Computer Security - Big Project: 5G security technical report

Theodor Signebøen Midtlien
Faculty IV - Electrical Engineering and Computer Science
Technische Universität Berlin
Berlin, Germany
t.midtlien@campus.tu-berlin.de

***Abstract*—This is a technical report of the course "Computer Security - Big Project" at the Technische Universität Berlin. The main goal of the project was to create a testbed to explore the implementation correctness of encryption and integrity negotiation between the 5G core network and commercial-off-the-shelf (COTS) user equipment (UE). A test setup was created with a modified open-source software 5G core with a Software Defined Radio (SDR) for the base station. However, COTS mobile phones were never able to connect to our network. Test cases to check security messages in the 5G control plane were done on a Queltec 5G research module. During the development of the testbed, a security bug was found in Open5gs which was reported and fixed. For further research on the topic, an easy-to-deploy VM-based testbed was created, with tooling that supports test cases to modify control plane messages.**

***Index Terms*—5G, security, mobile telecommunication, network protocols**

I. Introduction

The fifth generation of wireless networks, commonly referred to as 5G, promises a revolution in terms of speed, connectivity, and user experience. 5G aims to be the enabling technology for autonomous vehicles, massive Internet of Things (IoT), and the tactile internet. Telecom providers are rushing to deploy 5G networks to consumers, as it is estimated that 5G-enabled phones will account for 62% of smartphones shipped worldwide in 2023 [8]. This generation of mobile telecommunication promises to improve security. However, with every new generation, security issues have been found.

Rupprecht et al. explored the implementation correctness of security measures related to data encryption and network authentication in LTE networks [16]. They made a framework for testing LTE-enabled devices with their own LTE network and a Software Defined Radio (SDR). The implementation of the Security Mode Command was found not to be up to the specification for several COTS phones. Successful man-in-the-middle exploits were presented in their work. The project described in this report takes inspiration from their LTE-security research into the realm of 5G, exploring implementations of commercial-off-the-shelf (COTS).

During the execution of this project, researchers from the National University of Defense Technology in China released a paper doing research on security implementations of COTS [19] phones. They extracted security requirements from the 5G specification into concrete inspection cases to be verified. Implementation flaws were found and two PoC attacks were implemented, showing that security is still an issue in 5G. Their test setup was created with a modified version of a popular open-source 5G core network project, in a similar fashion as done in this project. At the time of writing, no code has been released for their project.

The rest of this report is structured as follows. Section II introduces preliminaries on the 5G architecture, registration procedure, and security measures. In section III the test setup implementation and problems that occurred are presented. The section coming after shows some experimental results of the test cases. Then section V describes a bug found in Open5gs. Section VI is about an easy-to-deploy testbed based on virtual machines. A short discussion about lessons learned is given in section VII, and lastly, section VIII motivates further research and the next steps after this project is finished.

II. 5G Prelimiary

A. 5G architecture

Figure 1 is an overview of the 5G standalone architecture, defined in 3GPP TS 23.501 [2]. There are three main components: 5G core (5GC), gNodeB, and user equipment (UE). The interfaces running between the components and internally are labeled. The network is logically split into a control plane and a user plane. The control plane carries signaling traffic regarding establishing links and authentication. While the user plane carries user data to and from external networks.

The 5GC network utilizes a Service-Based Architecture (SBA). Services are cloud-native applications connected by a Service Based Interface (SBI), where services expose a standardized API using HTTP. Each service, also called Network Functions (NF), has to register to the Network Repository Function (NRF). The NRF allows services to discover all other NFs, and enable load-balancing when multiple instances of the same NF exist in the cloud environment. This architecture allows self-contained services to be virtualized on general-purpose hardware, ensuring flexibility and scalability to telecom providers by orchestration. The Access and Mobility Function (AMF) is a central network function that interacts with both gNodeB and UE, on the N2 and N1 interface

respectably. It has functionality for the following (but not limited to): registration management, connection management, access authentication and authorization, and NAS ciphering and integrity protection.

gNodeB is a base station that supports the new radio access network that user equipment connects to. This connection is provided with the wireless Uu interface. The Uu interface runs a functional layer called Access Stratum (AS) which is the layer of the protocol stack that manages access on the radio interface and transports data. On the N2 interface, the AMF and gNodeB use the Next-Generation Application Protocol (NGAP) to do control plane signaling.

UE is a 5G-enabled mobile phone. Each UE has a Subscriber Identity Module (SIM) to identify itself in the mobile network. A valid subscriber has a unique Subscription Permanent Identifier (SUPI) that is stored on the SIM-card. The SIM-card also contains pre-shared cryptographic keys with their home network provider, which is used for authentication and negotiating session keys for encryption and integrity protection. To communicate with the core network, the UE uses the Non-Access Stratum (NAS) layer that runs above the AS on the Uu interface and with NGAP on the N2 interface.

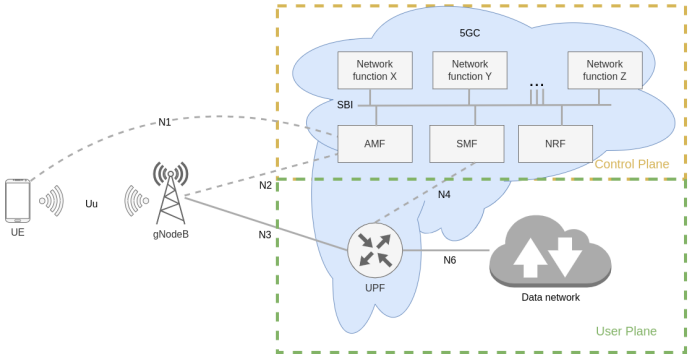

Fig. 1: 5G architecture with interfaces

B. 5G registration and security measures

Figure 2 shows the NAS registration procedure, defined in TS 24.501 [3]. This procedure is used when the UE wants to register to the core network and create a PDU session that tunnels data traffic to the data network. During the NAS registration request message, the UE announces which security algorithms it supports. This list of supported algorithms is stored in the AMF for the corresponding UE. In the Security Mode Command (SMC) message, the AMF chooses the highest version supported by the UE and sends the selected algorithms together with a replay of what the UE announced earlier. Upon receiving the Security Mode Command message, the UE verifies the MAC and checks that the replayed algorithms are not altered. If it accepts the message, the UE replies with a Security Mode Complete message and uses the algorithm picked by the AMF to encrypt- and integrity-protect the rest of the following control plane messages (protection of user plane is optional). The UE can

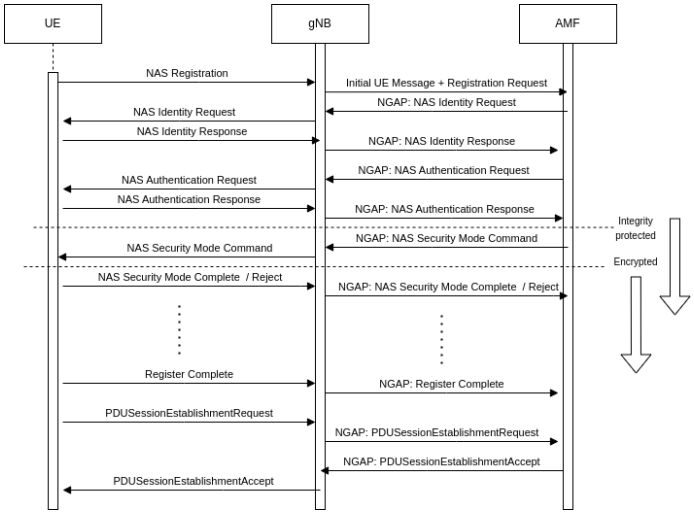

Fig. 2: NAS/NGAP registration procedure

Integrity Algorithms	Encrytion Algorithms	Based on
5G-IA0	5G-EA0	N/A
128-5G-IA1	128-5G-EA1	SNOW 3G
128-5G-IA2	128-5G-EA2	AES
128-5G-IA3	128-5G-EA3	ZUC
5G-IA4 - 5G-IA7	5G-EA4 - 5G-EA7	

TABLE I: NAS security algorithms

also reject the Security Mode Command with a Security Mode Reject message. The ”NAS security algorithms” field in the SMC is a 16-bit field divided into 2 octets. The second octet defines the type of ciphering algorithm in the 4 most significant bits and the type of integrity protection algorithm in the 4 least significant bits. The last bit of each type of algorithm is a spare bit, thus the message can support up to 8 (3 bits) different encryption and integrity algorithms. For now, the specification only defines 4 of each. Table I shows how the algorithms are defined. 5G-IA0 and 5G-EA0 indicate null integrity and null ciphering respectively. 5G-IA4 - 5G-IA7 and 5G-EA4 - 5G-EA7 are reserved for future use and not defined yet.

To provide confidentiality and prevent malicious behavior, 5G also mandates that the SUPI is never to be sent in clear text, but an encrypted version called the Subscription Concealed Identifier (SUCI). Earlier generation mobile networks sent their SUPI equivalent in clear text, which was a privacy issue and made spoofing viable.

III. Setup and Test Cases

A. Hardware

- Ettus Research USRP B210
- Quectel RMU500EK 5G module
- Delock NGFF(M.2) B Key to USB3.0 (With SIM Card Slot) Adapter
- Sysmocom sysoUSIM-SJS1/sysmoISIM-SJA2
- Panorama Antennas DMM-7-27-2SP
- iPhone 12 Pro (ios version 16.1.2)
- Motorola Moto G Stylus 5G (unkown android version)

- Dell XPS 13 7390 (Linux 6.4.12-arch1-1)
- Linode 8 GB, 4 Core shared CPU

B. Software

- Open5gs - 5G release 17 compliant. Modified version of v2.6.1 [12]
- srsRAN Project gNB version 23.5.0 [18]
- SCAT (5G enabled)
- Ubuntu 22.04 (5.15.0-82-generic)
- Python script
- MongoDB

Fig. 3: Software Defined Radio setup

C. Setup implementation details

Firstly, a software-only setup was created, where the RAN and UE were simulated in software. This was to make sure that the core network was set up correctly and working, before trying with a Software Defined Radio (SDR) by Ettus Research called Universal Software Radio Peripheral (USRP) as seen in Fig. 3. Additionally, time was used to inspect network traffic and to learn how the protocol worked. This setup consisted of Virtual Machines (VMs) on Linode, both running Ubuntu 22.04. One VM was the core network with Open5gs [14], an Open-Source implementation of 5GC written in C, and the other VM ran a simulated gNB and UE from UERANSIM [5]. A guide on Medium was followed for most of this setup [1]. An easily replicable modified version of this setup is presented in section VII.

After verifying that a PDU session could successfully be made, the simulated RAN was replaced with the USRP running srsRAN Project [17] v23.5.0 with the Ettus Research UHD radio driver [15] on a Dell XPS 13 laptop with 4 cores and 16GB ram. The simulated UE was replaced by a Quectel RMU500EK 5G module and board. For the module to be able to connect, a SIM card was needed. Both a Sysmocom sysmoISIM-SJA2 and the older sysoUSIM-SJS1programmable SIM cards were configured to have an IMSI/SUPI that was registered in the subscriber database in the core network. A modified version of SCAT [7] was used to generate 5G RRC/NAS GSMTAP packets, that were captured and analyzed in Wireshark with the scat lua dissector. This setup was able to successfully register and create a PDU session a few times, but a lot of time went into troubleshooting what caused the instability. An accident also occurred with the 5G module which will be explained together with other problems in subsection *D*.

To facilitate modification of the Security Mode Command, Open5gs was modified. The modified version hooked the security mode command builder function and communicated via TCP sockets with a Python script that read which EA/IA algorithm to use from a yaml file. The response from the UE of the modified security mode command message was stored in a Mongodb. This early PoC version was hardcoded and only supported test cases that modified the selected EA/IA and MAC in the security mode command message. Since time with the hardware was limited and a lot of troubleshooting had to happen simultaneously, this PoC version was used for preliminary experimental results. A more refined and modular version has since been created and is presented in section VI.

D. Challenges

The board that holds the Quectel 5G module was accidentally over-voltaged by connecting a stronger power adapter. The board does not even require a separate connected power adapter as it gets all its power from the USB3 interface. Unfortunately, the 5G module did not power on after the accident. A workaround was to still use the 5G module with the antennas on the board. An NGFF(M.2) to USB3 adapter with a SIM card slot was used to connect the 5G module directly to Ant.0 and Ant.3 as shown in Fig. 4.

Using the 5G module with SCAT, RSRP (average power level) measurements of the different frequency bands in the air were given. This measure highly affects the stability of the connection and if the module even connected to the 5GC successfully. Most of the time the registration failed early in the process during NR MAC RACH Msg3. Fine-tuning the transmit and receive power levels of the USRP, in addition to the physical distance and direction of the radio and the module antennas took a lot of tries. This is common for 5G as the radio waves are shorter and more directional and careful configuration has to be done to enable stable transmission.

Both the commercial off-the-shelf (COTS) UEs, (iPhone 12 Pro and Motorola Moto G Stylus 5G) were never able to detect or connect to the 5G core network. For the Android phone, an application called ”5G Only” was installed. It showed the 5G radio networks it detected with the Absolute radio-frequency channel number (ARFCN) and it’s corresponding signal strength. However, our network newer showed up in this app or SCAT which was also connected to the UE. Two main reasons could be the cause of this:

1. The antenna is only designed to work with 2G/3G/LTE in the range of 700 MHz to 3 GHz. The n78 band was chosen

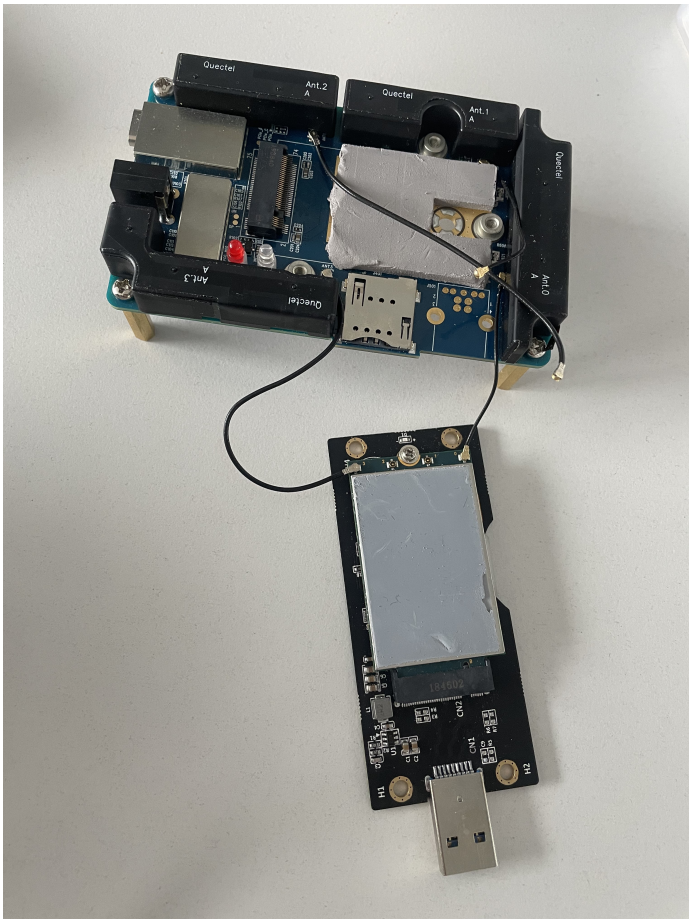

Fig. 4: Module and adapter setup

to make use of the NR regional license band, which is in the range of 3300 to 3800 MHz. Unfortunately, our antenna might not work well enough in this range for the COTS UE to detect the signal. This was discovered after the experiments were carried out. However, the 5G module did detect signals, but somewhat unstable. n2 (1930 MHz) FDD was also tried out but was also not detected.

2. This leads us to believe that there are clock synchronization issues. This is a known troubleshooting point mentioned in the well-written srsRAN project documentation [17]. Thus, our USRP probably needs an external hardware clock like a Leo Bodnar GPS Reference Clock to work with UE. It is worth noting that another set of antennas was used without any luck, which would also strengthen the theory of needing an external clock.

IV. Experimental Results

See table II for the created test cases and their results. The test cases were designed to test defined and undefined security algorithms as well as null integrity (as it should never accept it). No unexpected behavior happened and the implementation of the Queltec 5G module seemed to be fine. NIA1 and NEA0 were applied after modifying the messages, while the ”NAS security algorithms” fields were modified. The SECURITY

Integrity field	Ciphering field	MAC	Response
0x0	0x0	Unaltered	Reject
0x1	0x0	Unaltered	Complete
0x2	0x0	Unaltered	Reject
0x3	0x0	Unaltered	Reject
0x4	0x0	Unaltered	Reject
0x5	0x0	Unaltered	Reject
0x6	0x0	Unaltered	Reject
0x7	0x0	Unaltered	Reject
0x8	0x0	Unaltered	5GMM status
0x0	0x0	0x0000000	Reject
0x1	0x0	0x0000000	Reject
0x2	0x0	0x0000000	Reject
0x3	0x0	0x0000000	Reject
0x4	0x0	0x0000000	Reject
0x8	0x0	0x0000000	5GMM status
0x1	0x1	Unaltered	Reject
0x1	0x2	Unaltered	Reject
0x1	0x3	Unaltered	Reject
0x1	0x4	Unaltered	Reject
0x1	0x5	Unaltered	Reject
0x1	0x6	Unaltered	Reject
0x1	0x7	Unaltered	Reject
0x1	0x8	Unaltered	5GMM status
0x1	0xf	Unaltered	5GMM status
0xf	0x0	Unaltered	5GMM status

TABLE II: Test cases with experimental results

MODE REJECT contained the cause 24: unspecified. The UE responded with a 5GMM status message whenever an algorithm larger than 0x7 was used. This message is used to report certain error conditions detected upon receipt of 5GMM protocol data, and is a reasonable response to algorithms that should not be supported. The status message contained cause 96: invalid mandatory information. A pcap file for this test run can be found in PoC folder in the 5G-testbed repository on GitHub created for this project. [10]

V. Open5gs bug

In version v2.6.1 of Open5gs, a bug that caused denial of service of the NRF was found [14]. This bug was discovered when creating a refined setup after the PoC to facilitate more stable testing with a usable tool for other researchers. The idea was to create a custom network function to modify NAS messages. This would be modular, done in a more ”5G native” way, and be more easily ported to other 5G core implementations. The NF was written in Go. To make new NFs discoverable and usable by other NFs by the SBI, it should register at the NRF, as mentioned in section II. The NRF should support custom NF types as is mentioned in the 5G specification TS 29.510 V17.0.0 (section 5.2.2.2.2)[4]:

”The NRF shall allow the registration of a Network Function instance with any of the NF types described in clause 6.1.6.3.3, and it shall also allow registration of Network Function instances with custom NF types (e.g., NF type values not defined by 3GPP, or NF type values not defined by this API version).”

However, for Open5gs, this was not the case. When registering a NF with the the *nfType* set to *CUSTOM_INF* it fatally crashes the NRF:

```
curl -v -X PUT -d '{␣\
"nfInstanceId":"0b8a8d59-af80-4fb7-8645-
    b832fd69d94a",␣\
"nfType":"CUSTOM_INF",␣\
"nfStatus":"REGISTERED",␣\
"ipv4Addresses":["127.0.13.37"]}' \
--http2-prior-knowledge \
http://127.0.0.10:7777/nnrf-nfm/v1/nf-
    instances/0b8a8d59-af80-4fb7-8645-
    b832fd69d94a \
/
```

This bug was reported with a GitHub issue [11] and the maintainers fixed the bug promptly [9]. A CVE ID was also requested from Mitre, where CVE-2023-42367 has been reserved for the security bug.

Only nfTypes in defined by 3GPP are supported so far, all other types are rejected. The NRF should allow custom nfTypes as well. Thus not compiling to release 17, even though Open5gs claims to be compliant. This bug can have security consequences as most likely many 5G networks run some core based upon Open5gs. However, the nf-instances endpoint should be protected, so the attacker would need access to the SBI of the NRF somehow.

VI. Virtual machine-based testbed

Re-producible, easy to set up. Use Vagrant to create two virtual machines, one for 5GC and another for RAN/UE [10]. Go server to parse yaml file with test cases and modify intercepted messages from the modified core. The interceptor supports test cases with arbitrary hex strings modification of the messages, as well as intercepting any message type. This allows for powerful test cases. A custom SBI endpoint was created to support enabling/disabling intercepting messages in the core.

Fig 5 shows a simplified state-machine visualization of the flow of running test cases. Solid black lines are SBI interactions, the dotted black lines represent data flow via TCP sockets. Red solid lines are for state transitions and dotted red lines abstract away other states that might happen in between two states. When the Go interceptor starts, it makes an HTTP request to enable intercepting in Open5gs. In the imsi_state, the go interceptor sends the IMSI set by the test case file to open5gs, which saves it to know from which UE it should intercept. In the msg_type_state, the interceptor receives the outgoing message types from open5gs. If the received message type matches what the test case contains (otherwise looping state), then it replies indicating to intercept the plain message or the encrypted message. Going into msg_state, the interceptor receives the intercepted message, makes modifications and replies with the modified message, and transitions into res_state. In this state, the interceptor receives the message type of whatever message open5gs received after sending out the modified message. This response is saved in a Mongodb. When open5gs sends the response from recv_ngap, it disables intercepting. It has to be enabled again by the interceptor to allow more test cases. This way, Open5gs can operate in a ”normal” way as much as possible. After 5 seconds in this state, the interceptor will issue an SBI request to deregister the UE. If there are more test cases the interceptor trantitions into msg_type_state and enable interception again. If not, then the interceptor exits.

This setup could also be used with a hardware RAN, however, this has not been tested directly as this testbed was created after losing access to the hardware USRP. Just using the 5GC virtual machine and using the USRP as RAN, with COTS UE. Configurations for the USRP and the captured pcaps are given in the repository.

VII. Discussion

Troubleshooting radio is tedious, as the medium is inherently lossy and a lot of factors influence performance. Factors that were taken into consideration were: distance, TX/RX power, signal direction, frequency bands, physical cell id, tracking area code, clock synchronization, and different PLMN and imsi to force roaming. The use of a spectrum analyzer might have made this process a bit easier. Without it, troubleshooting was based on guesstimates. The time with hardware was limited, and with that time, the setup with the USRP was quite inconsistent. During the experimental results in section IV, manual intervention of restarting the UE had to be done.

However, a lot was learned about 5G along the way, especially the core network. The focus was initially UE implementation, but surprisingly a bug was found in the core implementation. During the later stages of the project, interaction with the Open5gs open-source community about the bug and missing features was done. Being an introduction to take part in further development of Open5gs.

This project motivates creating a consistent setup for other researchers who want to dive deeper into security in the 5G protocol. Time should be used for creating useful test cases, and doing actual verification by running tests consistently, not fighting with an inconsistent setup, and using a lot of time for troubleshooting.

The 5G specification is very complex, being put together of many long specification documents. All of the features may be overlooked and not implemented correctly in many parts of the system, not only the UE, as shown by the Open5gs bug. However, this might be harder to test, as telecom providers run their own custom software for the 5G core. These are black boxes in the network but are most likely based on the Open5gs.

VIII. Further Work

The main goal of what we initially wanted to achieve is still to be completed. A natural next step for further research is to take the refined testbed that has been created and connect a SDR RAN with an external clock to be able to connect and test COTS 5G-enabled devices.

The test cases could be extended to test the Identity Type field in the Identity Request message. This would test a clearly defined security requirement of the specification. As the UE shall not send its SUPI unencrypted.

To expand the testbed even further, more advanced fuzzing capabilities could be added. The Computer Security Group at Berlin University of Technology has created tooling for Man-in-the-Middle fuzzing of binary files[6]. A network protocol fuzzer, like fuzzowski [13], would also be highly relevant to use. This would be more rigorous testing of protocol fields, rather than manually crafting test cases.

Finally, work has to be done on missing features regarding custom network functions in Open5gs.

IX. Conclusion

References

[1] (x.x)eranga. "Deploying 5G Core Network with Open5GS and UERANSIM". In: (Nov. 2021). URL: https://medium.com/rahasak/5g-core-network-setup-with-open5gs-and-ueransim-cd0e77025fd7.

[2] 3GPP. *TS 23.501 V17.0.0; System architecture for the 5G System (5GS)*. 3GPP. 2021.

[3] 3GPP. *TS 24.501 V17.0.0; Non-Access-Stratum (NAS) protocol for 5G System (5GS)*. 3GPP. 2020.

[4] 3GPP. *TS 29.501 V17.0.0; 5G System; Principles and Guidelines for Services Definition*. 3GPP. 2020.

[5] aligungr. *UERANSIM*. https://github.com/aligungr/UERANSIM. 2023.

[6] fgsect. *FitM, the Fuzzer in the Middle*. https://github.com/fgsect/FitM. 2023.

[7] fgsect. *SCAT: Signaling Collection and Analysis Tool*. https://github.com/fgsect/scat. 2022.

[8] IDS. "Global Smartphone Shipments Expected to Decline 1.1% in 2023 as Recovery Is Pushed Forward into 2024 Amidst Weak Demand, According to IDC Tracker". In: (). URL: https://www.idc.com/getdoc.jsp?containerId=prUS50441423.

[9] Sukchan Lee. *Open5gs [NRF] Fixed NRF crash when Custom nfType*. https://github.com/open5gs/open5gs/commit/2aa12449aade5f50ed4710d9ac2eb8e1b96c43b9. 2023.

[10] Theodor Signebøen Midtlien. *5G-testbed*. https://github.com/theodorsm/5G-testbed. 2023.

[11] Theodor Signebøen Midtlien. *Open5gs issue 2576*. https://github.com/open5gs/open5gs/issues/2576. 2023.

[12] Theodor Signebøen Midtlien. *Open5gs testcase fork*. https://github.com/theodorsm/open5gs/tree/testcases. 2023.

[13] nccgroup. *Fuzzowski*. https://github.com/nccgroup/fuzzowski. 2023.

[14] Open5gs. *Open5gs*. https://github.com/open5gs/open5gs. 2023.

[15] Ettus Reseach. *USRP Hardware Driver (UHD™) Software*. https://github.com/EttusResearch/uhd. 2023.

[16] David Rupprecht, Kai Jansen, and Christina Pöpper. "Putting LTE Security Functions to the Test: A Framework to Evaluate Implementation Correctness". In: *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, Aug. 2016. URL: https://www.usenix.org/conference/woot16/workshop-program/presentation/rupprecht.

[17] srsran. *srsRAN gNB with COTS UEs*. https://docs.srsran.com/projects/project/en/latest/tutorials/source/cotsUE/source/index.html. 2023.

[18] srsran. *srsRAN Project*. https://github.com/srsran/srsRAN_Project. 2023.

[19] Chuan Yu et al. "SecChecker: Inspecting the Security Implementation of 5G Commercial Off-The-Shelf (COTS) Mobile Devices". In: *Computers & Security* 132 (Sept. 2023), p. 103361. ISSN: 01674048. DOI: 10.1016/j.cose.2023.103361.

APPENDIX

A. State machine

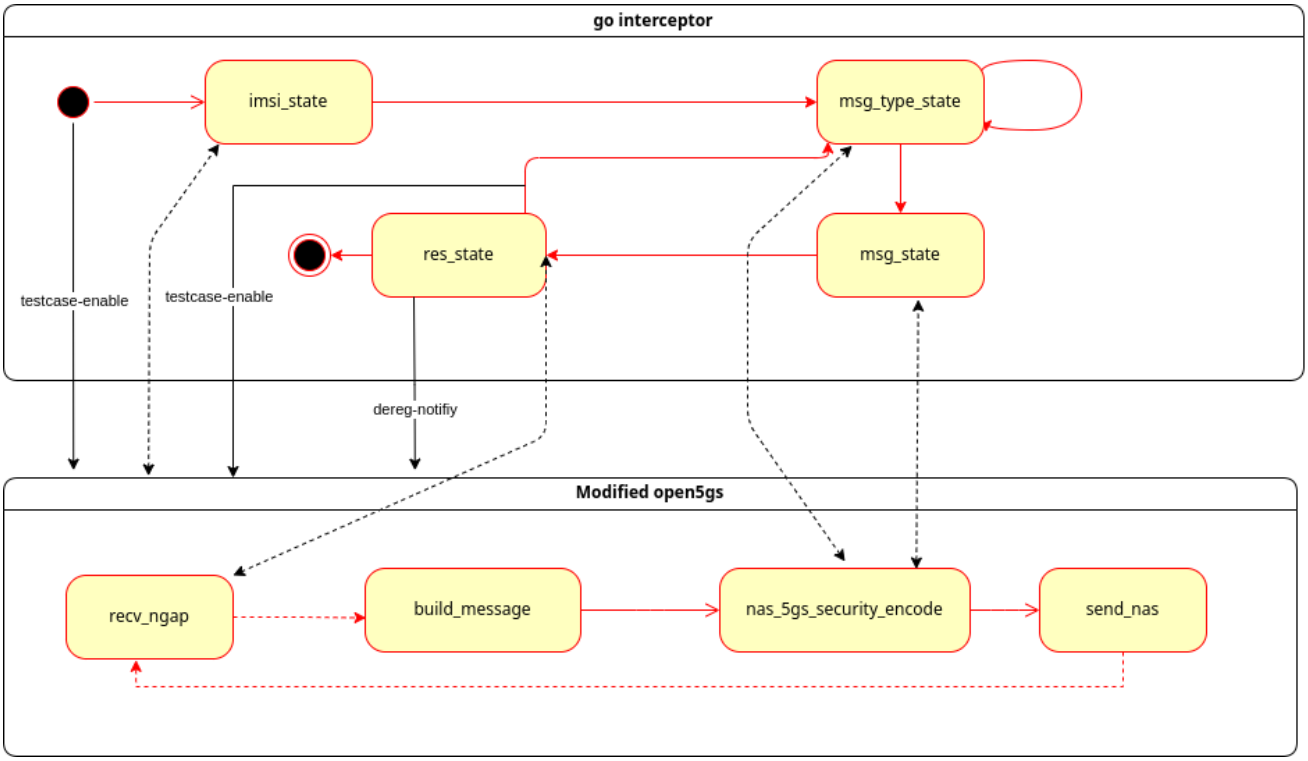

Fig. 5: Basic state machine visualization of the go interceptor and its interactions with the modified Open5gs version