5G Security Assurance: Lessons Learned from 3GPP SCAS Tests on Virtualized Network Architectures

Francesco Mancini[1,2,*], Lorenzo Cannella[3], Giorgia Messina[3], Marco Centenaro[5], Ivan Di Pietro[5], Alessandro Greco[5], Samuela Persia[5], Francesca Cuomo[3,4] and Giuseppe Bianchi[1,2]

[1]*CNIT NAM Lab, Rome, IT*
[2]*Univeristy of Rome "Tor Vergata", Rome, IT*
[3]*Sapienza University of Rome, Rome, IT*
[4]*National Inter-University Consortium for Telecommunications - CNIT, Rome, IT*
[5]*Italian National Cybersecurity Agency (ACN), Rome, IT*

Abstract

The increasing need for security assurance in 5G mobile networks is a priority at both national and European levels. However, implementing SCAS (Security Assurance Specifications) tests in the 5G Core (5GC) presents significant challenges, particularly when it comes to verifying the temporal sequence of events. Based on our own experience, we outline the key requirements for accurately implementing SCAS tests in such complex environments. We propose a general approach for executing these tests, emphasizing the critical issue of correctly verifying the sequence of events amidst concurrent and interdependent operations. Our approach is validated by testing three open-source 5GC implementations, highlighting the limitations of subjective interpretation of results.

Keywords

5G, Security Assurance, Core Networks

1. Introduction

The rapid growth of mobile applications and services across various sectors has driven the development of advanced communication systems capable of handling vast amounts of data generated by millions of devices. This evolution necessitates robust regulations, best practices, and methodologies to ensure both high network performance and strong security. At the national level, Italy's National Cybersecurity Agency (ACN) is defining procedures for 5G security evaluations, focusing on two key aspects: verifying the implementation of network security features and conducting vulnerability assessments and penetration tests. The National Assessment and Certification Centre (CVCN), part of ACN, is tasked with creating a methodology for assessing 5G components critical to national security, as identified by the DPCM of 15 June 2021 [1]. This aligns with European regulations requiring Member States to enhance the security and resilience of future mobile networks through comprehensive evaluations of network functionality and cybersecurity.

However, in practical terms, conducting SCAS-defined security tests by third-party specialists presents significant challenges. These tests often demand interaction with network functions that go well beyond simple "black-box" testing. Addressing these challenges, our research focuses on bridging the gap between SCAS specifications and their effective implementation in testing. We provide insights and define key requirements to enable testers to deploy tests that accurately reflect SCAS specifications. Our analysis identifies critical aspects of the implementation process, including handling the temporal sequencing of events – a non-trivial aspect of such tests. To validate our approach, we implemented a

Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), February 03-8, 2025, Bologna, IT
*Corresponding author.
francesco.mancini@cnit.it (F. Mancini); cannella.2046628@studenti.uniroma1.it (L. Cannella); messina.2059332@studenti.uniroma1.it (G. Messina); m.centenaro@acn.gov.it (M. Centenaro); i.dipietro@acn.gov.it (I. Di Pietro); a.greco@acn.gov.it (A. Greco); s.persia@acn.gov.it (S. Persia); francesca.cuomo@uniroma1.it (F. Cuomo); giuseppe.bianchi@uniroma2.it (G. Bianchi)
0000-0002-1924-8706 (F. Mancini); 0000-0002-9122-7993 (F. Cuomo); 0000-0001-7277-7423 (G. Bianchi)

series of representative tests using the ScasDK toolkit [2], currently being developed at CNIT NAM LAB/University of Rome "Tor Vergata". Through these tests, we analyzed the reactions of three open-source 5G Core (5GC) implementations, showcasing the practical applicability of our methodology while highlighting variances in network behavior not fully addressed by the SCAS specifications.

The paper is structured as follows. Sec. 2 provides the background on the European regulatory framework, 5G network architecture, and related work. Sec. 3 delves into the 3GPP SCAS methodology, emphasizing the challenges of implementing the specified tests and identifying key requirements. Sec. 4 details the test implementation process, while Sec. 5 presents the proof-of-concept evaluation and the results of the implemented tests. Finally, Sec. 6 concludes the paper by summarizing the key findings and contributions of this work.

2. Background and Related Work

2.1. 5G European Security Regulation

The security of 5G networks is a priority for the European Commission and central to its Security Union Strategy, given their role as critical infrastructure for essential services and societal functions. To address 5G cybersecurity risks, the Member States' authorities participating in the NIS Cooperation Group, supported by the European Commission and ENISA, published the "EU Toolbox of Risk Mitigating Measures" [3] in 2020. This framework outlines strategic and technical measures to secure 5G network rollout based on risks identified in the "EU coordinated risk assessment of the cybersecurity of 5G networks" [4]. Strategic measures emphasize regulatory oversight, enabling Member States to enforce strict security requirements on mobile network operators and diversify the supply chain to avoid single supplier dependencies. Technical measures focus on securing 5G infrastructure through protocols for network architecture, access controls, and Network Function (NF) integrity. The second progress report (2023) outlines the advancements of 24 Member States in implementing these measures, including legislative updates by national authorities [5]. Among of all technical measures indicated in the toolbox, the specific technical measures TM09 "*Using EU certification for 5G network components*" and TM10 "*Using EU certification for other non 5G-specific ICT products and services*" are related to the technical security aspects of all network components (native 5G or not) that are needed for the whole functionality of 5G networks. Currently, ENISA is finalizing the European 5G certification scheme. The final scheme [6] will be composed of three main parts:

- *GSMA-NESAS* (*Network Equipment Security Assurance Scheme*) framework, proposed by the *Global System for Mobile Communications* (GSMA), includes two key components: i) the SCAS (Security Assurance Specifications) framework, which establishes a common, testable baseline of security properties for various 5G products; and ii) a life-cycle framework aimed at enhancing security levels across the entire industry.
- *GSMA SAS SM/SAS UP (Security Accreditation Scheme Subscription Management/ Security Accreditation Scheme for UICC Production)* in which the life-cycle and SIM provisioning framework of GSMA will be taken into account to define the European scheme.
- *eUICC* in which the Common Criteria framework will be considered to define the European scheme for embedded SIM and embedded UICC. It will be based on the EUCC scheme, the European Certification scheme based on Common Criteria available at the beginning of 2025.

For this reason, the possible implementations of SCAS tests represent a fundamental aspect to define a national methodology for security evaluations of 5G, in accordance with the European indication to improve the security and resilience in the deployed 5G networks, in order to comply with the specific measures laid downin 5G Toolbox.

2.2. 5G Architecture Focus

The 3GPP technical specification (TS) 23.501 [7] describes the network architecture of a 5G system (5GS) in detail - see Fig. 1. The two network segments, that are, the RAN and the 5GC, exchange

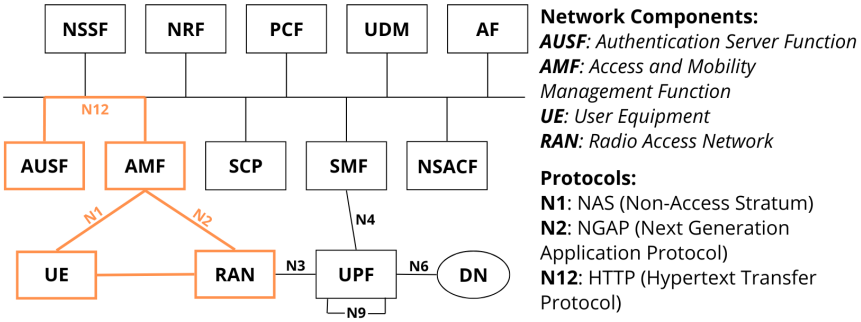


Figure 1: Architecture of a 5GS (courtesy of [7]). Specifically, the NFs and reference points highlighted will be the focus of our work.

signaling (control plane) traffic and data (user plane) via the N2 and N3 reference points, respectively. Moreover, the user equipment (UE) directly exchanges control-plane traffic with the 5GC exploiting the N1 reference point.

In this paper, we will focus on the 5GC NF which terminates the N1 and N2 reference points, that is, the AMF. As the entry point of the entire service-based architecture (SBA) of the 5GC control plane, the AMF plays a crucial role in the implementation of various functionalities such as, e.g., the management of the UE registration to the network (cf. Clause 5.3.2, [7]) and the management of UE connection (cf. Clause 5.3.4, [7]). The AMF is also fundamental in ensuring the confidentiality and integrity of signaling traffic and the subscriber privacy (cf. Clause 5.5, [8]), as well as, most importantly, in the authentication and key agreement procedure between the UE and the network (5G-AKA) (cf. Clause 6.1, [8]), exploiting its own security anchor function (SEAF) capabilities and communicating with the authentication server function (AUSF) via the N12 reference point. Based on this, guaranteeing the correct implementation and the cybersecurity of the AMF is of utmost importance to ensure the cybersecurity of the whole 5GS. Indeed, the AMF-specific SCAS, that is, the TS 33.512 [9], includes tests related to both the functional security of the NF and of cybersecurity of the NF, as far as its particular threats are concerned.

2.3. Related Work

The security assurance of 5G networks necessitates a comprehensive approach that encompasses various aspects of security testing, as emphasized by ENISA [10]. Research in this area emphasizes complementary methodologies to SCAS tests, including fuzzing, replay analysis, and adversarial testing, which collectively contribute to identifying and mitigating vulnerabilities in 5GS. Intelligent fuzzing frameworks have been extensively studied, particularly for the NAS protocol, enabling the evaluation of both User Equipment (UE) and 5GC resilience through advanced algorithms and test strategies [11, 12, 13]. Tools like 5Greplay facilitate replay analysis, enabling the simulation of attack scenarios to assess network robustness against malicious actions [14]. These methodologies address the novel challenges introduced by 5G's expanded feature set and diverse use cases while highlighting the emerging threats that require rigorous testing. Morover, adversarial testing has revealed the susceptibility of machine learning-enhanced 5G network components to carefully crafted inputs, demonstrating the critical need for robust validation mechanisms [15]. Complementing these methods, model-based testing (MBT) has gained traction in other domains for its ability to formalize test specifications and automate test generation. Model-based security testing (MBST), in particular, can derive executable test cases from system models, improving clarity and reducing ambiguities [16]. However, traditional MBST approaches often focus on single-interface systems and sequential interactions. In contrast, our approach extends the application of MBT by considering complex scenarios characteristic of 5G networks, which involve multiple system interfaces and diverse actions. By integrating both functional and non-functional testing dimensions, our methodology aligns with the intricate requirements of 5G environments, bridging gaps in existing practices and enhancing the efficacy of security assurance strategies.

3. 3GPP SCAS Tests Requirements

3.1. How can a tester implement a SCAS test?

Let us consider the perspective of a tester tasked with implementing a set of SCAS tests to certify the implementation of an AMF. As discussed in Sec. 2.2, the AMF is among the most complex NFs in the 5GC[1]. The process begins with the tester reviewing the relevant Technical Specifications. These documents define a suite of tests designed to verify that the 5G-specific security requirements are correctly implemented in the NF under test. For each security feature, the specification includes one or more tests that evaluate both the correctness of its basic functionality and its resilience in handling corner cases. Each test is structured into three main sections.

- Pre-conditions, i.e., the necessary conditions that must be met before executing the test.
- Execution Steps, i.e., a sequence of actions the tester must perform to carry out the test.
- Expected Results, i.e., the outcomes that the tester needs to observe and document to validate the test.

3.1.1. Pre-conditions

The tester begins by examining the *Pre-conditions* section, which outlines the configuration requirements for the testing network and the components supporting the NF under test. These often include other NFs, base stations (gNBs), and UEs, whether real or emulated. At first glance, the tester might assume that deploying standard network components would suffice to support test execution. However, the complexity of the 5GC architecture becomes quickly apparent. The distributed SBA in the control plane introduces the need for precise configuration and coordination of multiple network elements, even in a test environment.

3.1.2. Execution Steps

The tester reviews the *Execution Steps*, which describe actions and events in natural language. These often require supporting components to enforce anomalous behaviors, such as altering messages or managing precise timings. The complexity grows due to multiple communication protocols (e.g., NAS, NGAP, HTTP) and requirements like message delays or timeouts. This calls for programmable, adaptable components capable of handling heterogeneous protocols and synchronizing actions. The tester delves into the test steps, recognizing the need for specialized expertise to implement them effectively. In some test scenarios, a single operation may translate into a series of intricate sub-steps. For example, in the *TC_Sync_Fail_Seaf* test, the tester must handle sequence number resynchronization, ensuring fresh messages are used to prevent replay attacks. Constructing a synchronization failure message involves extracting data from the authentication procedure, which is not explicitly stated in the test description. In other cases, the necessary actions are implicit, requiring the tester to deduce them. For example, in case B of *TC_Res_Star_Verification_Failure* test, the tester must ensure the UE completes a registration and deregistration procedure to obtain a valid 5G-GUTI before initiating the test. These preparatory actions, essential for the test, are not explicitly detailed in the test specification, increasing the implementation effort for the tester.

3.1.3. Expected Results

The last operational section of a test case consists of the *Expected Results*. This section specifies the results the tester must provide to prove the success of the test. It typically defines three possible types of results:

- a screenshot of the execution;

[1]While this example focuses on the AMF, the challenges and motivations are similarly relevant to SCAS tests for other NFs.

- the log files of the component under test;
- capture of the test network traffic in a standard format (i.e., a pcap file).

The first type of format, while visually intuitive, lacks standardization, making it challenging to use as definitive proof of a test's success. Additionally, it is unsuitable for automated result evaluation systems. Log files, on the other hand, offer a format that is potentially machine-readable. However, their structure varies significantly depending on the implementation. Some logs may be structured (e.g., JSON), others unstructured, and in certain cases, logs might not be stored as files but instead managed through centralized logging systems. Even when accessible, logs may not contain all the necessary information to validate the success of a test. Furthermore, testers may not always have permission to retrieve the required data from the logs. Network traffic generated during the test could serve as a more reliable proof of success, as it captures the interactions directly. However, many communications in the 5GC are encrypted (e.g., HTTP traffic in the SBA is often secured with TLS). In such cases, testers must decrypt the traffic before saving it in a pcap file for analysis.

3.2. Requirements for test implementation

Based on the lessons learned from implementing SCAS tests, we identified key features commonly required by most test cases, as far as our analysis indicates. These requirements are grouped into categories, each one addressing a specific aspect of test implementation. The main requirements are as follows.

- Actors Involved, i.e., the number of active participants required during the test.
- Action Types, i.e., the types of actions the tester must perform during the test.
- Verifications, i.e., the verification processes needed to validate test outcomes and how they should be conducted.

A test may require a single actor that actively interacts with the NF under test. This case also include scenarios in which the component under test interacts with other NFs only for supporting its operations (e.g., while testing UDM, it requires to access to UDR to retrieve subscribers information but these component does not actively participate to the test). If a test requires multiple actors (e.g., an UE and an AUSF for testing the authentication procedure in the AMF), the tester has to keep in mind that has to implement a dedicated mechanism to coordinate such components. The actions performed during a test can be categorized as functional or non-functional. Functional actions involve operations on the content of communication, such as modifying, replaying, or dropping messages. Non-functional actions, on the other hand, address the quality of communication, including introducing delays or triggering timeouts in the NF under test. Many tests require a combination of both functional and non-functional actions. Verification requirements involve two main aspects: determining whether specific events occur (positive verification) or do not occur (negative verification), and assessing the relationships between

Table 1
SCAS Test Requirements

Requirement	Description	Possible Values
Actors Involved	Specifies the number of active participants required for the test.	*Single*, *Multiple*
Action Types	Defines the nature of actions required during the test.	*Functional*, *Non-Functional*
Verifications	Details the focus and method of test validation.	**Focus:** *Presence of events*, *Absence of events* **Method:** *Individual events*, *Temporal sequence of events*

events, especially their sequence. While some tests only require event occurrence validation, others necessitate confirming the correct temporal order of events, particularly for those originating from different parts of the network, where proper sequencing cannot be assumed. Tab. 1 summarizes these requirements.

4. Test implementation

In this section, we present our approach to implementing SCAS tests and address how we overcame the challenges associated with meeting the aforementioned requirements. Specifically, we focus on ensuring the correctness of temporal event sequence verification. While this task is straightforward when the tester controls a single component, we will demonstrate in the following sections that in distributed scenarios, such as ours, this assurance becomes significantly more complex. To provide a clear framework for creating support tools to implement these tests, we have abstracted our approaches from the specifics of the actual implementation.

4.1. General Representation of SCAS Tests

A SCAS test can be broadly represented as a sequence of the following phases.

- **Configuration** – The tester sets up the network or configures the support components required for the test.
- **Action** – The tester initiates an action within the test network, such as starting a procedure or triggering an anomalous behavior.
- **Verification** – The tester verifies that specific events (or a subset of them) have occurred correctly, ensuring compliance with the test requirements.

These phases are not intended to follow a rigid, sequential waterfall model. Instead, they are designed to be executed iteratively, offering the flexibility to adapt based on the specific demands of each test. For instance, a tester may begin by configuring a test component, then trigger an action, verify that an expected event occurs, perform another action, and finally validate that all required events have been correctly executed. This iterative approach allows the phases to be mixed, repeated, or adjusted as necessary to ensure comprehensive test coverage and accuracy.

4.2. Test implementation architecture

All of the SCAS tests discussed in this paper were implemented and applied to virtualized 5GC networks using an enhanced version of the ScasDK framework originally introduced in [2]. Indeed, one of our side goals was to understand how third parties, not directly involved in the platform's ongoing development and enhancement, would acquaint with the process of programming and running tests on top of the platform.

ScasDK architecture is inspired by the conceptual architecture widely adopted in software defined networking, and entailing the separation between a distributed data plane and a centralized control plane. In details:

- the **Data Plane Layer** consists of the test actuators, encompassing all the entities actively participating in the test. In our context, these include UEs, gNBs, and NFs;
- the **Control Plane Layer** implements the test logic, managing interactions with the data plane for configuring components, triggering test behaviors, capturing events, and analyzing test outcomes.

The control plane layer in our approach incorporates features designed to meet the outlined SCAS test requirements. For the implementation of the test, i.e., their software programming, ScasDK exposes an interface based on the Robot Framework, a widely-used open-source test automation tool suitable for industrial applications. Robot Framework offers testers a user-centric environment, enabling them to write tests in a comprehensible manner, generate detailed logs, and identify root causes of failed tests. We enhanced the framework with specialized libraries for 5G SCAS testing, categorized as follows:

- **Configuration Libraries**, i.e., tools for setting up test components (e.g., UEs, gNBs, NFs);
- **Action Libraries**, i.e., tools for initiating actions in the network under test;
- **Event Libraries**, i.e., tools for defining, capturing, and analyzing test events.

Further descriptions of ScasDK and the test controller can be found in Annex A and B.

4.3. Temporal Event Verification in SCAS Testing

One of the critical requirements in SCAS testing is verifying that specific events occur in a defined temporal sequence. Often, this requirement is inherently satisfied by standard-defined procedures. For instance, the 5G-AKA protocol employs a request-response model that inherently linearizes the sequence of events. In such cases, testers primarily validate event occurrence rather than explicitly confirming temporal order, as the protocol itself guarantees correct sequencing. However, deeper verification becomes crucial in scenarios where distributed NFs generate events concurrently. For example, in case B of *TC_Sync_Fail_Seaf* test, the test verifies whether the AMF correctly handles resynchronization under abnormal conditions. In this scenario, the AUSF intentionally delays its response to the AMF to induce a timeout. The AMF must wait for the timeout to expire before restarting a new authentication procedure. Here, testers must validate the event sequence by analyzing timestamps to ensure the timeout precedes the subsequent authentication request. A tester can apply the following two approaches to timestamp attribution.

- **Component-Generated Timestamps** – Each component in the network assigns its own timestamps. However, this approach becomes infeasible when testers lack control over the components, such as the AMF under test. Even if logs are accessible, clock drift between hosts can create inaccuracies, potentially misrepresenting the actual order of events. Additionally, there may be rare cases where the timestamps of two events are identical, complicating sequence verification.
- **Tester-Generated Timestamps** – The tester assigns timestamps upon event detection. This approach eliminates discrepancies caused by clock drift since a single, unified clock source is used. However, network delays can distort the tester's perspective, leading to potential misinterpretation of the event sequence. For instance, an event transmitted before a timeout might appear to occur after it due to delays in communication.

To address these challenges, we enhanced ScasDK to include a mechanism capable of verifying the temporal sequence of events using Tester-Generated Timestamps within the proxy-based architecture. This architecture intercepts and relays traffic to the test controller while temporarily holding the flow until event processing is complete, ensuring that event sequences are linearized under controlled conditions. In cases involving parallel events where network delays could impact sequence accuracy, we implement synchronization techniques combined with protocol-level logic to deduce relationships between events.

In scenarios involving concurrent events where network delays might distort the perceived event order, we utilize synchronization techniques augmented with protocol-level insights to interpret event relationships. For instance, in the case B of *TC_Sync_Fail_Seaf* test, delays could cause the tester to observe events in reverse order, making it seem as though the AMF waited for the timeout before sending a new authentication request. However, the AMF should include a fresh challenge in the new request, which can only be obtained from the AUSF after the timeout or upon receiving the new challenge. If the AMF sends the new authentication request prematurely without waiting for the AUSF, it will reuse the same challenge. This behavior violates protocol requirements and indicates that the new request was sent before the timeout expired. While our method improves the reliability of event sequence verification, scenarios with highly concurrent events still present challenges. Figure 2 illustrates how our approach addresses the limitations of basic methods, enhancing accuracy in event order validation.

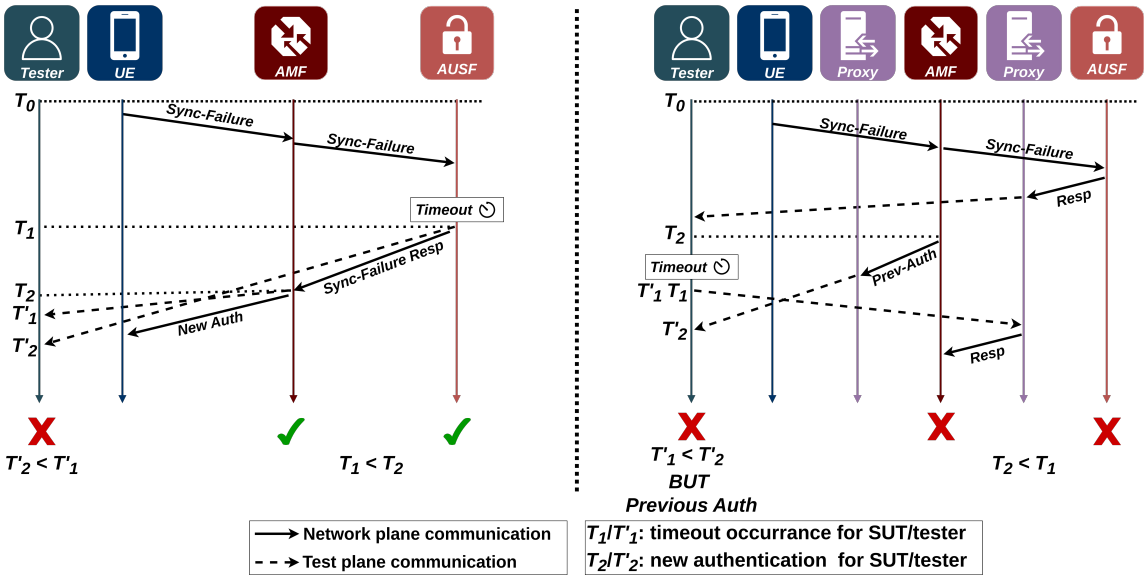


Figure 2: Basic timestamp-based verification vs enhanced protocol-based version

5. Proof of concept and results

We validated our approach by implementing a subset of SCAS tests for the AMF component. Table 2 demonstrates how each requirement is addressed by at least one test, highlighting the effectiveness of our analysis in identifying meaningful requirements. We applied these tests to three open-source 5GC implementations: free5GC v3.3.0, Open5GS v2.7.0, and OpenAirInterface (OAI) v2.1.0. To ensure the reliability of our results, we compared test outcomes with the actual communication flows captured during execution. For our case study, we focused on the case B of *TC_Sync_Fail_Seaf* test, as it incorporates the most requirements. For each implementation, we executed the test and captured network traffic at the AMF to serve as a ground truth. The following analysis underscores the challenges of interpreting results, especially in light of ambiguities introduced by test specification limitations. Additional details, including test traffic and implementation notes, are provided in the annex C.

5.1. TC_Synch_Fail_Seaf - Case B

We present the analysis of the outcomes, while in the annex C.1 we present the detailed info of the traffic extracted. Our findings validate our approach and implementation, as all test outcomes align with the observed traffic flows. However, they also expose limitations in the SCAS test specifications, which lack clear guidance for addressing ambiguities and interpreting results in cases of communication flow deviations or unforeseen network behaviors. This underscores the need to enhance test specifications to minimize subjective interpretations and promote consistent evaluation across different implementations.

In **free5GC** implementation the test passed, with communication behavior precisely matching the test specifications. The UE sent a *sync_failure* message, which the AMF forwarded to the AUSF. The AUSF delayed its response, as specified, triggering the timeout. Upon receiving the delayed response, the AMF immediately initiated a new authentication request to the UE. This behavior satisfied all test requirements and validated the approach.

Also in **Open5GS** implementation the test passed, but with deviations in the communication flow. The AMF sent a Registration Reject message before the timeout, citing an internal error. After a delay, the UE restarted the authentication procedure and completed it successfully. Despite these deviations, the AMF initiated a new authentication request after the timeout, as required by the test specification. However, the unexpected Registration Reject message introduces ambiguity, which is not clarified by the test specification. This raises the question: *Does the test remain valid if the communication flow deviates from the expected sequence?*

	Actors		Actions		Verifications	
	Single	Multiple	Functional	Nonfunctional	Occurrence	Temporal
AMF tests						
TC_SYNC_FAIL_SEAF_AMF – Case A	–	•	•	–	–	•
TC_SYNC_FAIL_SEAF_AMF – Case B	–	•	•	•	–	•
TC_NAS_NULL_INT_AMF - Case A	–	•	•	–	•	–
TC_NAS_NULL_INT_AMF - Case B	–	•	•	–	•	–
TC_NAS_INT_SELECTION_USE_AMF	•	–	•	–	•	–
TC_UE_SEC_CAP_HANDLING_AMF - Case 1	•	–	•	–	•	–
TC_UE_SEC_CAP_HANDLING_AMF - Case 2	•	–	•	–	•	–
TC_UE_SEC_CAP_HANDLING_AMF - Case 3	•	–	•	–	•	–
TC_UE_SEC_CAP_HANDLING_AMF - Case 4	•	–	•	–	•	–

Table 2
Requirements of each implemented test

In contrast, in **OAI** the test failed due to behavior that contradicted the specifications. The AMF sent an Authentication Failure message to the UE, attributing the failure to the UE itself. Consequently, the UE did not initiate a new authentication procedure, leading to disconnection from the network. This prevented the UE from accessing services despite being authorized. These deviations violate test requirements and highlight significant issues in the AMF's implementation. Furthermore, the ambiguity in interpreting whether the observed behavior satisfies the test objectives stems from insufficient guidance in the test specification.

6. Conclusion

This paper examined the application of 3GPP SCAS tests to virtualized 5G network architectures, emphasizing test analysis, implementation complexity, and temporal accuracy. Using a centralized clock and protocol knowledge, our framework enhances event sequencing and validation in distributed systems like the 5GC. Validation against free5GC, Open5GS, and OAI revealed significant behavioral variations not addressed by SCAS specifications, highlighting gaps that reduce consistency and requires subjective interpretation. Future work will focus on automating test case generation through machine learning, improving scalability and effectiveness in 5G security assurance.

Acknowledgments

This work was partially supported by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP F83C22001690001, E83C22004640001 and B53C22004050001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART"), and CUP F83C22001670001 and B53C22003990006, partnership on "SEcurity and RIghts In the CyberSpace" (PE00000014 - program "SERICS").

Declaration on Generative AI

During the preparation of this work, the authors used GPT-3.5 in order to: grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] Italian Prime Minister Decree of 15 June of 2021, DPCM 3 Perimetro, [online], 2021. URL: https://www.gazzettaufficiale.it/atto/vediMenuHTML?atto.dataPubblicazioneGazzetta=2021-08-19&atto.codiceRedazionale=21A05087&tipoSerie=serie_generale&tipoVigenza=originario.

[2] F. Mancini, G. Bianchi, ScasDK - a development kit for security assurance test in multi-network-function 5G, in: Proceedings of the 18th International Conference on Availability, Reliability and Security, ARES '23, Association for Computing Machinery, New York, NY, USA, 2023. URL: https://doi.org/10.1145/3600160.3605044. doi:10.1145/3600160.3605044.

[3] Cybersecurity of 5G networks EU Toolbox of risk mitigating measures, Technical Report, NIS Cooperation Group, 2020. URL: https://digital-strategy.ec.europa.eu/en/library/cybersecurity-5g-networks-eu-toolbox-risk-mitigating-measures.

[4] EU coordinated risk assessment of the cybersecurity of 5G networks, Technical Report, NIS Cooperation Group, 2019. URL: https://digital-strategy.ec.europa.eu/en/news/eu-wide-coordinated-risk-assessment-5g-networks-security.

[5] Second report on Member States' Progress in implementing the EU Toolbox on 5G Cybersecurity, Technical Report, NIS Cooperation Group, 2023. URL: https://digital-strategy.ec.europa.eu/en/library/second-report-member-states-progress-implementing-eu-toolbox-5g-cybersecurity.

[6] ENISA EU5G Team, EU 5G Scheme Development, [online], 2023. URL: https://www.3gpp.org/ftp/TSG_SA/TSG_SA/TSGS_99_Rotterdam_2023-03/INBOX/DRAFTS/20230322-EU5G-3GPPSA-final.pptx.

[7] System architecture for the 5G System (5GS), TS 23.501, 3GPP, 2024. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144, Rel. 19.

[8] Security architecture and procedures for 5G System, TS 33.501, 3GPP, 2024. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169, Rel. 19.

[9] 5G Security Assurance Specification (SCAS); Access and Mobility management Function (AMF), TS 33.512, 3GPP, 2024. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3445, Rel. 18.

[10] Security in 5G Specifications - Controls in 3GPP, Technical Report, European Union Agency for Cybersecurity (ENISA), 2021. URL: https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Report%20-%20Security%20in%205G%20Specifications.pdf.

[11] S. Khandker, M. Guerra, E. Bitsikas, R. P. Jover, A. Ranganathan, C. Pöpper, Astra-5g: Automated over-the-air security testing and research architecture for 5g sa devices, in: Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 89–100. URL: https://doi.org/10.1145/3643833.3656141. doi:10.1145/3643833.3656141.

[12] E. Kim, D. Kim, C. Park, I. Yun, Y. Kim, BaseSpec: Comparative analysis of baseband software and cellular specifications for l3 protocols (2021). URL: https://dx.doi.org/10.14722/ndss.2021.24365.

[13] F. Mancini, S. Da Canal, G. Bianchi, Amfuzz: Black-box fuzzing of 5g core networks, in: 2024 19th Wireless On-Demand Network Systems and Services Conference (WONS), 2024, pp. 17–24. doi:10.23919/WONS60642.2024.10449510.

[14] Z. Salazar, H. N. Nguyen, W. Mallouli, A. R. Cavalli, E. Montes de Oca, 5Greplay: a 5G Network Traffic Fuzzer - Application to Attack Injection, in: Proceedings of the 16th International Conference on Availability, Reliability and Security, ARES '21, Association for Computing Machinery, New York, NY, USA, 2021. URL: https://doi.org/10.1145/3465481.3470079. doi:10.1145/3465481.3470079.

[15] G. Apruzzese, R. Vladimirov, A. Tastemirova, P. Laskov, Wild Networks: Exposure of 5G Network Infrastructures to Adversarial Examples, IEEE Transactions on Network and Service Management 19 (2022) 5312–5332. doi:10.1109/TNSM.2022.3188930.

[16] I. Schieferdecker, J. Grossmann, M. A. Schneider, Model-based security testing, in: A. K. Petrenko, H. Schlingloff (Eds.), Proceedings 7th Workshop on Model-Based Testing, MBT 2012, Tallinn, Estonia, 25 March 2012, volume 80 of *EPTCS*, 2012, pp. 1–12. URL: https://doi.org/10.4204/EPTCS.80.1. doi:10.4204/EPTCS.80.1.

A. ScasDK Architecture

The ScasDK framework provides a practical solution for testing the security of 5G networks by using proxies to monitor and modify network traffic between different components. These proxies act as intermediaries, intercepting communications between network functions without altering their original software. They can be dynamically configured to adapt to specific test scenarios, enabling detailed examination and manipulation of protocols unique to 5G. A centralized controller orchestrates the entire process, setting up proxies to perform tasks such as altering data, analyzing messages, or introducing delays. This flexible approach allows testers to simulate various conditions and behaviors in real-time, ensuring comprehensive security evaluations. When tests reveal issues, the system offers detailed feedback to help identify and address potential vulnerabilities in the network.

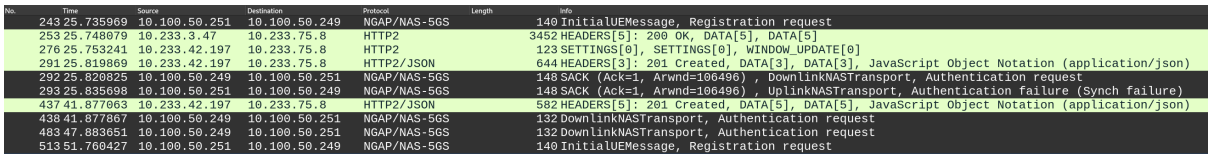
Proxies
Proxies in the ScasDK framework play a key role in managing and analyzing network traffic during tests. They intercept messages between 5G network components, forwarding them unchanged or modifying them by using user-defined hooks. This adaptability allows testers to record data, manipulate communications, and introduce delays. For instance, the NGAP/NAS proxy is tailored for interactions between gNB and AMF, while the HTTP proxy employs the open-source Envoy platform to facilitate control-plane communication among network functions. Both proxies can be dynamically configured via an interface, allowing external tools to adjust and analyze messages in real-time. This approach ensures seamless communication between network components, even when they come from different vendors or lack support for standard protocols.

Controller
The test controller is the central hub of the ScasDK testing framework, responsible for managing and coordinating the entire testing process. It sets up the environment by configuring key components like proxies and emulated devices with necessary parameters (e.g., network addresses or operator IDs). During a test, the controller ensures all elements work together seamlessly, reconfiguring them dynamically as needed. It can also generate network traffic directly or delegate this task to other components, depending on the complexity of the interactions required. By overseeing operations, collecting data, and evaluating results, the controller ensures that each test runs smoothly and provides meaningful insights into network behavior and security.

B. Test Controller Model

The Robot Framework is an open-source automation tool designed to simplify test creation and execution. Its keyword-driven approach, paired with extensibility through libraries written in Python or Java, makes it highly adaptable to specific testing needs, including those defined in SCAS tests. As part of our implementation, we developed the following specialized libraries:

- **Network Components Libraries**: These libraries represent supported test components, such as UEs, RANs, and NFs, within the Robot Framework. They provide functionality to configure these components and enable them to perform actions, including sending messages or initiating specific procedures.
- **Proxy Libraries**: Proxy libraries model instances of proxies deployed in the network. They allow testers to configure them and define hooks to be executed during testing.
- **Hook Libraries**: These libraries empower testers to design custom hooks to log, monitor, and modify network traffic. Users can define events triggered by these hooks, with the library automatically logging event occurrences and associating timestamps. Furthermore, the library provides functions to validate the occurrence of events and retrieve their timestamps in order to verify the temporal sequence.



No.	Time	Source	Destination	Protocol	Length	Info
243	25.735969	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	140	InitialUEMessage, Registration request
253	25.748079	10.233.3.47	10.233.75.8	HTTP2	3452	HEADERS[5]: 200 OK, DATA[5], DATA[5]
276	25.753241	10.233.42.197	10.233.75.8	HTTP2	123	SETTINGS[0], SETTINGS[0], WINDOW_UPDATE[0]
291	25.819869	10.233.42.197	10.233.75.8	HTTP2/JSON	644	HEADERS[3]: 201 Created, DATA[3], DATA[3], JavaScript Object Notation (application/json)
292	25.820825	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	148	SACK (Ack=1, Arwnd=106496) , DownlinkNASTransport, Authentication request
293	25.835698	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	148	SACK (Ack=1, Arwnd=106496) , UplinkNASTransport, Authentication failure (Synch failure)
437	41.877063	10.233.42.197	10.233.75.8	HTTP2/JSON	582	HEADERS[5]: 201 Created, DATA[5], DATA[5], JavaScript Object Notation (application/json)
438	41.877867	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	132	DownlinkNASTransport, Authentication request
483	47.883651	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	132	DownlinkNASTransport, Authentication request
513	51.760427	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	140	InitialUEMessage, Registration request

Figure 3: Test trace of "*TC_Sync_Fail_Seaf* - case B" in free5GC Core Network

C. SCAS Test Implementation and Analysis

In this section we present the description of two representative test SCAS for AMF we implemented, namely "Tc_Sync_Fail_Seaf - Case B" and "Tc_Ue_Sec_Cap_Handling" . We focus on implementation detail we applied in order to accomplish the test requirements.

C.1. Tc_Sync_Fail_Seaf - Case B

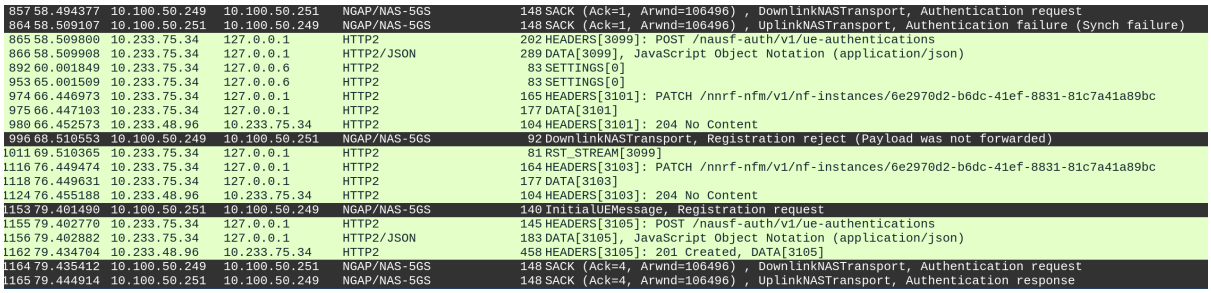
The first test implemented with ScasDK, called "Tc_Sync_Fail_Seaf- Case B", focused on simulating a synchronization failure and testing how the AMF handles specific timing and event sequences. This required addressing several challenges, such as creating a failure scenario, managing a timeout, and ensuring events happened in the correct order. The test involved three main components: the UE (user equipment), AMF, and AUSF. Communication between the UE and AMF relied on the NAS protocol, while the AMF and AUSF interacted using the HTTP protocol. To simulate the failure, a modification was made to the NAS messages, changing the authentication token and triggering a mismatch that caused the desired synchronization failure. To test the timeout, the communication between the AMF and AUSF was delayed for 15 seconds, according to T3520 timer definition, by intercepting and holding HTTP messages. A custom tool was used to manage this delay. Additionally, the ScasDK framework was updated to record timestamps for key events, ensuring that their timing could be accurately measured. This made it possible to confirm whether the AMF waited the correct amount of time before proceeding. Finally, the test checked that the AMF didn't start a new authentication process before the timeout expired and the delayed response was processed. If the AMF behaved correctly—only continuing the process after the timeout and response—the test was considered successful.

In the following we further analyze the reaction of the three Core Network implementations we tested.

free5GC. In the free5GC implementation, the observed network flow aligns with the expected test specifications, as shown in Figure 3. Specifically, the UE sends the synchronization failure message at 25.835698 seconds after the capture begins (packet number 293). Approximately 15 seconds later, we observe the delayed response from the SEAF (packet number 437). Immediately afterward, the AMF sends a new authentication request to the UE (packet number 438), completing the flow as specified in the test case.

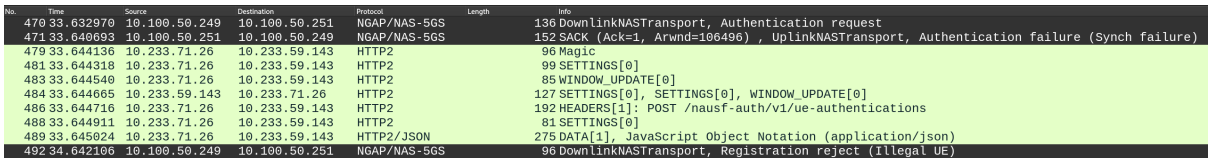
Open5GS. In Open5GS, the test also passes, albeit with a different network flow, as depicted in Figure 4. Here, the delayed synchronization failure request is sent at 58.509908 seconds after the capture starts (packet number 866). The AMF subsequently responds with an authentication request and a registration failure message approximately 10 seconds later (packet number 996). This behavior is attributed to an internal timeout in the AMF's HTTP client. However, the registration failure cause, specifically "Payload Not Forwarded", allows the UE to initiate a new registration request. Following this, the new authentication request is received, thereby satisfying the test requirements.

OAI. In the case of OAI, the test fails. The flow, illustrated in Figure 5, appears similar to that of Open5GS initially, with the AMF responding 1 second after the synchronization failure request due to its internal HTTP client timeout. However, the failure's root cause differs: the AMF responds with an



857	58.494377	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	148	SACK (Ack=1, Arwnd=106496) , DownlinkNASTransport, Authentication request
864	58.509107	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	148	SACK (Ack=1, Arwnd=106496) , UplinkNASTransport, Authentication failure (Synch failure)
865	58.509800	10.233.75.34	127.0.0.1	HTTP2	202	HEADERS[3099]: POST /nausf-auth/v1/ue-authentications
866	58.509908	10.233.75.34	127.0.0.1	HTTP2/JSON	289	DATA[3099], JavaScript Object Notation (application/json)
892	60.001849	10.233.75.34	127.0.0.6	HTTP2	83	SETTINGS[0]
953	65.001509	10.233.75.34	127.0.0.6	HTTP2	83	SETTINGS[0]
974	66.446973	10.233.75.34	127.0.0.1	HTTP2	165	HEADERS[3101]: PATCH /nnrf-nfm/v1/nf-instances/6e2970d2-b6dc-41ef-8831-81c7a41a89bc
975	66.447103	10.233.75.34	127.0.0.1	HTTP2	177	DATA[3101]
980	66.452573	10.233.48.96	10.233.75.34	HTTP2	104	HEADERS[3101]: 204 No Content
996	68.510553	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	92	DownlinkNASTransport, Registration reject (Payload was not forwarded)
1011	69.510365	10.233.75.34	127.0.0.1	HTTP2	81	RST_STREAM[3099]
1116	76.449474	10.233.75.34	127.0.0.1	HTTP2	164	HEADERS[3103]: PATCH /nnrf-nfm/v1/nf-instances/6e2970d2-b6dc-41ef-8831-81c7a41a89bc
1118	76.449631	10.233.75.34	127.0.0.1	HTTP2	177	DATA[3103]
1124	76.455188	10.233.48.96	10.233.75.34	HTTP2	104	HEADERS[3103]: 204 No Content
1153	79.401490	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	140	InitialUEMessage, Registration request
1155	79.402770	10.233.75.34	127.0.0.1	HTTP2	145	HEADERS[3105]: POST /nausf-auth/v1/ue-authentications
1156	79.402882	10.233.75.34	127.0.0.1	HTTP2/JSON	183	DATA[3105], JavaScript Object Notation (application/json)
1162	79.434704	10.233.48.96	10.233.75.34	HTTP2	458	HEADERS[3105]: 201 Created, DATA[3105]
1164	79.435412	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	148	SACK (Ack=4, Arwnd=106496) , DownlinkNASTransport, Authentication request
1165	79.444914	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	148	SACK (Ack=4, Arwnd=106496) , UplinkNASTransport, Authentication response

Figure 4: Test trace of "*TC_Sync_Fail_Seaf* - case B" in Open5GS Core Network



No.	Time	Source	Destination	Protocol	Length	Info
470	33.632970	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	136	DownlinkNASTransport, Authentication request
471	33.640693	10.100.50.251	10.100.50.249	NGAP/NAS-5GS	152	SACK (Ack=1, Arwnd=106496) , UplinkNASTransport, Authentication failure (Synch failure)
479	33.644136	10.233.71.26	10.233.59.143	HTTP2	96	Magic
481	33.644318	10.233.71.26	10.233.59.143	HTTP2	99	SETTINGS[0]
483	33.644540	10.233.71.26	10.233.59.143	HTTP2	85	WINDOW_UPDATE[0]
484	33.644665	10.233.59.143	10.233.71.26	HTTP2	127	SETTINGS[0], SETTINGS[0], WINDOW_UPDATE[0]
486	33.644716	10.233.71.26	10.233.59.143	HTTP2	192	HEADERS[1]: POST /nausf-auth/v1/ue-authentications
488	33.644911	10.233.71.26	10.233.59.143	HTTP2	81	SETTINGS[0]
489	33.645024	10.233.71.26	10.233.59.143	HTTP2/JSON	275	DATA[1], JavaScript Object Notation (application/json)
492	34.642106	10.100.50.249	10.100.50.251	NGAP/NAS-5GS	96	DownlinkNASTransport, Registration reject (Illegal UE)

Figure 5: Test trace of "*TC_Sync_Fail_Seaf* - case B" in OAI Core Network

"Illegal UE" cause, leading to the disconnection of a valid UE from the network. Unlike in Open5GS, the UE does not attempt to reconnect, resulting in a complete failure to meet the test requirements.

C.2. Tc_Ue_Sec_Cap_Handling

The "Tc_Ue_Sec_Cap_Handling" focused on testing how the AMF handles user registration when the security parameters are incomplete or invalid. Specifically, it examined how the AMF reacts to messages missing encryption or integrity algorithms or using unsupported mandatory algorithms. According to security standards, the AMF should reject these registrations with a "Registration Reject" message. This test involved two main components: the UE and the AMF, communicating through the NAS protocol. The challenges included identifying the required algorithms, modifying registration requests, and verifying the AMF's response.

To cover all scenarios, four separate tests were performed: i) No 5GS encryption algorithms; ii) No 5GS integrity algorithms; iii) Mandatory 5GS encryption algorithms not supported, and iv) Mandatory 5GS integrity algorithms not supported. Proxies were used to intercept and modify the registration request messages sent by the UE. These proxies adjusted the encryption and integrity algorithm fields, either setting them to zero or excluding mandatory algorithms, based on official 3GPP guidelines. The AMF's response was then monitored. If it sent a "Registration Reject" message to the UE in each scenario, the test passed, confirming the AMF correctly handled the invalid security parameters.

The results showed that the free5GC Core Network passed all four scenarios, demonstrating full compliance. In contrast, OAI failed in every scenario, indicating a lack of proper handling for invalid security parameters. Open5GS presented mixed results: it passed the second and fourth scenarios (those involving integrity algorithms) but failed the first and third scenarios related to encryption algorithms.