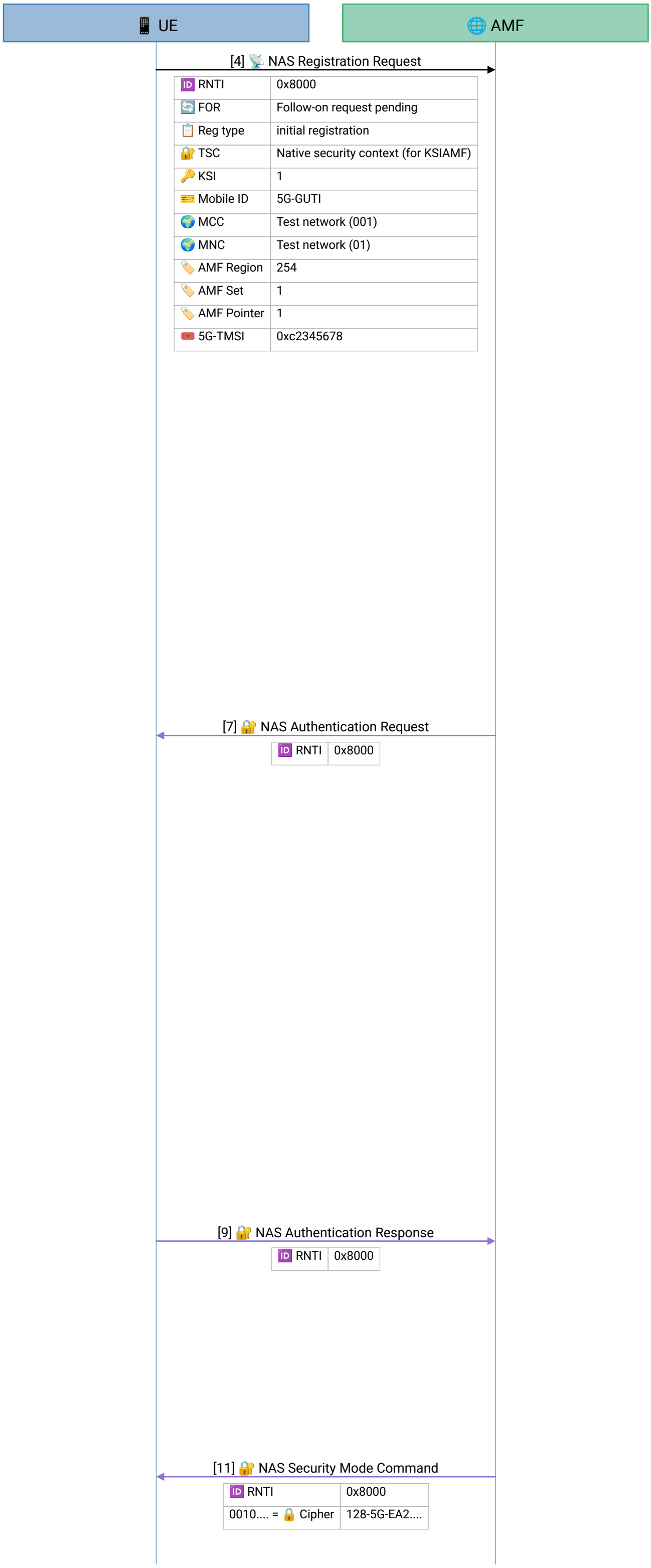



**[Frame 4] The first NAS message of the capture, and it is already half sealed.** This Registration Request rides inside the RRC Setup Complete, and its outer envelope reads `Security header type: Integrity protected (1)`, message authentication code `0x97a0a3f4`, `Sequence number: 6`. Integrity protected, not ciphered — and sequence number 6 rather than 0, so the UE is reusing a NAS security context it already held before the capture began. `NAS key set identifier: 1`, `Type of security context flag (TSC): Native security context (for KSIAMF)`. The readable 5GMM body:

- `5GS registration type: initial registration (1)`, with the `Follow-On Request bit (FOR)` set to `Follow-on request pending` — the UE is telling the AMF it has more to send once registration completes.
- `5GS mobile identity`, length 11, `Type of identity: 5G-GUTI (2)`: `MCC: Test network (001)`, `MNC: Test network (01)`, `AMF Region ID: 254`, `AMF Set ID: 1`, `AMF Pointer: 1`, `5G-TMSI: 0xc2345678`. `UE security capability`, element ID `0x2e`, length 2. Supported: `5G-EA0`, `128-5G-EA1`, `128-5G-EA2`, `128-5G-EA3`, `128-5G-IA1`, `128-5G-IA2`, `128-5G-IA3`. Not supported: `5G-EA4` through `5G-EA7`, `5G-IA0`, and `5G-IA4` through `5G-IA7`. And then the part that sets the tone for this whole page. The last IE is a `NAS message container`, element ID `0x71`, `Length: 61`, and its contents are `Encrypted data`. The RRC layer handed NAS a 92-octet payload; 61 of those octets are unreadable inside a message whose envelope reads in full.

This is TS 24.501 §4.4.6 working exactly as specified. An initial NAS message sent when a security context already exists is transmitted twice over: a cleartext skeleton carrying only the IEs the AMF needs before it can decipher anything — registration type, ngKSI, mobile identity, UE security capability — and the complete message, ciphered, in the container. Everything else the UE wants to say is in the sealed 61 octets.

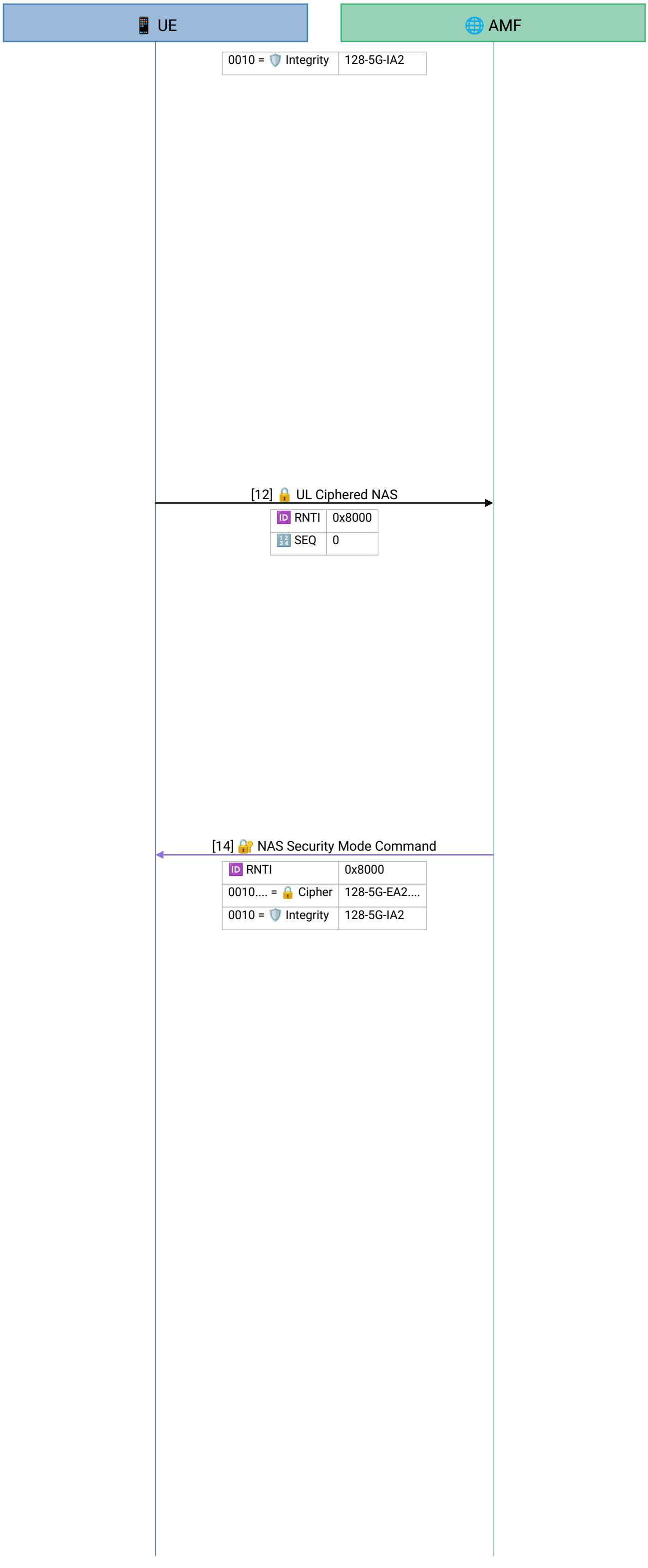
**[Frame 7] The AMF challenges, in the clear.** `128.772 ms` after the Registration Request. `Security header type: Plain NAS message, not security protected (0)` — the only NAS message in this session with no security protection at all, because the AMF has decided to authenticate afresh and the new context does not exist yet.

- `NAS key set identifier - ngKSI: Native security context`, value `2`. Frame 4 arrived under ngKSI 1; this Authentication Request mints ngKSI 2. From the Security Mode Command onward, every NAS message in the capture that names a key set names that one — frames 11, 14, 133 and 150.
- `ABBA`, length 2, contents `0000`. The Anti-Bidding down Between Architectures parameter, bound into the key derivation so a downgrade to a weaker generation cannot be forced.
- `Authentication Parameter RAND`, element ID `0x21`: `a3de0c6d363e30c364a4078f1bf8d577`.
- `Authentication Parameter AUTN`, element ID `0x20`, length 16: `6e323b36c46c8000a3df0e6e323bb6c4`, which Wireshark splits into SQN `xor AK: 6e323b36c46c`, `AMF: 8000` and `MAC: a3df0e6e323bb6c4`. Note what a plain-text challenge does and does not leak. RAND and AUTN are public by design — they are useless without the long-term key K in the USIM and in the home network. The AMF field here is the Authentication Management Field of TS 33.102, not the core-network node; the node is also called AMF and this is a genuine 5G naming collision.

**[Frame 9] The UE answers, protected under the old context.** `40.223 ms` later. `Security header type: Integrity protected (1)`, message authentication code `0x891cabb4`, `Sequence number: 7` — the next uplink sequence number after frame 4's 6, still counting in the pre-existing context. The new keys are derived but not yet in force; that is the Security Mode procedure's job.

The body is one IE: `Authentication response parameter`, element ID `0x2d`, length 16, `RES: d7c360c5a57c8f4f03bd1e406604eec2`. That 16-octet value is RES*, the response TS 33.501 has the UE compute from RAND, K and the serving-network name; the AMF forwards it for comparison against the expected XRES*.

**[Frame 11] The first Security Mode Command, and the request that makes this page interesting.** `76.777 ms` later. `Security header type: Integrity protected with new 5GS security context (3)`, message authentication code `0x41bf0b2b`, `Sequence number: 0`




— the downlink counter restarts at zero because this is the first message of a brand-new NAS security context.

- `NAS security algorithms: Type of ciphering algorithm: 128-5G-EA2 (2), Type of integrity protection algorithm: 128-5G-IA2 (2).` TS 24.501 spells these 5G-EA and 5G-IA; TS 33.501 spells the same two algorithms NEA2 and NIA2. AES in counter mode, and AES-CMAC.
- `NAS key set identifier - ngKSI: native, 2` — the context frame 7 announced.
- `UE security capability - Replayed UE security capabilities, length 2.` Bit for bit what the UE sent in the clear at frame 4. Replaying them is a bidding-down defense: the UE checks the echo against what it sent, and any tampering en route shows up here.
- `Additional 5G security information, element ID 0x36, length 1: Retransmission of initial NAS message request (RINMR): Requested, Horizontal derivation parameter (HDP): Not required.` That RINMR bit is the one to watch. It tells the UE to put the *entire* initial NAS message — the complete Registration Request, not the cleartext skeleton — into the NAS message container of its Security Mode Complete. Frame 11 is the only frame in the capture that sets it.

**[Frame 12] The conversation goes dark, precisely here.** `14.229 ms` later. `Security header type: Integrity protected and ciphered with new 5GS security context (4), message authentication code 0x97f62856, Sequence number: 0` — the uplink counter's own restart under the new context — and then a single line: `Encrypted data.` This is the Security Mode Complete, and it is the first NAS message of the capture whose body cannot be read. It is a 74-octet container, and none of it dissects. Nothing was lost, nothing is malformed: NAS ciphering is now in force, the keys that would open it were never published, and no amount of Wireshark configuration will change that. Forcing `nas-5gs.null_decipher` on a genuinely ciphered body does not decrypt it — it makes the dissector parse ciphertext as if it were IEs and invent 5GMM fields that were never sent. This flow deliberately does not enable it.

What we can still read is the envelope: header type, message authentication code, sequence number. That envelope turns out to carry more than you would expect — see frame 14.

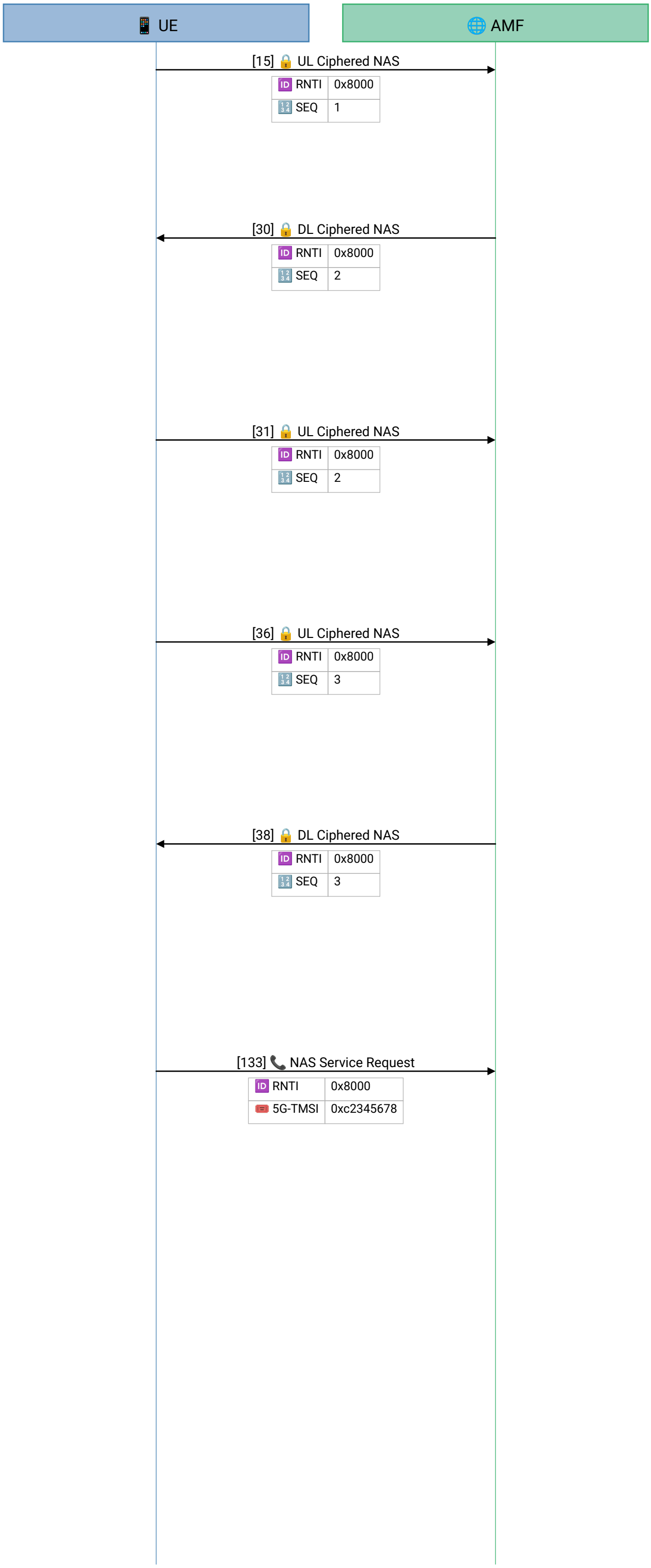
**[Frame 14] A second Security Mode Command — and it is a different message, not a retransmission.** `88.769 ms` later. `Security header type: Integrity protected with new 5GS security context (3), message authentication code 0x7acf026e, Sequence number: 1.` The downlink counter advanced from 0, so the AMF genuinely sent a second command.

It repeats three things from frame 11 exactly: `128-5G-EA2 / 128-5G-IA2, ngKSI native 2,` and the same replayed 5G UE security capabilities. It drops the `Additional 5G security information` IE, so RINMR is no longer requested. And it adds two IEs frame 11 did not have:

- `NAS security algorithms - Selected EPS NAS security algorithms, element ID 0x57: EPS encryption algorithm 128-EEA1 (1), EPS integrity algorithm 128-EIA1 (1).` `UE security capability - Replayed S1 UE security capabilities, element ID 0x19, length 4.` Supported: `EEA0, 128-EEA1, 128-EEA2, 128-EEA3, 128-EIA1, 128-EIA2, 128-EIA3, UEA0, UEA1, UIA1.` Not supported: `EEA4` to `EEA7, EIA0, EIA4` to `EIA6, EPS-UPIP, UEA2` to `UEA7,` and `UIA2` to `UIA7.` Those two IEs are for interworking with the 4G core: they pre-agree which EPS algorithms apply if this UE ever moves to S1 mode. And they are the closest this capture comes to reading a ciphered message.

Follow the chain, and note that it is the trace that carries it, not the specification. Frame 4's cleartext `UE security capability` IE is **length 2** — two octets of 5G-EA and 5G-IA bits, with no EPS or UMTS bits in them at all. A query across the whole capture finds EPS and UMTS algorithm bits on exactly one frame: this one, where the AMF is echoing them back. TS 33.501 §6.7.2 requires those replayed capabilities to be the ones the AMF received from the UE. So the AMF is quoting bits that never crossed this trace in the clear, one message after asking for the full initial NAS message and receiving frame 12's sealed 74 octets. The sealed 61 octets in frame 4 and those 74 in frame 12 are the only places they can have come from.

We cannot open frame 12. We can bound what was inside it.




**[Frame 15] The second Security Mode Complete, also sealed.** `11.229 ms` later. Header type 4 again, message authentication code `0x104cf5ef`, `Sequence number: 1`, `Encrypted data`. A 10-octet container against frame 12's 74 — consistent with a Security Mode Complete that has no initial NAS message to carry this time, because frame 14 did not ask for one.

From here to the end of this RRC connection, no NAS body reads at all.

**[Frame 30] Steady state, downlink.** `243.782 ms` later, and the security header type settles to `Integrity protected and ciphered (2)` — type 2 rather than type 4, because the "new 5GS security context" marker belongs only to the messages that activate a context. Message authentication code `0x6b39d3f8`, `Sequence number: 2`, `Encrypted data`, 53-octet container.

This is what a working NAS connection looks like without keys: an envelope and a length. The direction, the counter and the size are all you get, and for a surprising amount of troubleshooting they are all you need.

**[Frame 31] Steady state, uplink — and two counters, not one.** `16.213 ms` later. Header type `2`, message authentication code `0x7bc1bd11`, `Sequence number: 2`, `Encrypted data`, 10-octet container.

Both frame 30 and frame 31 carry sequence number 2 and they are unrelated. NAS keeps a separate COUNT per direction, exactly as PDCP does on the radio side: the downlink chain here runs 0 (frame 11), 1 (frame 14), 2 (frame 30), and the uplink chain runs 0 (frame 12), 1 (frame 15), 2 (frame 31). Reading a NAS sequence number without noting its direction is a reliable way to misdiagnose a replay.

**[Frame 36] A 92-octet uplink message, and no way to name it.** `1.368993 s` later — the longest quiet stretch of the session so far. The radio layers were not idle through it: frames 33, 34 and 35 are Timing Advance commands, the cell keeping the UE's uplink aligned while NAS had nothing to say. Header type `2`, message authentication code `0x35b37a52`, `Sequence number: 3`, `Encrypted data`, and a 92-octet container.

92 octets is exactly the size of frame 4's whole NAS payload. That is all the shape we get. Whatever 5GMM or 5GSM procedure this is, its message type is inside the ciphertext.

**[Frame 38] The frame the Wireshark issue was filed about.** `301.786 ms` later. Header type `2`, message authentication code `0x9b072356`, `Sequence number: 3`, `Encrypted data`, and a 122-octet container — the largest NAS container in this session.

This is the NAS payload riding inside the RRC Reconfiguration that builds the data bearer. The RRC around it decodes completely, down to the RLC timers and the SDAP header switches. The NAS inside it does not decode at all. One frame, two security contexts, one key set published and one not.

It is also the last NAS message of this RRC connection.

**[Frame 133] A new radio connection, the same NAS context.** `13.629329 s` after frame 38. In between, the radio side released the connection at frame 128 and set up a fresh one at frames 131 and 132 — and then re-ran the Access-Stratum Security Mode procedure from scratch immediately *after* this message, at frames 136 and 137.

Read that ordering carefully, because it is the whole point in miniature. This frame is the RRC Setup Complete of the new connection, and the NAS Service Request rides inside it. So the Service Request goes out **before** the new Access-Stratum context exists — integrity protected by NAS keys that predate the radio connection carrying it. Nothing about the release touched NAS. This Service Request reads `Security header type: Integrity protected (1)`, message authentication code `0xb98305d2`, `Sequence number: 4` — continuing straight on from frame 36's uplink 3 — and `NAS key set identifier: 2`, `Type of security context flag (TSC): Native security context (for KSIAMF)`: still the context frames 7 to 14 established. The NAS security context survived an idle period that destroyed the radio one entirely. That is the two-layer split stated as plainly as a trace can state it.

The readable body:

- `Service type: Mobile terminated services (2).` The network paged the UE; the UE is answering.
- `5GS mobile identity`, length 7, `Type of identity: 5G-S-TMSI (4)`: `AMF Set ID: 1`, `AMF Pointer: 1`, `5G-TMSI: 0xc2345678` — the same TMSI frame 4's

📱 UE

🌐 AMF

5G-GUTI carried.

- NAS `message container`, element ID `0x71`, `Length: 17`, `Encrypted data`. Header type 1 and a sealed container: the same cleartext-IE rule as frame 4, on a different message. A Service Request is an initial NAS message too, so it is sent integrity protected but not ciphered, with only the IEs the AMF needs up front in the clear and the rest in the container. That is why this page can read its last message after six sealed ones — not because anything was decrypted, but because 3GPP requires the opening move of a NAS connection to be legible.

One NAS message of the capture is not on this timeline. Frame 150 is another Service Request, `33.279932 s` after frame 133 and so past the session extractor's 30-second idle timer. It carries `Sequence number: 5` under the same ngKSI 2 — the uplink NAS count still marching on, on the capture's third radio connection.

Session Result: timeout — idle 30.000s after last activity (active span: 15.920s)