in pointing to the next satellite to send the message.