IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 January 2024, accepted 20 February 2024, date of publication 23 February 2024, date of current version 29 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3369613

RESEARCH ARTICLE

Formal-Guided Fuzz Testing: Targeting Security Assurance From Specification to Implementation for 5G and Beyond

JINGDA YANG, (Graduate Student Member, IEEE), SUDHANSHU ARYA, (Member, IEEE), AND YING WANG, (Member, IEEE)
School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ 07030, USA
Corresponding author: Ying Wang (ywang6@stevens.edu)

This work was supported by the Defense Advanced Research Project Agency (DARPA) under Grant D22AP00144.

ABSTRACT Softwarization and virtualization in 5G and beyond necessitate thorough testing to ensure the security of critical infrastructure and networks. This involves identifying vulnerabilities and unintended emergent behaviors, from protocol designs to their software stack implementation. Formal methods are efficient in abstracting specification models at the protocol level, while fuzz testing provides comprehensive experimental evaluations of system implementations. However, the state-of-the-art in formal and fuzz testing is both labor-intensive and computationally complex. To provide an efficient and comprehensive solution, we propose a novel, first-of-its-kind approach that combines the strengths and coverage of formal and fuzzing methods. This approach efficiently detects vulnerabilities across protocol logic and implementation stacks in a hierarchical manner. We design and implement formal verification to detect attack traces in critical protocols. These traces then guide subsequent fuzz testing, and feedback from fuzz testing is used to broaden the scope of formal verification. This innovative approach significantly improves efficiency and enables the auto-discovery of vulnerabilities and unintended emergent behaviors from the 3GPP protocols to software stacks. We demonstrate this approach with the 5G Non-Stand-Alone (NSA) security processes, which have more complicated designs and higher risks due to compatibility requirements with legacy and existing 4G networks, compared to 5G Stand-Alone (SA) processes. We focus on the Radio Resource Control (RRC), Non-access Stratum (NAS), and Access Stratum (AS) authentication processes. Guided by the identified formal analysis and attack models, we exploit 61 vulnerabilities, including 2 previously undiscovered ones, and demonstrate these vulnerabilities via fuzz testing on srsRAN platforms. These identified vulnerabilities contribute to fortifying protocol-level assumptions and refining the search space. Compared to state-of-the-art fuzz testing, our unified formal and fuzzing methodology enables auto-assurance by systematically discovering vulnerabilities.

INDEX TERMS NSA 5G, formal methods, fuzz testing, self-reinforcing solution, specifications.

I. INTRODUCTION

Verticals in 5G and next-generation infrastructure create a diverse and intricate environment consisting of software, hardware, configurations, instruments, data, users, and various stakeholders [1]. However, due to the system's complexity and the often overlooked emphasis on security by domain scientists, this formed ecosystem necessitates a comprehensive evaluation and validation. This is critical for enhancing research and improving the transitional security posture of Critical Infrastructure (CI) [2].

Despite two major state-of-the-art approaches, formal verification and fuzz testing, being proposed to detect various vulnerabilities and unintended emergent behaviors in the 5G

The associate editor coordinating the review of this manuscript and approving it for publication was Junho Hong.

network, limitations in large-scale systems and stacks still persist. Formal verification can provide a high-level concept of protocol security and logical proof of vulnerabilities, as described by Hussain et al. [3]. In contrast, fuzz testing offers a detailed and comprehensive experimental platform for detecting potential vulnerabilities in 5G code implementation, as evaluated by Klees et al. [4]. However, challenges and open issues related to selective fuzz testing and formal analysis in various scenarios remain, as noted by Souri and Norouzi and Beaman et al. [5], [6].

The limitations and challenges of implementing fuzz testing and formal analysis to detect vulnerabilities in 5G networks are multi-faceted. Formal verification offers a high-level conceptual understanding of protocol security but may not cover all real-world scenarios. In contrast, fuzz testing provides comprehensive experimental testing but faces difficulties with numerous computational resources. Our proposed approach aims to achieve a balance by leveraging formal verification to guide the fuzz testing process. This means that formal verification, with its precision, is used to initially detect high-risk logical vulnerabilities. These vulnerabilities then serve as starting points for fuzz testing, which explores various inputs and implementation vulnerabilities in a more scalable manner. This technique aims to overcome the limitations of both fuzz testing and formal analysis, enabling model checkers to detect a wide range of vulnerabilities in large, complex 5G systems.

Our paper contributes significantly to the fields of programming languages and infrastructure cybersecurity by designing and implementing a formal verification framework for 5G authentication and authorization specifications. This framework not only detects attack traces but also forms attack models, which guide subsequent fuzz testing and incorporate feedback to broaden the scope of formal verification. Such integration is crucial in the rapidly evolving landscape of 5G and beyond, where conventional methods may fall short due to the increasing complexity and scale.

The proposed interdisciplinary approach has the potential to make meaningful contributions to programming languages by improving the methods and tools available for formal verification and fuzz testing. This contribution is particularly relevant in ensuring that emerging communication technologies can be rapidly and securely integrated into existing digital infrastructure. The approach also significantly contributes to infrastructure cybersecurity by enhancing the security measures against a wide range of vulnerabilities and attack vectors, thereby strengthening system assurance and resilience against potential threats. By continually refining this approach, we aim to contribute to the development of more secure, reliable, and efficient systems in the 5G era and beyond.

Specifically, our contribution includes:

- Novel Approach for Enhancing Security Assurance: Combining formal verification and fuzz testing provides a systematic method to identify vulnerabilities, significantly enhancing security assurance for critical infrastructure and communication systems.
- Reduced Resource Intensiveness: Streamlining traditional verification and fuzz testing processes makes these methods more accessible and less resource-intensive, encouraging broader participation in cybersecurity research.
- Interdisciplinary Application: Our protocol-independent approach's applicability across various domains fosters interdisciplinary collaboration and innovation in cybersecurity solutions.
- Automatic Vulnerability Detection: Automation in our approach enables automatic vulnerability discovery, freeing researchers to focus on robust mitigation strategies.
- Real World Validation: we present a comprehensive application and evaluation of our security methodology in real 5G environments, demonstrating its viability and practical value.

In summary, our proposed approach aims to revolutionize vulnerability detection and management in large-scale systems, benefiting the research community and society by enhancing critical infrastructure and communication networks' security and reliability. We believe our work will drive innovation and substantial contributions in Programming Languages and Infrastructure Cybersecurity.

In the subsequent sections of this paper, we provide a concise overview of the structure of our proposed comprehensive formal verification and fuzz testing integrated vulnerability detection framework (Section III). Subsequently, we elucidate the mechanism behind our proposed dependency-based protocol abstraction and evaluation approach (Section IV), followed by presenting examples of dependency analysis (Section IV-D). Furthermore, we apply the dependency-based protocol abstraction and evaluation approach to the Non Standard-Alone (NSA) 5G communication establishment process (Section IV-A), where we present and analyze the results of formal verification (Section V). Additionally, we propose proven or novel solutions for each detected formal attack model. Subsequently, leveraging the identified assumptions, we apply our proposed fuzz testing framework to verify and analyze the implementation of the NSA 5G communication establishment process (Section VII). Lastly, in Section IX, we utilize intuitive visualizations to analyze the efficiency of different fuzzing strategies across various fuzzing scopes.

II. RELATED WORK AND BACKGROUND

5G technologies are rapidly becoming crucial to national and regional infrastructures and offer unprecedented connectivity benefits. However, these technologies also present an attack surface of unprecedented size due to the complexity of both the specifications and implementations of 5G stacks. Previous researchers have proposed various vulnerability detection approaches [3], [7], [8], with two categories being intensively researched: formal verification and fuzz testing.

Formal verification is a technology that translates natural language-defined protocols into symbolic logic language, enabling the establishment of the validity of propositions through a finite process of mathematical verification. Several formal analysis frameworks have been proposed in existing research to determine which security guarantees are satisfied in 5G protocols. These frameworks apply formal methods and automated verification in symbolic models, such as Tamarin [9], and 5G Reasoner [10]. Hussain et al. [3] proposed a cross-layer formal verification framework, which combines model checkers and cryptographic protocol verifiers through the application of the abstraction-refinement principle. In addition to formal verification frameworks, different formal strategies have been introduced to prove security assumptions, like those in [11]. For example, the pre-authentication message sent unencrypted has been acknowledged as the root cause of many known LTE and 5G protocol exploits [12], [13], [14]. Furthermore, some registration and access control protocols, including authentication and key agreement (AKA), RRC, etc., have applied formal methods in various frameworks [3], [10], [15]. When applied in 5G security design, necessary lemmas, helping lemmas, sanity-check lemmas, and lemmas that check relevant security properties against 5G protocols are verified [15].

A fuzz tester (or fuzzer) is a tool that iteratively and randomly generates inputs to test the quality of a target program [4]. Compared to formal analysis, fuzz testing has proven successful in discovering critical security bugs in real software [4]. For instance, [16] implemented a Radio Resource Control (RRC) fuzz testing experiment for air interface protocols. Significant effort has been devoted to devising new fuzzing techniques, strategies, and algorithms. Fuzz testing is used intensively for large-scale system cybersecurity purposes, and various strategies have been proposed to efficiently detect cyber vulnerabilities. He et al. [17] proposed a state transition fuzzing framework applicable to different types of message identifiers. To reduce the randomness and blindness of fuzzing, [18] introduced a vulnerability-oriented fuzz (VulFuzz) testing framework, prioritizing fuzzing cases by security vulnerability metrics.

Leveraging the advantages of both formal verification and fuzz testing has become a popular research topic. In [19], extreme cases like buffer overflow or incorrect format are discussed, combined with the advantages of protocol and mutation. Besides these extreme cases, rule-based fuzzing [20] focuses on covering all protocol-based cases. Under the limited directions defined by formal verification, coverage-guided fuzz [21] was proposed to test the security of cyber-physical systems. Furthermore, the state-of-the-art vulnerability detection approach [22] proposed a possible combination of formal verification and fuzz testing. For long-term, multi-time attacks, Ma et al. [23] proposed a state transaction method to analyze serial attacks. Based on formal verification, fuzz testing can efficiently locate high-risk areas. However, significant gaps remain in highly relying on pre-assumptions of prior knowledge awareness and focusing on the specific implementation of targeted protocols. Therefore, LZfuzz [24] was proposed to eliminate the requirement for access to well-documented protocols and implementations, focusing instead on plain-text fuzzing. Osborne and Pascutto [25] proposed a framework applying fuzz testing with area limitations in real-world experiments to narrow the fuzzing scope. To address the challenges of computation power, without presupposing but leveraging available prior domain knowledge, we presented a multi-dimensional, multi-layer, protocol-independent fuzzing framework in [26]. This framework aims for protocol vulnerability detection and unintended emergent behavior identification in fast-evolving 5G and NextG specifications and large-scale open programmable 5G stacks.

Compared to previous approaches, where formal verification and fuzz testing were manually guided and not seamlessly integrated, our proposed framework represents a significant advancement. The key challenge in previous work was the inability to automatically apply these methods to detect cybersecurity vulnerabilities. In contrast, our approach establishes a positive feedback loop between formal verification and fuzz testing, enabling automated and continuous vulnerability detection. This innovative integration enhances the efficiency and effectiveness of the entire vulnerability detection process, addressing a longstanding challenge in the field. By automating the interaction between formal verification and fuzz testing, we achieve a dynamic and self-improving system that can adapt to evolving threats and system changes, ultimately providing a more robust cybersecurity solution.

III. SYSTEM OVERVIEW

Aiming to provide auto-assurance for 5G and beyond specifications to stack implementations, we present a vulnerability and unintended emergent behaviors detection system. As shown in Fig. 1, the system leverages the amplification and cross-validation of fuzz testing and formal verification. Our proposed framework establishes a virtuous recursive loop through the following steps:

A. PROTOCOL ABSTRACTION

Starting with the 3GPP technical specifications (TS) and requirements (TR), we first convert natural language-based specifications into unambiguous symbolic expressions known as an authentication and authorization flow-graph (AAF). This flow-graph is then transformed into a properties table, and a dependency graph is generated. The dependency graph serves as a foundation for automatically deriving formal analysis models. This approach frees the formal analysis process from labor-intensive and expertise-dependent tasks, enabling auto-formal verification. It also facilitates incremental evolving verification by incorporating

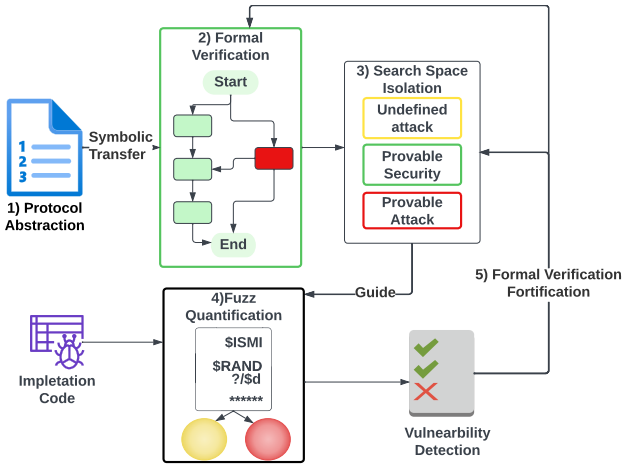


FIGURE 1. System components and connector view.

new 3GPP protocol releases into existing formal methods, thus eliminating the need to restart the protocol abstraction process for each new release.

B. FORMAL-BASED VULNERABILITY DETECTION AND ATTACK MODELS

Utilizing the dependency graph, we apply formal methods via the ProVerif platform to conduct a logical proof of security properties and potential vulnerabilities. This enables a robust and comprehensive evaluation of the system's security integrity. The formal methods applied to the abstracted protocols not only detect vulnerabilities in protocol design but also guide fuzz testing by providing space isolation.

C. SEARCH SPACE ISOLATION

The output of formal verification divides the search space into three sets: no vulnerabilities, detected attack traces, and uncertain areas needing further investigation. This division effectively narrows down the uncertain regions and guides the direction of fuzz testing.

D. FORMAL GUIDED FUZZ TESTING

With attack models detected through formal analysis, we direct and generate a list for fuzz testing. Unlike formal analysis, which focuses on specifications, initiated fuzz testing is performed on runtime binary systems, particularly targeting predefined uncertain areas and areas with identified attack traces. This guided fuzz testing aims to identify runtime vulnerabilities, complementing the detection of vulnerabilities through logical proofs on protocols and evaluating the impact of the formally detected attack models and traces. It also serves as a stochastic approach for uncertain areas that cannot be verified through formal methods.

E. FORTIFICATION OF PROTOCOL AND FORMAL VERIFICATION

Based on the vulnerabilities and unintended emergent behaviors detected by formal methods and guided fuzz

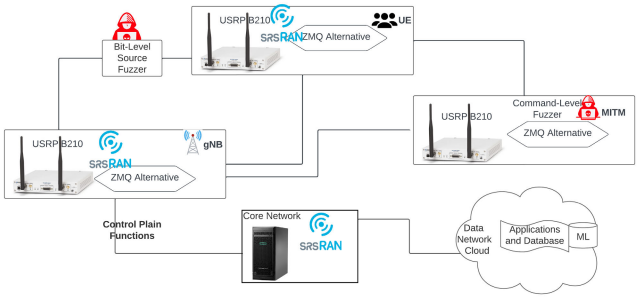


FIGURE 2. Experimental platform structure and setup [26].

testing, we develop solutions and fortifications to enhance the protocol's robustness and resilience or to narrow down search spaces. By more precisely defining the space, formal verification can be further optimized, thereby extending the scope of the security assurance area.

In summary, to address the challenges related to labor-intensive expertise and the computational complexity inherent in state-of-the-art formal and fuzz testing, our proposed system uses protocol abstraction and self-learnable iterative formal verification. This approach significantly reduces the burden of labor-intensive formal verification. Additionally, we implement formal guided fuzzing to effectively lessen the computational complexity associated with traditional fuzz testing.

Our formal verification approach relies on the ProVerif, a widely recognized and well-established formal verification tool for cryptographic protocols. ProVerif, rooted in formal methods, employs symbolic verification techniques to assess the security properties of these protocols. Our formal verification process involves modeling the targeted 5G protocol and defining the security properties to be verified, encompassing communication structures, message exchanges, and cryptographic operations. ProVerif conducts a rigorous analysis to uncover potential security flaws and attack paths. Beside formal verification, in fuzz testing, we utilize both zeroMQ as a virtualized simulation platform and the srsRAN framework as a real-world testbed, providing a platform to assess vulnerabilities under real network conditions. We further demonstrate the proposed framework by leveraging our existing platform for fuzz testing-based digital twins [26], [27], [28] in the context of 5G cybersecurity, as illustrated in Fig. 2. Both over-the-air (OTA) and zeroMQ modes in legitimate communications are executed using srsRAN. By interfacing with our digital twin platform, we enable mutation-based identifier fuzzing (Bit-Level Fuzz Testing) and permutation-based command fuzzing (Command-Level Fuzz Testing). These techniques can be used for implementation-level verification, extending formal discovery, and triggering search spaces guided by the results of formal methods. Utilizing formal result analysis, formal-guided fuzz testing, and subsequent fortification, our proposed framework constructs a reinforcing loop to enhance the system's resilience.

IV. PROTOCOL ABSTRACTION

A. PROTOCOL AND SYMBOLIC CONVERSION FOR FORMAL ANALYSIS

The NSA 5G architecture can be divided into the legacy LTE authentication process and the LTE-to-5G connection reconfiguration. Compared to the Standard-Alone (SA) 5G network architecture, the NSA 5G architecture is more widely adopted but also more vulnerable due to the cross-generation of protocols, which introduces vulnerabilities from LTE. Furthermore, data in the data-link layer during wireless communication can be vulnerable to hacking and misuse if data security is not established [29]. Therefore, we focus on the pre-authentication process of LTE in the NSA 5G architecture. As shown in Fig. 3, the LTE authentication in the NSA architecture can be divided into the following four parts:

1) **RRC Connection Setup:** RRC connection setup process aims to build up connections in RRC layer. First, User Equipment (UE) sends the RRC Connection Request command with UE-identity and establishment cause to gNodeB (gNB). Then, gNB replays with radio resource configuration to UE. If the setup process is valid, UE will send RRC Connection Setup Complete command with necessary identifiers to gNB and prepare for the following Non-Access Stratum (NAS) security setup. In the RRC connection setup process, we verify the reliability, consistency, and stability of communications between UE and gNB. Confidentiality will not be considered because the RRC connection setup process is designed for a non-encrypted environment.
2) **Mutual Authentication:** UE and core network (CN) adapt Evolved Packet System (EPS) AKA algorithm as encryption and decryption tools to set up mutual authentication. In our designed formal EPS algorithm, there are four required identifiers to get the corresponding values, $AUTN$, RES, and K_{ASME}. Even if we assume the EPS algorithm is impregnable, the previous messages containing international mobile subscriber identity (IMSI) and temporarily generated $rand\_id$ are neither ciphered nor integrity protected. The unencrypted mutual authentication process is vulnerable to disclosing the user identity under man-in-the-middle (MITM) attacks. Based on exploited vulnerabilities and properties, we test the security impact of user identity by formal verification and simulate the MITM attack mode.
3) **NAS Security Setup:** After mutual authentication, CN needs to decide encryption algorithm and integrity algorithm. To ensure the security of NAS communication setup, UE and CN communicate with integrity protection to decide encryption and integrity algorithm, and K_{ASME}, which is the top-level key to be used in the access network. Then UE and CN can get the corresponding session key for encryption and integrity of following symmetric NAS communication.
4) **AS Security Setup:** NAS security setup shares K_{ASME} between CN and UE. However, there is still necessary to establish another channel for user status management, like RRC. Therefore, CN generates a key K_{eNB} for gNB based on K_{ASME} and NAS up-link count and forward the K_{eNB} to evolved NodeBs (eNB) through the private network. Same with NAS security setup, eNB and UE share the K_{eNB} and selected encryption and integrity algorithm with integrity-protected communications. Then eNB and UE use the generated RRC encryption key, K_{RRCenc}, integrity key, K_{RRCint}, and generated User Plane (UP) encryption key, K_{UPenc}, to establish symmetric ciphered and integrity protected RRC and UP communication.

B. PROPERTIES DEFINITION AND EXTRACTION

Following the flowgraph shown in Fig. 3, we further extract four major security properties: confidentiality, integrity, authentication, and accounting, from the 3GPP specifications, which are critical for formal analysis. These four properties represent distinct aspects of security enhancement in the specifications:

1) **Confidentiality** represents the ability to prevent private information from leakage.
2) **Integrity** denotes the capability to keep the information unmodified.
3) **Authentication** means whether the receiver can identify who and when to send the message.
4) **Accounting** is identifying whether the current message follows the right order in session.

Based on the four security properties, we have generated an identifier-based Properties Table (PT), as shown in Table 1, to reflect the specifications in the control messages. The value in the security property columns indicates the identifier on which the current identifier depends (where 'N' signifies no dependency). From Note 1 in Table 1, we can infer that the RRC connection setup process, which includes three steps, is unprotected in terms of confidentiality, integrity, authentication, and accounting. For identifiers that are protected in some properties, we examine the critical keywords/identifiers in each property. The properties of these critical keywords/identifiers become the assumptions for examining that property. For example, as shown in Note 2 of Table 1, we consider the integrity of $AUTN_{HSS}$ under the assumption of a safe random number (RAND) or a leaked RAND.

The content of the Properties Table serves as the input for assumptions and properties in the subsequent formal analysis. The table reveals dependencies between rows, which determine the flow-graph for each formal model used in vulnerability detection.

C. DEPENDENCY GRAPH GENERATION

To further visualize the dependencies presented in the Properties Table, we have generated a Dependency Graph (DG) as shown in Fig.4. The Dependency Graph allows us to extract the dependency trace of identifiers, evaluate the chain effects along dependency relationships, and assess

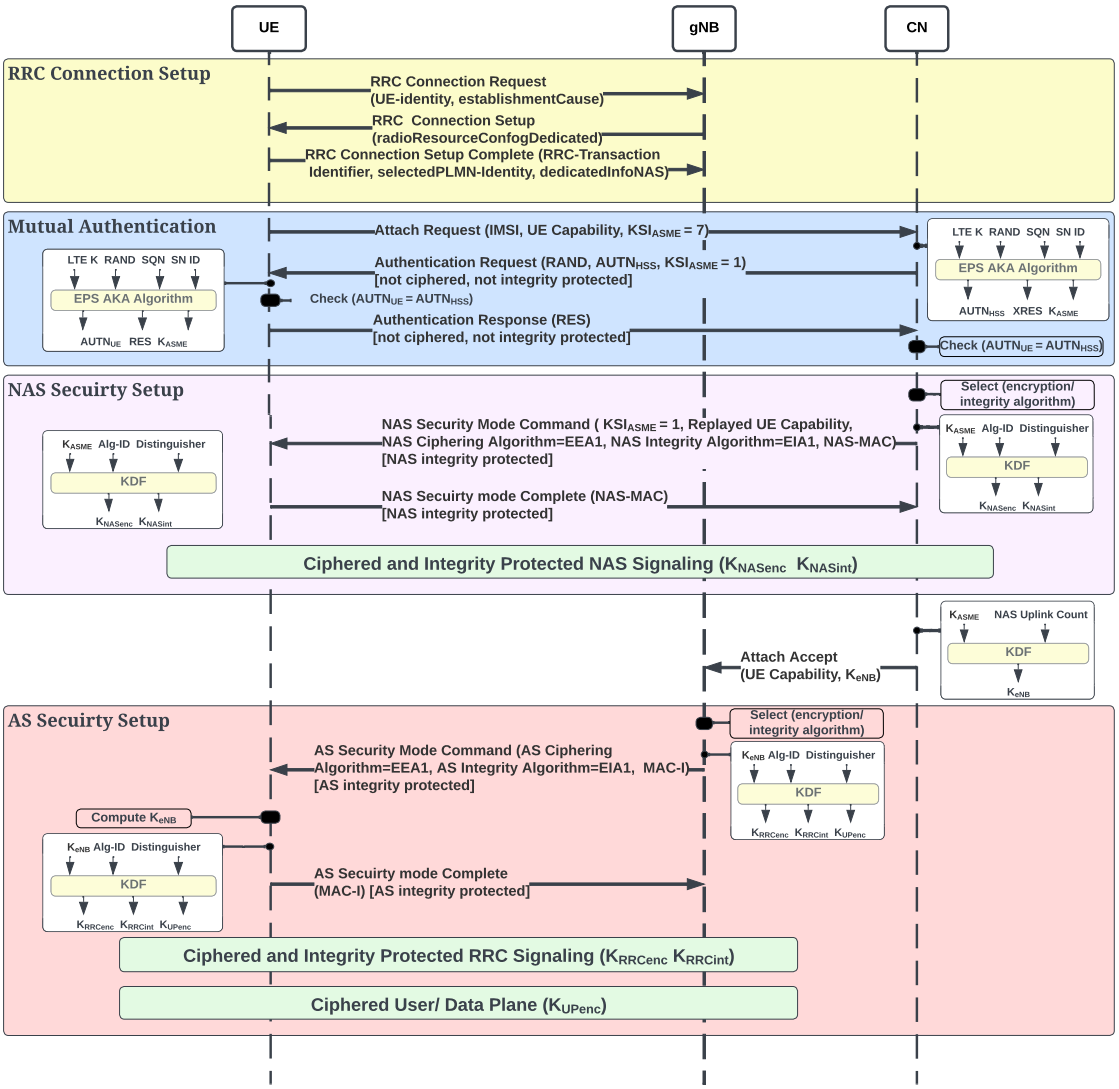


FIGURE 3. NSA and as authentication and authorization flowgraph.

multi-level security risks. For example, as also indicated in Note 3 of Table1, K_{NASenc} has a higher integrity security level than *NAS-MAC*. This is because the integrity of K_{NASenc} is protected by *NAS*; *Ciphering*; *Algorithm*, *NAS*; *Integrity*; *Algorithm*, and KSI_{AMSE}, which in turn are protected by K_{NASint}. However, the integrity of *NAS-MAC* is solely protected by K_{NASint}. Based on the security risk levels, we first verify the vulnerabilities of the low-risk identifiers and then validate the security of high-risk identifiers, building upon the proven assumptions of the low-risk identifiers. Following the security property tracks provides guidance for the testing target, which narrows down the target range and enhances the efficiency of formal and fuzz testing.

Our security level evaluation system follows the Depth-first search (DFS) principle and inherits the security level from the parent node (dependency node). As shown in Alg.1, the recursive algorithm adds the security level of dependent property to their security level vector. Based on different consideration and application scenarios, we use the Hadamard product of weight vector and security level vector in Equation 1 to determine the global security level. For instance, we can set the weight vector to $[1, 1, 0.5, 0.5]$ if we prioritize confidentiality and integrity.

$$S = [\alpha_c, \alpha_i, \alpha_{au}, \alpha_{ac}] \cdot [c, i, au, ac]^T \tag{1}$$

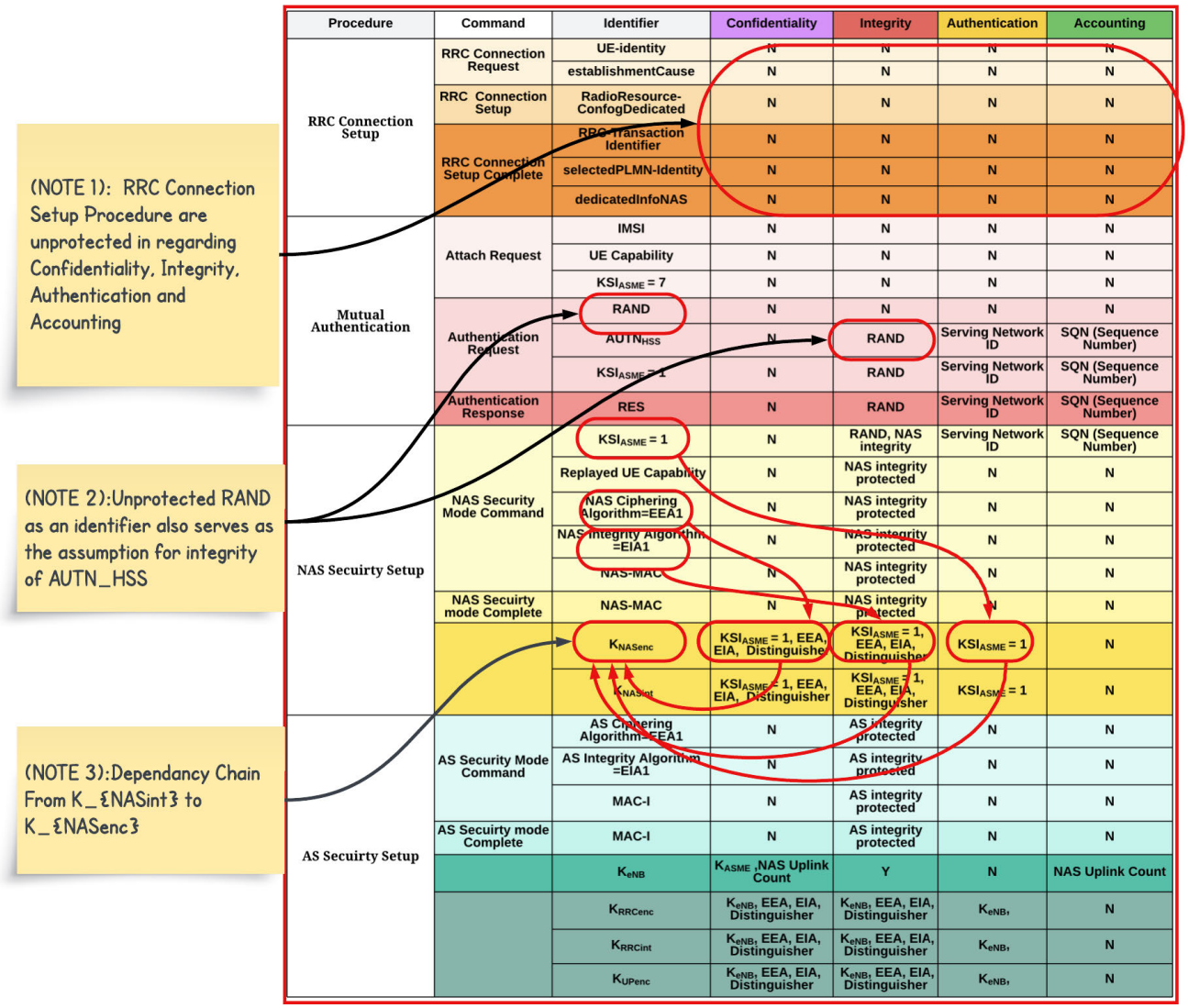
D. DEPENDENCY ANALYSIS

Based on the defined dependency graph above, we use some samples to illustrate the process mechanism of how to extract the highest risk path to the special identifier.

1) RRC CONNECTION SETUP DEPENDENCY ANALYSIS

From Fig. 4, we conclude get that all identifiers in RRC Connection Setup are not protected by encryption or integrity check. We can conclude that the security level of identifiers in RRC Connection Setup= $[0, 0, 0, 0]$.

TABLE 1. Properties table of protocol.



Procedure	Command	Identifier	Confidentiality	Integrity	Authentication	Accounting
RRC Connection Setup	RRC Connection Request	UE-identity	N	N	N	N
		establishmentCause	N	N	N	N
	RRC Connection Setup	RadioResource-ConfogDedicated	N	N	N	N
	RRC Connection Setup Complete	RRC Transaction Identifier	N	N	N	N
		selectedPLMN-Identity	N	N	N	N
		dedicatedInfoNAS	N	N	N	N
Mutual Authentication	Attach Request	IMSI	N	N	N	N
		UE Capability	N	N	N	N
		KSI_{ASME} = 7	N	N	N	N
	Authentication Request	RAND	N	N	N	N
		$AUTN_{HSS}$	N	RAND	Serving Network ID	SQN (Sequence Number)
		KSI_{ASME} = 1	N	RAND	Serving Network ID	SQN (Sequence Number)
	Authentication Response	RES	N	RAND	Serving Network ID	SQN (Sequence Number)
NAS Secuirty Setup	NAS Security Mode Command	KSI_{ASME} = 1	N	RAND, NAS integrity	Serving Network ID	SQN (Sequence Number)
		Replayed UE Capability	N	NAS integrity protected	N	N
		NAS Ciphering Algorithm=EEA1	N	NAS integrity protected	N	N
		NAS-Integrity Algorithm =EIA1	N	NAS integrity protected	N	N
		NAS-MAC	N	NAS integrity protected	N	N
	NAS Secuirty mode Complete	NAS-MAC	N	NAS integrity protected	N	N
		K_{NASenc}	KSI_{ASME} = 1, EEA, EIA, Distinguisher	KSI_{ASME} = 1, EEA, EIA, Distinguisher	KSI_{ASME} = 1	N
		K_{NASint}	KSI_{ASME} = 1, EEA, EIA, Distinguisher	KSI_{ASME} = 1, EEA, EIA, Distinguisher	KSI_{ASME} = 1	N
AS Secuirty Setup	AS Security Mode Command	AS Ciphering Algorithm=EEA1	N	AS integrity protected	N	N
		AS Integrity Algorithm =EIA1	N	AS integrity protected	N	N
		MAC-I	N	AS integrity protected	N	N
	AS Secuirty mode Complete	MAC-I	N	AS integrity protected	N	N
		K_{eNB}	K_{ASME} ,NAS Uplink Count	Y	N	NAS Uplink Count
		K_{RRCenc}	K_{eNB}, EEA, EIA, Distinguisher	K_{eNB}, EEA, EIA, Distinguisher	K_{eNB},	N
		K_{RRCint}	K_{eNB}, EEA, EIA, Distinguisher	K_{eNB}, EEA, EIA, Distinguisher	K_{eNB},	N
		K_{UPenc}	K_{eNB}, EEA, EIA, Distinguisher	K_{eNB}, EEA, EIA, Distinguisher	K_{eNB},	N

Algorithm 1 Security Level Evaluation

Data: r = Boolean vector of dependency relation.
procedure Security_Evaluation(node_v)
1: [c, i, au, ac] = [1, 1, 1, 1]
2: **while** no dependent node v' exists **do**
3: [c, i, au, ac] += Security_Evaluation(v') $\odot$ r
4: **end while**
5: **return** [c, i, au, ac]
end procedure

2) K_{NASENC} DEPENDENCY ANALYSIS

K_{NASenc} is the most critical identifier in NAS authentication process and responsible for the following NAS communication encryption. To prove the security of K_{NASenc}, we extract a logical dependency graph of K_{NASenc}, Fig. 5, from the whole dependency graph of authentication graph, Fig. 4. From Fig. 5, we can conclude that there are three direct integrity-dependent identifiers and only one direct authentication-dependent identifier. We discuss the security level from two aspects of security properties:

1) Authentication: Based on the K_{ASME} derivation function, attackers can derive the SN_id from the K_{ASME}. However, attackers can not generate the K_{ASME} from the K_{NASenc}. Based on the authentication conduction of these three identifiers, the invertibility of the path is critical for authentication tracking. The coexistence of the authentication dependency relationship and inevitability can prove the feasibility of invertible conduction from bottom to up.
2) Integrity: The trustworthiness, consistency, and accuracy of the data throughout its life cycle is termed as integrity. Based on the dependency relationship of

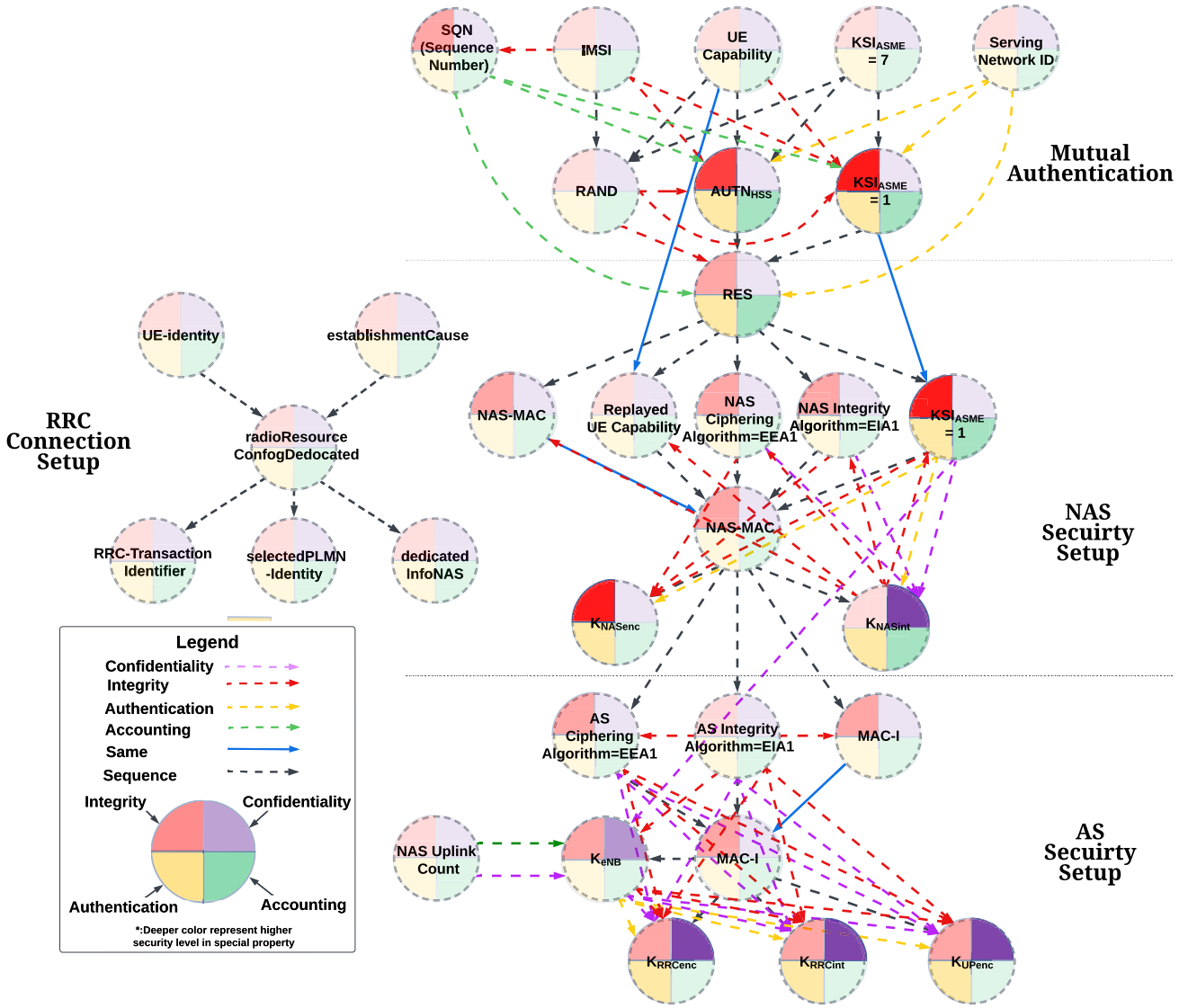


FIGURE 4. Dependency graph of protocol.

K_{NASenc}, as shown in Fig. 5, only with the ability to modify three direct identifiers, secret attackers can modify K_{NASenc} secretly. Furthermore, attackers can modify three direct identifiers only when they can modify all five second-level identifiers, which are directly connected to three direct identifiers. We can conclude that the minimum requirement of K_{NASenc} modification is 5 identifiers in 3 command, including Attach Request, Authentication Request, and NAS Security Mode Command.

From the above proof, we can get the security level of $K_{NASenc} = [0, 5, 1, 0]$.

V. FORMAL-BASED VULNERABILITY DETECTION AND ATTACK MODELS

Based on the 5G authentication and authorization specification abstraction in Sec.IV, we deploy formal models and analysis to describe the logical attack models and detect potential attack traces. In the ensuing section, we present four samples of vulnerabilities detection at disparate stages of the NSA 5G authentication process and analyze the mechanisms of the exploited attack traces: (1) User Credentials Disclosure; (2) Deny of Service (DoS) or Cutting of Device using Authentication Request, Exposing K_{NASenc} and K_{NASint}; (3) Exposing K_{RRCenc}, (4) K_{RRCint} and K_{UPenc}. Our key findings are encapsulated in Table 3 in the result Section IX-A.

A. USER CREDENTIALS DISCLOSURE

In this attack, the adversary can exploit the transparency of RRC Connection Setup process to effortlessly access critical user identity information, which includes but is not limited to the UE identity and establishment cause. This illicit access enables the adversary to acquire user information

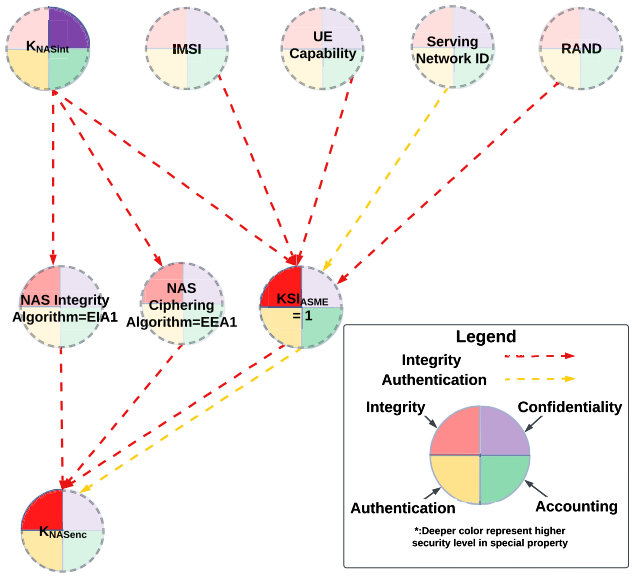


FIGURE 5. Dependency relationship of K_{NASenc}.

and use the ensuing session key for nefarious activities such as eavesdropping and manipulation of subsequent communications.

1) ASSUMPTION

The adversary can exploit the transparency of RRC Connection Setup process to directly access any identifier within the message. Furthermore, the adversary is also capable of establish a fake UE or a MITM relay to eavesdrop and manipulate the messages within the RRC Connection Setup process. To verify the security properties of identifiers within the RRC Connection Setup process, including aspects such as confidentiality and consistency, we converted the aforementioned assumptions into ProVerif code.

2) VULNERABILITY

As depicted in Fig. 3, the UE initiates the process by sending an RRC connection request to the CN. Upon receiving this request, the CN responds by transmitting the *radioResourceConfigDedicated* back to the UE. The UE, in turn, obtains authentication from the CN and responds with the $RRC - TransactionIdentifier$, $selectedPLMN - Identity$ and *dedicatedInfoNAS* to finalize the RRC connection setup. Nevertheless, this process presents an exploitable vulnerability as an adversary can access all message identifiers. Such unprotected identifiers run the risk of being eavesdropped upon and modified, potentially enabling the adversary to orchestrate a MITM relay attack.

3) ATTACK TRACE DESCRIPTION

Employing formal verification, we analyzed the confidentiality of identifiers within the RRC Connection Setup process. Through this methodical investigation, we identified two categories of identifiers with the most significant impact: user

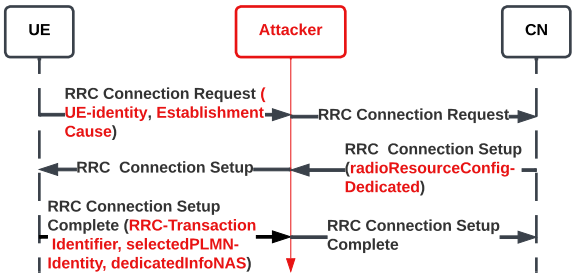


FIGURE 6. User credentials disclosure.

identities and RRC configuration identifiers. As illustrated in Fig. 6, an attacker can access the identifiers marked in red, delineating the pathway of the attack. In the initial scenario, an adversary with the access to the user identity, like $UE - identity$, is capable of launch DoS attack with real $UE - identity$. Contrary to traditional DoS attacks, which aim to overwhelm a system's capacity, an $UE - identity$-based DoS attack efficiently disrupts the CN verification mechanism through repeated use of the same $UE - identity$, leading to authentication confusion. And in second case, with computationally derived $RRC - TransactionIdentifier$, the adversary can establish a fake base station or perform a MITM relay attack by manipulating these identifiers. In the latter case, the adversary positions between the UE and the CN, intercepting and modifying communications in real-time. Consequently, this attack model presents a severe threat to the security and integrity of the mobile network's communication.

B. DOS OR CUTTING OF DEVICE USING AUTHENTICATION REQUEST

In the mutual authentication process, not only Attach Request command sent from UE is neither ciphered nor integrity protected, but the Authentication Request command sent from CN is also. Attackers can directly record and replay commands to cut off UE.

1) ASSUMPTION

After CN receives the Attach Request command sent from UE, CN replies Authentication Request command to confirm whether UE is going to attach to the network and share the session key. However, because the Authentication Request command is neither ciphered nor integrity protected, UE will be hard to verify who and when send the command.

2) VULNERABILITY

Due to the non-confidentiality of the Authentication Request command, attackers can repeat the authentication request command to multi UEs, as shown in Fig. 7. It is hard for UE to identify which authentication request command is valid. Multi-times of authentication request command broadcasting can lead to DoS attacks or cutting of UE. Compared to the User Credentials Disclosure, the formal model for "DoS or Cutting of Device using Authentication

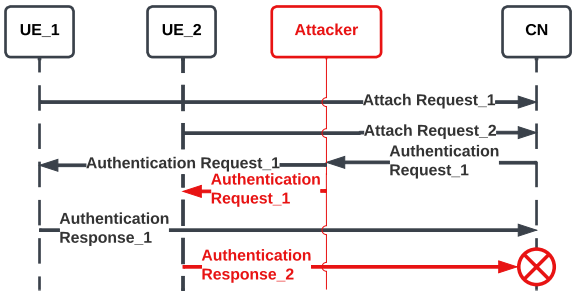


FIGURE 7. DoS attack.

Request" is significantly more complicated. Thus, we present the formal proof of cutting off connection result shown in Fig. 8 about the interaction between 5G RAN, real-UE and fake-UE.

C. EXPOSING K_{NASENC} AND K_{NASINT}

NAS security establishment is only protected with integrity but not encryption, which allows attackers to access all the information but not to modify them. Attackers can fake as UE or base station with enough information of authentication process.

1) ASSUMPTION

Commands of the security authentication process in NAS security setup is only protected by K_{NASenc}, a key generated based on the identifiers of the first command.

2) VULNERABILITY

Because commands of NAS security mode setup are not ciphered, attackers can access the necessary identifiers and generate the corresponding session key for the following communications based on the corresponding key derivation function (KDF). Then, attackers can pretend to be a base station to communicate with victim UE, as shown in Fig. 9. With proof of formal verification, attackers can block the communication from UE to gNB and continue the NAS security setup process as the base station.

D. EXPOSING K_{RRCENC}, K_{RRCINT} AND K_{UPENC}

Similar to NAS security setup process, Access Stratum (AS) security setup process is only integrity protected. All necessary identifiers of the following RRC and UP communications are transparent to attackers.

1) ASSUMPTION

Similar to NAS security setup process, all commands of AS security setup process are only integrity protected without encryption. Attackers can generate RRC and UP session keys based on eavesdropped identifiers, like Fig. 10.

2) VULNERABILITY

Based on the eavesdropped K_{RRCenc}, K_{RRCint} and K_{UPenc}, attackers can monitor, hijack, and modify the commands between UE and CN.

VI. SEARCH SPACE ISOLATION

The output of formal verification divides the search space into three sets: no vulnerabilities, attack trace detected, and uncertain areas that need further investigation. The division of the search space effectively narrows down the uncertain regions and enables the scalability of vulnerability detection. Fig. 11 is the visual representation of the vulnerability space. The blue area indicates the formal converted areas. Based on the conclusion from formal analysis, some traces are formally provable secure, represented by green sets in Fig. 11, and some traces are provable attacks, characterized by dark purple sets, and there is attack variance, represented by yellow sets, which are not provable by formal methods. In addition, large spaces cannot be converted by the formal method, including implementation errors and non-logical describable areas, or spaces that could be more labor-intensive and impractical to perform formal analysis.

Thus, we introduce fuzz testing to connect with and be guided by the formal result. The formal guided fuzz testings function for two purposes:

- Compensate for areas that remain uncovered by formal verification.
- Evaluate the potential risks and impacts of the formal provable attack sets.
- Detect identifier level unintended emergent behaviors.

VII. METHODOLOGY OF FORMAL GUIDED FUZZ TESTING

As detailed in Section V, formal verification divided the system's security landscape into three zones: safe, non-safe, and unprovable. While the safe area necessitates no further scrutiny, the non-safe and unprovable areas warrant further investigation using fuzz testing. Specifically, we leverage fuzz testing to evaluate the risks of impact of the non-safe areas within implementation stacks, as well as to ascertain the security level within the regions previously undetermined. By leveraging our previously developed viFuzzing platform [27], [28], [30] that enables bit-level and command-level fuzz testing for 5G and Beyond protocols and implementation stacks, we effectively perform formal guided fuzz testing and demonstrate in the range described in Fig.11. In this session, we present two sets of bit-level fuzzing and nine sets of command-level fuzzing to illustrate the operation of our formally guided fuzzing framework.

We set up a relay attack mechanism interfacing our developed platform viFuzzing and srsRAN [31] following the attack traces detected by formal verification. The detailed description can be referred to [32]. Fig. 12 demonstrates the setup. We have set up the srsRAN as the UE to manage the USRP B210 device, enabling communication with the Amarisoft Call Box. This call box serves as both the gNB and the CN. For a more open and customizable Radio Access Network (RAN) and Core Network, the Amarisoft Call Box can be substituted with srsRAN [32].

The overview structure of the framework that implements formal guided fuzz testing is shown in Fig. 13, which illustrates the dependency and flowgraph between formal

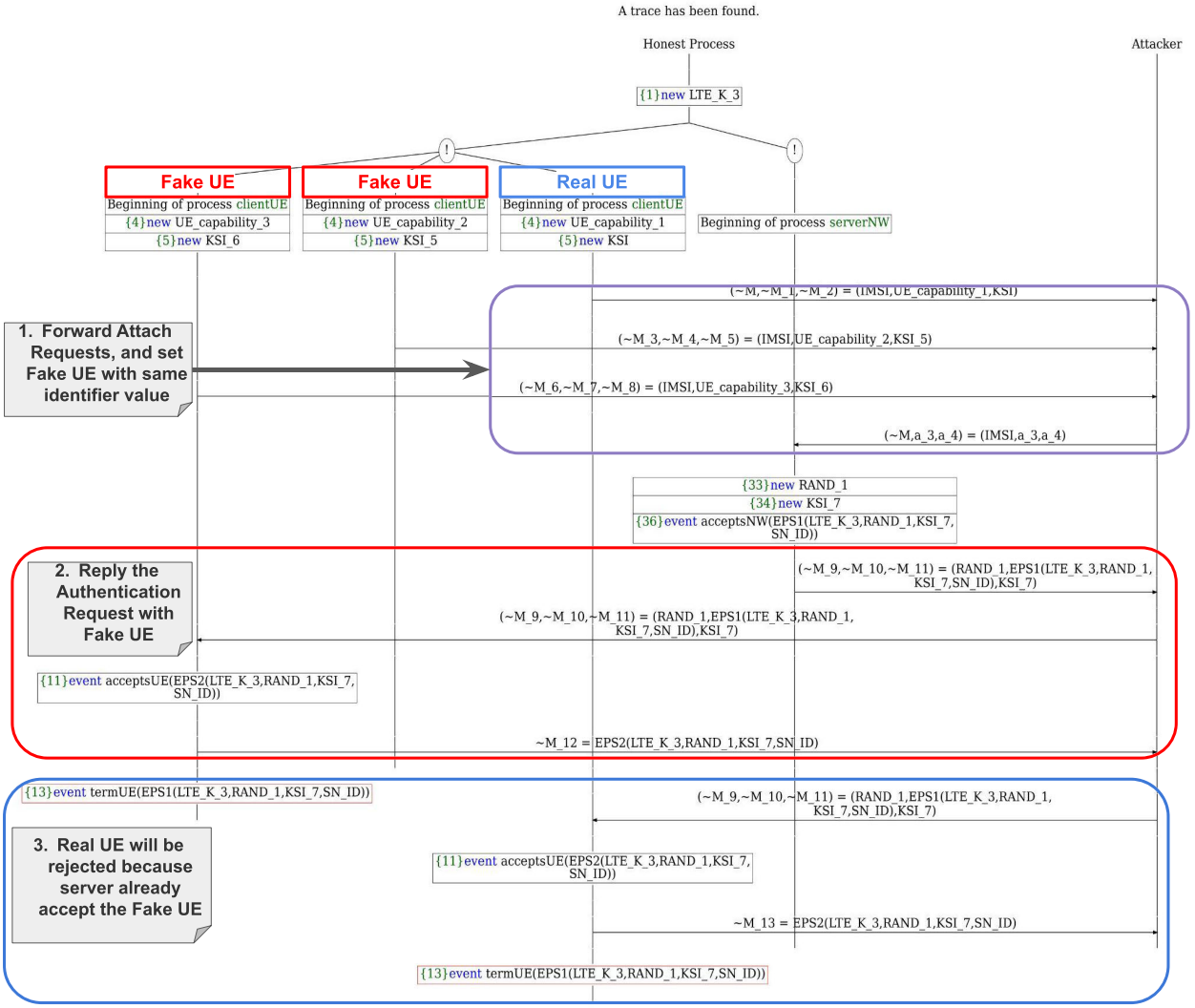


FIGURE 8. MITM in mutual authentication.

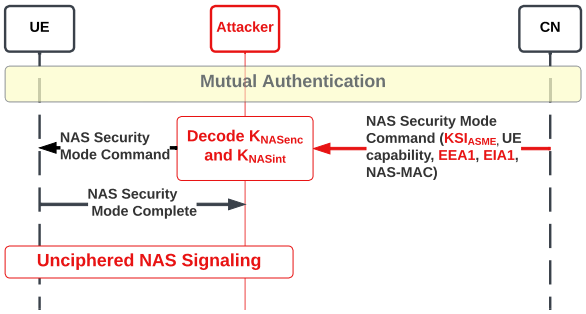


FIGURE 9. Exposure of NAS.

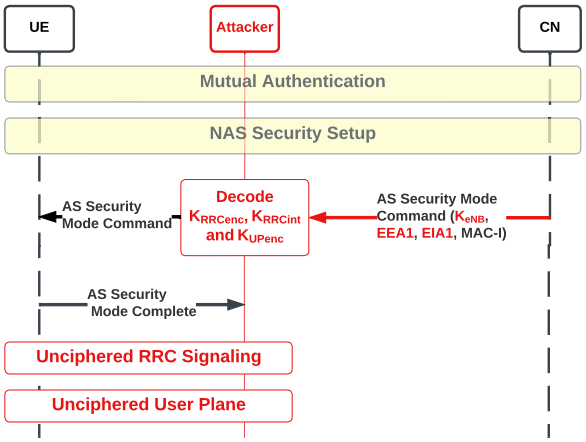


FIGURE 10. Exposure of AS.

verification detections and fuzz testing results. We further present the formal guided fuzz testing cases that addressed the four detected vulnerabilities using formal analysis in Sec. V.

A. MODIFICATION OF ESTABLISHMENTCAUSE

Based on the proved result of formal verification, we fix the value of C-RNTI and replay the RRC connection request commands with different values of identifier *EstablishmentCause*. Through the fuzzing result from Table. 2, modification of *EstablishmentCause* can lead to the expected result from formal verification, but the modification of UE-Identity can not affect the connection

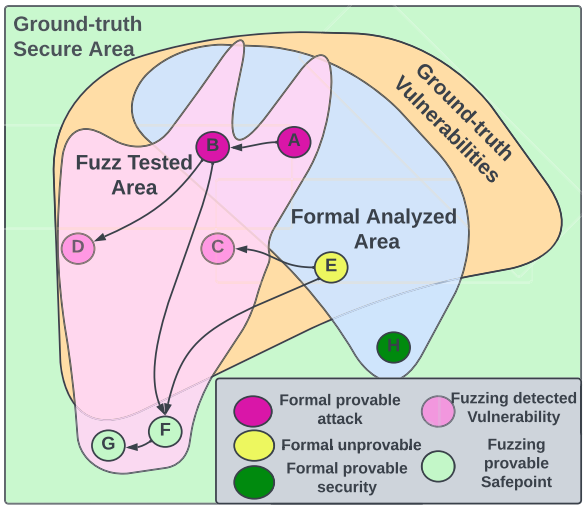


FIGURE 11. Definition of vulnerability region.

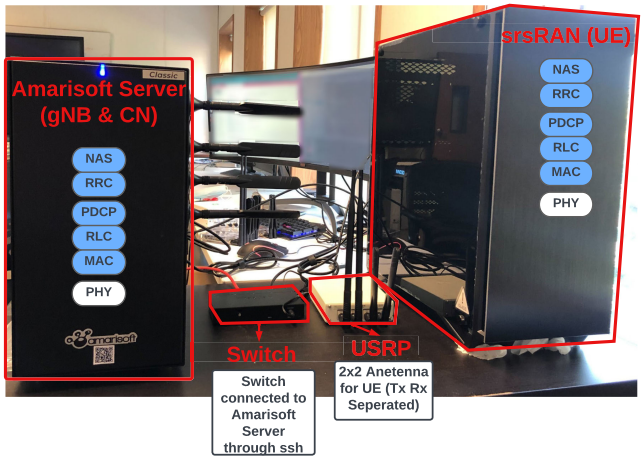


FIGURE 12. OTA mode experimental setup and configuration [32].

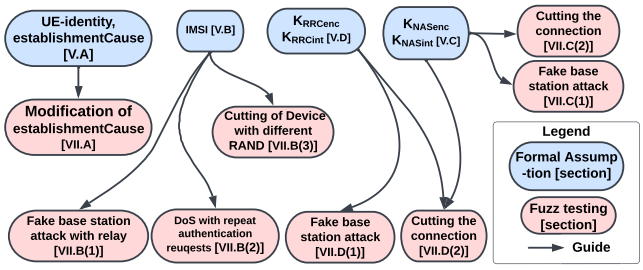


FIGURE 13. Integrated solution of formal and fuzz testing.

as expected. We prove that the implementation of the srsRAN [31] platform prevent some vulnerabilities of NSA 5G communication protocol.

Besides bit-level fuzzing, we also use command-level fuzzing to test the vulnerability of incarceration with rrc_{reject} and $rrc_{release}$ [3]. When we fixed the C-RNTI, we found the reply with rrc_{reject} and $rrc_{release}$ can lead to disconnection and repeat rrc_{reject} and $rrc_{release}$ can lead to failed connections.

TABLE 2. Fuzzing result of *establishmentCause* modification.

Command	Identifier	Fuzzing Value	Result
RRC Connection Request	ue-Identity	00	successful
		01	successful
		10	successful
	Establishment Cause	0000	emergency
		0001	nulltype
		0010	high_prio_access
		0011	nulltype
		0100	mt_access
		0101	nulltype
		0110	mo_sig
		0111	nulltype
		1000	mo_data
		1001	nulltype
		1010	delay_tolerant_access_v1020
		1011	nulltype
		1100	mo_voice_call_v1280
		1101	nulltype
		1110	spare1
		1111	nulltype

B. REPEAT AUTHENTICATION REQUEST COMMAND

Based on Section V-B, the attacker can disconnect multi UEs with the repeat of Authentication Request. Therefore, in our fuzzing attack model, the attacker can record the Authentication Request command from one UE and forward the recorded Authentication Request command to other UEs. To verify the performance of the fuzzing framework, we set up three following scenarios:

1) **Only attacker can send command to UE.** In this case, UE replies authentication response and try to establish a connection, which proves what we found in Section V-B by the formal method.
2) **One CN and multi attackers compete to send same command to UE.** Even if UE gets confused by multi-times of authentication requests, UE still has the ability to reply by sending an authentication response to CN.
3) **One CN and multi attackers compete to send different command to UE.** In this scenario, while attackers use different RAND and disclosure IMSI to generate different Authentication Request commands and forward different commands to UE, UE is more likely to reply to the attackers' requests.

C. EXPOSURE OF K_{NASENC} AND K_{NASINT}

From Section V-C, we can conclude that the attacker in the MITM relay model has the ability to act as either UE or CN. Compared to complex initial steps in the traditional fuzz testing model, our proposed fuzzing framework only needs a few steps to prove the feasibility and detect the implementation vulnerabilities. We illustrate the detailed fuzzing implementation based on formal assumptions in the following:

1) **MITM attack as fake base station.** Unlike the traditional fuzz testing approach, our framework can do fuzz testing with only access to communicated commands. The following steps illustrate the process flow of our novel proposed framework:
   - First, our framework records normal communication commands.
   - Then, our framework forwards the commands between UE and CN as normal until mutual authentication establishment with fixed same IMSI and RAND.
   - After mutual authentication is established, our framework intercepts the commands from UE and reply with corresponding commands based on the record communication history.

   The result proves that attackers have the ability to deploy MITM attack as the fake base station.
2) **Cutting the connection between UE and CN.** Besides fuzz testing of the fake station with blocked signals, our framework can verify the feasibility of signal competition. The detailed process is listed as follows:
   - First, our framework records multi-times of normal communication commands with different IMSI and RAND.
   - Then, our framework establishs mutual authentication with another IMSI and RAND.
   - Unlike the previous fuzz testing case, our framework replies with corresponding commands and forwards the commands from CN, which simulates the DoS attack.

   Most DoS attacks cut off the connection between UE and CN. The result proves the vulnerabilities of NAS security setup process. We can conclude the multi NAS security mode commands attack is an efficient attack model.

D. EXPOSURE OF K_{RRCENC}, K_{RRCINT} AND K_{UPENC}

Similar to fuzz testing on NAS security setup, we design two kinds of fuzzing strategies:

1) **MITM attack as fake base station.** Same with NAS fuzzing case, attackers can successfully fake as a base station when blocking the signals from CN.
2) **Cutting the connection between UE and CN.** DoS attacks with multi times of AS security mode commands have a high probability of cutting off the connection between UE and CN.

VIII. CASE STUDY FOR FORMAL GUIDED FUZZ TESTING

Based on the results from formal analysis and guided fuzz testing, vulnerabilities detected by fuzz testing are feedback to the formal result and search space, which lead to the fortification of protocol and formal verification. This is a crucial component in improving the resilience of 3GPP specifications.

A. USER CREDENTIALS DISCLOSURE

The adversary can exploit the transparency of RRC Connection Setup process to effortlessly access critical user identity information, which includes but is not limited to the UE identity and establishment cause. This illicit access enables the adversary to acquire user information and use the ensuing session key for nefarious activities such as eavesdropping and manipulation of subsequent communications.

Given the significance and susceptibility of identifiers within the RRC Connection Setup process, it is imperative to implement integrity protection measures for the $RRC - TransactionIdentifier$. Additionally, adopting a hash value approach can assist in preventing the disclosure of UE identity, further reinforcing security measures in this critical process.

B. DOS OR CUTTING OF DEVICE USING AUTHENTICATION REQUEST

In the mutual authentication process, not only Attach Request command sent from UE is neither ciphered nor integrity protected, but the Authentication Request command sent from CN is also. Attackers can directly record and replay commands to cut off UE.

Based on the analysis of detected vulnerabilities, it is necessary to develop a verification mechanism to identify the validation of commands. The encryption or integrity protection of Authentication Requests becomes necessary for mutual authentication to guarantee the security of initial identifiers for the security establishment process. Based on the principle of minimum change of the current protocol, we propose the following three solutions:

- **Ensured confidentiality Authentication and Key agreement (EC-AKA) [33].** EC-AKA proposed new asymmetric encryption to enhance user confidentiality before symmetric encryption is determined. However, this solution increases the cost of stations like public key broadcasting.
- **Hash value to represent IMSI [34].** This approach can prevent attackers from getting the users' identities. However, attackers can still modify or deploy DoS attacks.
- **Hash value with integrity protection [35].** Khan et al. proposed a combined solution, which uses hash values to represent IMSI and adds checksum value to protect integrity. Furthermore, the following commands in the LTE security setup process can be encrypted by original IMSI, which is invisible to the attacker but known to UE and CN. Hash value with integrity protection is an optimal solution that can provide enough security for user identity at a low cost.

C. EXPOSING K_{NASENC} AND K_{NASINT}

NAS security establishment is only protected with integrity but not encryption, which allows attackers to access all the information but not to modify them. Attackers can fake as

UE or base station with enough information of authentication process.

Same with Section VIII-B, there are two encryption methods to protect the NAS security setup:

1) Broadcasting asymmetric public key from gNB can be applied to encrypt the commands.
2) NAS security setup process can encrypt with original IMSI as symmetric key, while the hashed IMSI is used for RRC connection setup.

D. EXPOSING K_{RRCENC}, K_{RRCINT} AND K_{UPENC}

Similar to NAS security setup process, AS security setup process is only integrity protected. All necessary identifiers of the following RRC and UP communications are transparent to attackers.

As proposed in previous sections, we can use asymmetric encryption to cipher the communicated commands between UE and gNB. And we also can use hashed IMSI as the symmetric key to encrypt the commands.

IX. RESULT ANALYSIS

A. VULNERABILITY FINDINGS VIA FORMAL METHOD AND GUIDED FUZZ TESTING

The detailed detected attack models and vulnerabilities have been described in details in the previous sessions. The summary of the vulnerabilities findings are listed in Table 3. At the protocol level, 4 attack model categories, including modification of Radio Resource Control (RRC) connection, Denial of Service (DoS) or device disconnection using Authentication Request, exposure of K_{NASenc} and K_{NASint}, and exposure of K_{RRCenc}, K_{RRCint}, and K_{UPenc}, are extrapolated from the attack traces inferred through formal verification. Following the proposed formal guided fuzz testing framework shown in Fig.1. In bit-level guided fuzzing, our system uncovers 8 vulnerabilities. In command-level fuzzing, our framework detected 44 vulnerabilities. Via the systematic approach, the list of vulnerabilities and proposed solutions and fortifications significantly enhance the resilience of the 3GPP specification and large-scale implementations, like srsRAN in our demonstration. More importantly, unlike the state-of-the-art by-piece vulnerability detection, it addressed the foundations for achieving assurance for Future G authentication and authorization in providing the panoramic vision and examination of the to-date 5G specifications.

B. SYSTEM ASSESSMENT OF COMPUTATION COMPLEXITY IN FORMAL GUIDED BIT-LEVEL FUZZING

Fuzz testing is a systematic brute-force vulnerability detection approach that involves providing large amounts of random data to find security vulnerabilities. However, it is not computationally feasible to complete vulnerability detection for the whole 5G NSA protocol, even for a single command. State of the art rule-based bit-level fuzz testing strategy has been proposed, such as [20], which narrows the scope of fuzz testing to specific identifiers by following the protocol rules. Although the rule-based mutation fuzz testing strategy achieves an order of magnitude reduction in computational complexity, there are still meaningless randomly generated inputs. Our proposed formal-guided fuzz testing strategy follows formal verification assumptions and generates three sets of a few representative inputs: formal-based legal inputs, formal-based illegal inputs, and randomly generated inputs. Formal-based inputs must follow the protocol-defined rules or format, but not randomly generated inputs.

One of the novelties and advances lies in the scalability of our proposed system as the number of commands increases in complex protocols. To verify complex protocols via formal methods, formal analysis requires significant manpower and computational power. Meanwhile, attempting to cover the entire space via fuzz testing in the current state-of-the-art methodology requires an enormous number of test cases and impractical computation time, as the size of fuzz testing in the brute fuzzing strategy exhibits exponential growth. On the contrary, our presented formal-guided fuzz testing approach maintains linear growth as the number of commands increases. In this session, we perform a quantitative comparison between brute force fuzz testing, state of the art bit-level fuzzing, and the formal guide fuzz testing.

As depicted in Eq. 2, the brute-force fuzzing strategy indiscriminately flips bits within randomly selected command sets. Conversely, the rule-based fuzzing strategy [20], as expressed in Equation 3, confines bit modifications to the identifiers within randomly chosen command sets. In contrast to these approaches, our formal-guided fuzz testing identifies the bit-level fuzzing command first. Focusing on the target commands and restricts alterations to various types of identifiers, as elucidated in Eq. 4.

For the brute force fuzz testing complexity:

$$N_{brute\_force} = 2^{\sum_{k=0}^{K} |c_k|} \tag{2}$$

where N_{brute_force} denotes the number of fuzz testing cases via Brute Force. $C = [c_1, c_2, \cdots, c_K]$ is the sets of potential commands in the target procedures fuzz testing. K represents the number of target commands in fuzz testing, whereas $|c_k|$ is the number of bits in command k.

For state of the art rule-based fuzz testing complexity:

$$N_{rule\_based} = 2^{\sum_{k=0}^{K} |c\_I_k|} \tag{3}$$

where $c\_I_{k,i} \in [c\_I_{k,1}, c\_I_{k,2}, \cdots, c\_I_{k,i}]$ represents the identifier sets in command c_k and $|c\_I_{k,i}|$ is the number of bits in identifier i in command c_k, $\sum_{i=1}^{I_k} |c\_I_{k,i}| \ll |c_k|$ where I_k is the number of identifiers in command c_k.

For formal guided fuzz testing complexity:

$$N_{formal\_guided} = \sum_{k=1}^{T} \sum_{j=1}^{|ct\_I_k|} type(ct\_I_{k,j}) \tag{4}$$

where $T = |C\_target|$ is the number of target commands, whereas $C\_target = [ct_1, ct_2, \cdots, ct_T]$. It is to be noted that C_target represents a subset of commands that were detected

TABLE 3. Summary of vulnerability findings and comparison with existing exploits.

Formal Derived Attack Models	Vulnerability	Assumption	Related Existing Exploits	Solution	Guidance to fuzz	Executable Vulnerabilities via Guided Fuzzing
Modification of RRC Connection	Modified commands can disable the RRC functions	known C-RNTI or TMSI	Related to [3]	Integrity protection	Fuzz testing can start with different RRC status.	54
DoS or Cutting of Device using Authentication Request.	UE accepts authentication request without integrity.	None	Related to [36] [33] [34] [35]	• Ensured confidentiality Authentication and Key agreement (EC-AKA) [34] • Hashed IMSI [35] • Hashed IMSI with integrity check [36]	Repeat authentication request commands can be fuzzed at random time to test DoS and cutting of device attack.	3
Exposing K_{NASenc} and K_{NASint}	All NAS information can be monitored, hijacked and modified.	known IMSI, MITM relay	Related to [37]	• Asymmetric encryption • Hashed IMSI based encryption	NAS fuzz testing can start with known K_{NASenc} and K_{NASint}.	2
Exposing K_{RRCenc}, K_{RRCint} and K_{UPenc}	All RRC and UP information can be monitored, hijacked and modified.	known IMSI, MITM relay	New Discovery	• Asymmetric encryption • Hashed IMSI based encryption	RRC fuzz testing can start with known K_{RRCenc} and K_{RRCint}; UP fuzz testing can start with known K_{UPenc}.	2

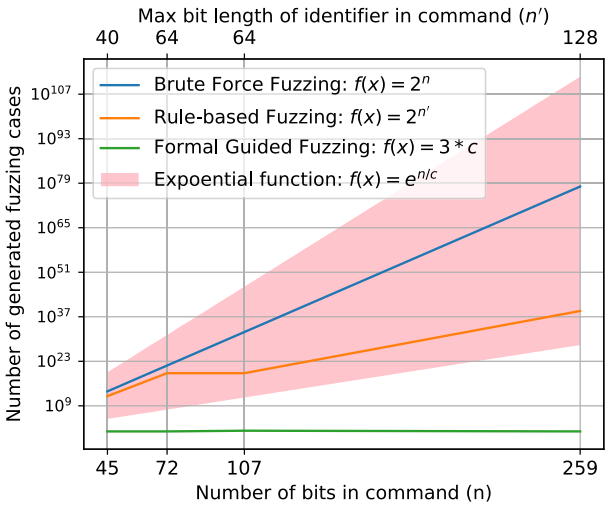


FIGURE 14. Comparison of different bit-level fuzzing strategy efficiency.

by a formal analysis as vulnerable commands that needed to be tested with fuzzing, that is, $C\_target \subseteq [c_1, c_2, \cdots, c_K]$. $|ct\_I_k|$ denotes the number of identifiers in target command ct_k. $ct\_I_{k,j}$ is the identifier j of target command ct_k, while $type(ct\_I_{k,j})$ is the number of logical types of identifier j in target command ct_k, including legal and valid value, legal and invalid value, and illegal random value.

The comparison of computation complexity following Eq. 4 with 4 fuzz strategies is shown in Fig. 14, in which fuzzing strategies are selected based on various application scenarios.

1) **Connection Request command bit-level fuzzing:** Based on the guidance of formal verification in Section V-A, the RRC Connection Request command, which includes 40 bits of UE-Identity, 4 bits of *EstablishmentCause*, and 1 bit of spare, is vulnerable to DoS or MITM attacks. Traditional brute-force fuzz testing generates more than 2^{45} fuzzing cases, and rule-based fuzzing generates $2^{40} + 2^4 + 1$ fuzzing cases based on the defined identifiers. However, our formal guided fuzzing strategy requires only 9 fuzzing cases, including one legal UE-Identity case, one illegal UE-Identity case, one random out-of-rule UE-Identity case, 2 legal/illegal *EstablishmentCause* cases, 1 random out-of-rule *EstablishmentCause* case, one legal spare case, one illegal spare case, and one out-of-rule spare case.
2) **Authentication Request command bit-level fuzzing:** Formal verification proved the Authentication Request command is the critical part for DoS or fake station attacks. Inside the Authentication Request command, there are 128 bits of *RAND*, 128 bits of $AUTN_{HSS}$ and 3 bits of KSI_{ASME}. Our proposed formal guided fuzzing strategy generates 3×3 fuzzing cases, while brute-force fuzzing generates 2^{259} cases and rule-based fuzzing generates $2^{128} + 2^{128} + 2^3$ cases.
3) **NAS Security Mode command bit-level fuzzing:** To verify the formal assumptions in MITM and cutting of the connection attacks, we make bit-level fuzzing on NAS Security Mode command. NAS Security Mode command has 3 bits of KSI_{AMSE}, 4 octets of UE capability, 4 bits of *EEA*1, 4 bits of *EIA*1, and 8 octets of $NAS - MAC$. Brute force fuzzing needs all possible permutations and random inputs, at least 2^{107} cases. Rule-based fuzzing generates at least $2^3 + 2^{32} + 2^4 + 2^4 + 2^{64}$ cases. However, our proposed formal guided fuzzing only needs 3×5 cases.

4) **AS Security Mode command bit-level fuzzing:** To verify the formal assumptions, AS Security Mode command bit-level fuzzing is necessary. Similar to NAS Security Mode command, AS Security Mode command contains 4 bits of EEA1, 4 bits of EIA1, and 8 octets of MAC-I. Like illustrated in NAS Security Mode command bit-level fuzzing, formal guided fuzzing generates 3×3 cases. In contrast, brute force fuzzing generates at least 2^{72} cases, and rule-based fuzzing generates $2^4 + 2^4 + 2^{64}$ cases.

Fig. 14 provides an intuitive visualization that compares the effectiveness of different fuzzing strategies. The upper and lower bounds of the pink area are represented by values of "c=1" and "c=4" in Fig. 14. Notably, it is evident that brute force fuzzing and rule-based fuzzing exhibit exponential growth patterns. In contrast, our proposed formal guided fuzzing approach demonstrates linear growth, requiring considerably less computational power for vulnerability verification and localization. The superiority of our method in terms of efficiency and scalability enables a realistic testing and vulnerability detection across the entire specifications, and provides the assurance and confidence in 5G system, especially when applied to the critical infrastructures.

C. SYSTEM ASSESSMENT OF COMPUTATION COMPLEXITY IN FORMAL GUIDED COMMAND-LEVEL FUZZING

In addition to detecting vulnerabilities at the bit level of a command using fuzzing, it is also necessary to verify formal attack traces through command-level fuzzing. Unlike bit-level fuzzing, where no single representative case can cover all out-of-rule scenarios, command-level fuzzing potentially involves an unlimited number of cases. To efficiently locate command-level vulnerabilities, we proposed a probability-based command-level fuzzing framework in our previous work [26]. Utilizing formal assumptions of RRC and User Identity Disclosure attack, we fixed the C-RNTI and IMSI on the srsRAN platform to simulate user identity disclosure. This strategy reduced the number of fuzzing cases to 3,080. Furthermore, leveraging the identity disclosure assumption, we collected all different commands on downlink channels and fuzzed all possible permutations. As illustrated in Fig. 15, our proposed probability-based framework requires only 36.5% of the number of fuzzing cases compared to a random fuzzing strategy. The figure also shows the number of cases needed to detect different percentages of vulnerabilities. In comparison to the conventional linear growth, computation-consuming random fuzzing strategy, our developed probability-based fuzzing approach demonstrates significantly improved performance [26]. The incorporation of prior knowledge further enhances the effectiveness of our method, leading to even greater efficiency gains. Theoretically, our proposed approach has the potential to complete millions of command-level fuzzing iterations within a modest scope of five thousand test cases. This significant reduction in the number of required test cases underscores the efficiency and effectiveness of our methodology.

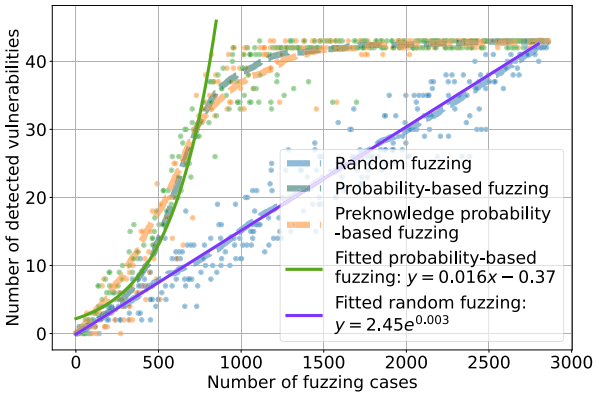


FIGURE 15. Comparison of Benchmark random-based fuzzing and probability-based fuzzing [26].

D. COMPARATIVE ANALYSIS WITH EXISTING METHODS

After evaluating the performance of our proposed system, we create a table that examines our system and existing methods in various aspects, including effectiveness, efficiency, and coverage. Table 4 offers a comprehensive comparative analysis with existing methods. Table 4 becomes clear that our system displays similar time complexity to formal verification and provides similar effectiveness to other existing fuzz testing approaches.

E. DISCUSSION OF INTEROPERABILITY IN REAL-WORLD IMPLEMENTATION

Serving as critical infrastructure applied across various verticals, ensuring interoperability among different vendors implementing 5G and beyond technologies, while maintaining security standards, is crucial. In our paper, we presented a multi-dimensional, multi-layer protocol-independent fuzzing framework aimed at detecting vulnerabilities and unintended emergent behaviors in the rapidly evolving specifications of 5G and NextG, as well as large-scale open programmable 5G stacks. This framework is designed with the diversity of vendor implementations in mind and is dedicated to extensive testing across these variations.

Formal-model-based fuzz testing, which explores random variations of a scenario guided by formal constraints, is used for unintended emergent behavior assessment. It performs fuzz testing in the out-of-assumption domains of formal models on a pilot stack (e.g., srsRAN). The test cases and models offer automation and scalability when applied to other open-source 5G software available from the Linux Foundation and existing 5G codebases from carriers in national operation. In the testbed that we developed, HyFuzz, we mixed srsRAN and Amarisoft as the vendors for validation and proved the adaptation of it.

In implementation, we further address the concern for interoperability by creating a positive feedback loop between

TABLE 4. Comparative analysis with existing methods.

Model	Effectiveness	Efficiency (Time complexity $\mathcal{O}$) (n: the number of commands/ the length of command)	Coverage
5GReasoner [3]	Simulation Protocol independent	$\mathcal{O}(c)$	Single-step/ Multi-steps
Mutation-based Fuzz Testing [19]	Mutl-platforms TCP	$\mathcal{O}(n2)$	Single-step
LZFuzz [24]	Mutl-platforms ICMP, HTTP	$\mathcal{O}(nlog(n))$	Single-step
Ours	Multi-platforms (simulation and over-the-air platform) Protocol independent	$\mathcal{O}(c)$	Single-step/ Mutli-steps

formal verification and fuzz testing within our framework. This means that as vulnerabilities or interoperability issues are discovered, they are fed back into the testing cycle, allowing for iterative refinement and adaptation to a wide range of vendor implementations and scenarios. This approach aims to strike a balance between rigorous security standards and the practical need for interoperability across different 5G technologies.

In practice, implementing fuzz testing and formal analysis for 5G vulnerability detection faces specific challenges: formal verification might overlook practical issues, and fuzz testing requires extensive resources. Our method uses formal verification for initial high-risk vulnerability identification, guiding subsequent, more expansive fuzz testing. This strategy enhances precision and coverage of 5G security assessments. The presented approach could be applied to various fifth-generation (5G) and beyond open programmable platforms [38], [39], [40] or other cognitive/software-defined communication systems [41]

X. CONCLUSION

Motivated by the limitations of state-of-the-art vulnerability detection methods, which are highly computationally complex in fuzz testing and labor-intensive formal verification, we present a first-of-its-kind formal guided fuzz testing approach in this paper for efficient and systematic 5G vulnerability detection. Specifically, formal verification is implemented in our proposed approach to detect attack traces in 5G protocols, guiding subsequent fuzz testing. Formal verification is employed to detect 4 attack models, and then fuzz testing, following the detected high risk area by formal verification, detects 61 vulnerabilities in the 5G NSA authentication and authorization procedure. All detected vulnerabilities, which includes both exploits discussed in existing research and new findings that have not been previously revealed, are verified via real-life experiments using srsRAN. Our approach combines the strengths and coverage of formal and fuzzing methods to efficiently detect vulnerabilities across protocol logic and implementation stacks in a hierarchical and interactive manner. To close the loop, feedback from detected attack models and vulnerabilities is incorporated to fortify system designs and enhance resilience.

To provide an intuitive reflection in addressing the computation complexity, we assess the complexity of our approach with conventional fuzz testing results and the state-of-the-art rule-based approaches. Conventional fuzz testing would necessitate a staggering 9×10^{77} fuzzing cases. However, under the formal assumption of RRC and User Identity Disclosure attack, our framework reduces the number of fuzzing cases to a manageable 3080, which is further curtailed to 1027 through a probability-based fuzzing strategy, showcasing the framework's superior efficiency. Beside the computation complexity, one significant advantage of our approach is its scalability. Our approach can be applied in diverse contexts by not being limited to a single protocol and accommodating varying needs and requirements across different systems and infrastructures. This adaptability broadens its applicability, making it a valuable asset for researchers addressing cybersecurity challenges in various domains. Additionally, our protocol-independent approach promotes interoperability and compatibility with existing infrastructure cybersecurity solutions. It seamlessly integrates with different protocols and systems, simplifying implementation and facilitating the adoption of enhanced security measures. This promotion, in turn, contributes to a more resilient and secure cyber ecosystem, benefiting both individual researchers and the broader community through enhanced collaboration and knowledge-sharing.

XI. FUTURE WORK

In this paper, we emphasize the authentication process is its crucial role in wireless communication. A secure authentication process is essential for protecting subsequent communications from potential threats, such as information leakage or connection failures. By ensuring a strong authentication mechanism, we establish a solid foundation for secure communication.

However, it is worth noting that post-authentication communication is often most susceptible to efficiency-related challenges, such as Signal-to-Noise Ratio (SNR) issues. We plan to expand our efforts by developing more comprehensive measurement approaches, including factors like SNR and latency. This expansion will enable us to apply our framework to broader areas of 5G and beyond systems, including data-link layers.

Furthermore, we plan to investigate how our approach can be adapted to address emerging technologies and standards evolving from 5G to 6G and beyond. This includes looking into the integration of machine learning algorithms for predictive analytics, the application in IoT environments, and the enhancement of network slicing and edge computing capabilities.

In addition, the limitation of our proposed approach lies in the labor-intensive nature of formal verification. While formal verification is a powerful tool for detecting vulnerabilities, developing formal models and proofs often requires significant expertise and manual effort. This can be time-consuming and may not scale well, particularly for complex protocols or large-scale systems. To address this limitation, our research will focus on overcoming the labor-intensive nature of formal verification by leveraging advanced technologies such as Large Language Models (LLMs) like GPT or LLAMA. Our goal is to automate the generation of initialized formal code, making formal verification more accessible and efficient. By achieving this automation, we aim to develop a comprehensive automated system for detecting vulnerabilities in network protocols. This system can autonomously identify vulnerabilities when provided with protocol specifications as only inputs. Additionally, the new model will consider a wider variety of data, like log files and the state of the cache, to enable multi-dimensional input and analysis. In addition to 5G specifications, we will expand the verification and vulnerability detection to various specifications and implementations, including IoT and other areas.

ACKNOWLEDGMENT

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

REFERENCES

[1] J. Alcaraz-Calero, I.-P. Belikaidis, C. J. B. Cano, P. Bisson, D. Bourse, M. Bredel, D. Camps-Mur, T. Chen, X. Costa-Perez, P. Demestichas, M. Doll, S. E. Elayoubi, A. Georgakopoulos, A. Mämmelä, H.-P. Mayer, M. Payaro, B. Sayadi, M. S. Siddiqui, M. Tercero, and Q. Wang, "Leading innovations towards 5G: Europe's perspective in 5G infrastructure public-private partnership (5G-PPP)," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–5.

[2] M. Shatnawi, H. Altaleb, and R. Zoltán, "The digital revolution with NESAS assessment and evaluation," in *Proc. IEEE 10th Jubilee Int. Conf. Comput. Cybern. Cyber-Medical Syst. (ICCC)*, Jul. 2022, pp. 000099–000104.

[3] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino, "5GReasoner: A property-directed security and privacy analysis framework for 5G cellular network protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 669–684.

[4] G. Klees, A. Ruef, B. Cooper, S. Wei, and M. Hicks, "Evaluating fuzz testing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 2123–2138.

[5] A. Souri and M. Norouzi, "A state-of-the-art survey on formal verification of the Internet of Things applications," *J. Service Sci. Res.*, vol. 11, no. 1, pp. 47–67, Jun. 2019.

[6] C. Beaman, M. Redbourne, J. D. Mummery, and S. Hakak, "Fuzzing vulnerability discovery techniques: Survey, challenges and future directions," *Comput. Secur.*, vol. 120, Sep. 2022, Art. no. 102813.

[7] Y. Wang, A. Gorski, and L. A. DaSilva, "AI-powered real-time channel awareness and 5G NR radio access network scheduling optimization," in *Proc. 17th Int. Conf. Des. Reliable Commun. Netw.*, 2021, pp. 1–7.

[8] Y. Wang, S. Jere, S. Banerjee, L. Liu, S. Shetty, and S. Dayekh, "Anonymous jamming detection in 5G with Bayesian network model based inference analysis," in *Proc. IEEE 23rd Int. Conf. High Perform. Switching Routing (HPSR)*, Jun. 2022, pp. 151–156.

[9] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *Proc. 25th Int. Conf., (CAV)*, vol. 8044, 2013, pp. 696–701.

[10] C. Cremers and M. Dehnel-Wild, *Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion*. Reston, VA, USA: Internet Society, 2019.

[11] A. Peltonen, R. Sasse, and D. Basin, "A comprehensive formal analysis of 5G handover," in *Proc. 14th ACM Conf. Secur. Privacy Wireless Mobile Netw.*, Jun. 2021, pp. 1–12.

[12] M. Labib, V. Marojevic, J. H. Reed, and A. I. Zaghloul, "Enhancing the robustness of LTE systems: Analysis and evolution of the cell selection process," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 208–215, Feb. 2017.

[13] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking LTE on layer two," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 1121–1136.

[14] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert, *Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems*. Reston, VA, USA: Internet Society, 2017.

[15] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5G authentication," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 1383–1396.

[16] H. Wang, B. Cui, W. Yang, J. Cui, L. Su, and L. Sun, "An automated vulnerability detection method for the 5G RRC protocol based on fuzzing," in *Proc. 4th Int. Conf. Adv. Comput. Technol. Inf. Sci. Commun. (CTISC)*, Apr. 2022, pp. 1–7.

[17] F. He, W. Yang, B. Cui, and J. Cui, "Intelligent fuzzing algorithm for 5G NAS protocol based on predefined rules," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2022, pp. 1–7.

[18] L. J. Moukahal, M. Zulkernine, and M. Soukup, "Vulnerability-oriented fuzz testing for connected autonomous vehicle systems," *IEEE Trans. Rel.*, vol. 70, no. 4, pp. 1422–1437, Dec. 2021.

[19] X. Han, Q. Wen, and Z. Zhang, "A mutation-based fuzz testing approach for network protocol vulnerability detection," in *Proc. 2nd Int. Conf. Comput. Sci. Netw. Technol.*, Dec. 2012, pp. 1018–1022.

[20] Z. Salazar, H. N. Nguyen, W. Mallouli, A. R. Cavalli, and E. Montes de Oca, "5Greplay: A 5G network traffic fuzzer–application to attack injection," in *Proc. 16th Int. Conf. Availability, Rel. Secur.*, Aug. 2021, pp. 1–8.

[21] S. Sheikhi, E. Kim, P. S. Duggirala, and S. Bak, "Coverage-guided fuzz testing for cyber-physical systems," in *Proc. ACM/IEEE 13th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, May 2022, pp. 24–33.

[22] M. Ammann, L. Hirschi, and S. Kremer, "Dy fuzzing: Formal dolev-yao models meet protocol fuzz testing," *Cryptol. ePrint Arch.*, 2023.

[23] R. Ma, S. Ren, K. Ma, C. Hu, and J. Xue, "Semi-valid fuzz testing case generation for stateful network protocol," *Tsinghua Sci. Technol.*, vol. 22, no. 5, pp. 458–468, Sep. 2017.

[24] S. Bratus, A. Hansen, and A. Shubina, "Lzfuzz: A fast compression-based fuzzer for poorly documented protocols," Tech. Rep., 2008.

[25] N. Osborne and C. Pascutto, "Leveraging formal specifications to generate fuzzing suites," in *Proc. OCaml Users Developers Workshop, Co-Located 26th ACM SIGPLAN Int. Conf. Funct. Program.*, 2021, pp. 1–3.

[26] J. Yang, Y. Wang, Y. Pan, and T. X. Tran, "Systematic meets unintended: Prior knowledge adaptive 5G vulnerability detection via multi-fuzzing," 2023, *arXiv:2305.08039*.

[27] J. Yang, Y. Wang, T. X. Tran, and Y. Pan, "5G RRC protocol and stack vulnerabilities detection via listen-and-learn," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2023, pp. 236–241.

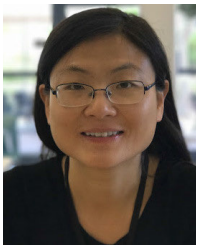
[28] D. Dauphinais, M. Zylka, H. Spahic, F. Shaik, J. Yang, I. Cruz, J. Gibson, and Y. Wang, "Automated vulnerability testing and detection digital twin framework for 5G systems," in *Proc. IEEE 9th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2023, pp. 308–310.

[29] M. Krichen, M. Lahami, O. Cheikhrouhou, R. Alroobaea, and A. J. Maâlej, "Security testing of Internet of Things for smart city applications: A formal approach," *Smart Infrastruct. Appl., Found., Smarter Cities Societies*, pp. 629–653, 2020.

[30] J. Yang, Y. Wang, Y. Pan, and T. X. Tran, "Systematic and scalable vulnerability detection for 5G specifications and implementations," *IEEE J. Sel. Areas Commun.*, 2023.
[31] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. 10th ACM Int. Workshop Wireless Netw. Testbeds, Experim. Eval., Characterization*, Oct. 2016, pp. 25–32.
[32] J. Yang and Y. Wang, "A nextg hybrid testing platform for multi-step deep fuzzing and performance assessment from virtualization to over-the-air," *IEEE Cloudnet*, 2023.
[33] J. B. Bou Abdo, H. Chaouchi, and M. Aoude, "Ensured confidentiality authentication and key agreement protocol for EPS," in *Proc. Symp. Broadband Netw. Fast Internet (RELABIRA)*, May 2012, pp. 73–77.
[34] 3GPP, *Universal Mobile Telecommunications System (UMTS); LTE; Mobility Management Entity (MME) Visitor Location Register (VLR) SGs Interface Specification*, Version 8.5.0 Standard TS 29.118, 3rd Gener. Partnership Project (3GPP), 2015.
[35] M. Khan, P. Ginzboorg, K. Järvinen, and V. Niemi, "Defeating the downgrade attack on identity privacy in 5G," in *Proc. Int. Conf. Res. Secur. Standardisation*. Springer, 2018, pp. 95–119.
[36] J.-K. Tsay and S. F. Mjølsnes, "A vulnerability in the umts and LTE authentication and key agreement protocols," in *Proc. Comput. Netw. Secur., 6th Int. Conf. Math. Methods, Models Architectures Comput. Netw. Secur., MMM-ACNS*. Springer, 2012, pp. 65–76.
[37] M. T. Raza, F. M. Anwar, and S. Lu, "Exposing lte security weaknesses at protocol inter-layer, and inter-radio interactions," in *Proc. Secur. Privacy Commun. Netw. 13th Int. Conf.*, Niagara Falls, ON, Canada. Springer, 2017, pp. 312–338.
[38] *O-RAN: Towards an Open and Smart RAN*, O-RAN Alliance, Oct. 2018.
[39] *SrsRAN is a 4G/5G Software Radio Suite Developed By SRS*, Softw. Radio Syst., Dublin, Dublin, Ireland, 2021.
[40] Y. Wang, A. Gorski, and A. P. da Silva, "Development of a data-driven mobile 5G testbed: Platform for experimental research," in *Proc. IEEE Int. Medit. Conf. Commun. Netw. (MeditCom)*, Sep. 2021, pp. 324–329.
[41] Y. Wang and C. W. Bostian, "Dynamic cellular cognitive system," U.S. Patent 8 610 094, Jan. 10, 2012.

JINGDA YANG (Graduate Student Member, IEEE) received the B.E. degree in software engineering from Shandong University and the M.Sc. degree in computer science from The George Washington University. He is currently pursuing the Ph.D. degree with the School of System and Enterprises, Stevens Institute of Technology. His research interests include formal verification and vulnerability detection of wireless protocol in 5G.

SUDHANSHU ARYA (Member, IEEE) received the M.Tech. degree in communications and networks from the National Institute of Technology, Rourkela, India, in 2017, and the Ph.D. degree from Pukyong National University, Busan, South Korea, in 2022. He was a Research Fellow with the Department of Artificial Intelligence Convergence, Pukyong National University. He is currently a Research Fellow with the School of System and Enterprises, Stevens Institute of Technology, Hoboken, NJ, USA. His research interests include wireless communications and digital signal processing, with a focus on free-space optical communications, optical scattering communications, optical spectrum sensing, computational game theory, and artificial intelligence. He received the Best Paper Award from ICGHIT 2018 and the Early Career Researcher Award from Pukyong National University, in 2020.



YING WANG (Member, IEEE) received the B.E. degree in information engineering from Beijing University of Posts and Telecommunications, the M.S. degree in electrical engineering from the University of Cincinnati, and the Ph.D. degree in electrical engineering from Virginia Polytechnic Institute and State University. She is currently an Associate Professor with the School of System and Enterprises, Stevens Institute of Technology. Her research interests include cybersecurity, wireless AI, edge computing, health informatics, and software engineering.