Jamming Smarter, Not Harder: Exploiting O-RAN Y1 RAN Analytics for Efficient Interference

Abiodun Ganiyu[1], Dara Ron[1], Syed Rafiul Hussain[2], and Vijay K Shah[1]
[1]NextG Wireless Lab, North Carolina State University, Raleigh, USA
[2]Computer Science and Engineering, The Pennsylvania State University, Pennsylvania, USA
{aganiyu, dron, vijay.shah}@ncsu.edu, hussain1@psu.edu

***Abstract*—The Y1 interface in O-RAN enables the sharing of RAN Analytics Information (RAI) between the near-RT RIC and authorized Y1 consumers, which may be internal applications within the operator's trusted domain or external systems accessing data through a secure exposure function. While this visibility enhances network optimization and enables advanced services, it also introduces a potential security risk – a malicious or compromised Y1 consumer could misuse analytics to facilitate targeted interference. In this work, we demonstrate how an adversary can exploit the Y1 interface to launch selective jamming attacks by passively monitoring downlink metrics. We propose and evaluate two Y1-aided jamming strategies: a clustering-based jammer leveraging DBSCAN for traffic profiling and a threshold-based jammer. These are compared against two baselines strategies – always-on jammer and random jammer – on an over-the-air LTE/5G O-RAN testbed. Experimental results show that in unconstrained jamming budget scenarios, the threshold-based jammer can closely replicate the disruption caused by always-on jamming while reducing transmission time by 27%. Under constrained jamming budgets, the clustering-based jammer proves most effective, causing up to an 18.1% bitrate drop while remaining active only 25% of the time. These findings reveal a critical trade-off between jamming stealthiness and efficiency, and illustrate how exposure of RAN analytics via the Y1 interface can enable highly targeted, low-overhead attacks, raising important security considerations for both civilian and mission-critical O-RAN deployments.**

***Index Terms*—Stealthy Jamming attacks, RAN Analytics Information, Y1, O-RAN, 5G**

I. Introduction

The Open Radio Access Network (O-RAN) paradigm is reshaping the telecommunications industry by promoting openness, modularity, and vendor interoperability. Through standardized interfaces and disaggregated components, O-RAN enables flexible integration of third-party applications and intelligent control mechanisms across the RAN [1]. This architectural shift supports AI-driven network automation, fine-grained optimization, and cost-effective network customization, capabilities critical to commercial networks and tactical and defense communications systems. The architecture's support for low-latency control, scalable radio coordination, and dynamic analytics enables future battlefield networks to operate with enhanced situational awareness, rapid network reconfiguration, and resilient command-and-control.

The relevance of O-RAN to military communications is gaining traction. Defense agencies and researchers are increasingly exploring how O-RAN's software-defined architecture can enable resilient, secure, and mission-adaptive networks [2]. The Department of Defense (DoD) and other allied institutions have identified the potential for dual-use platforms, where commercial-grade O-RAN systems can be adapted to meet military-grade requirements such as hardened security, real-time responsiveness, and contested spectrum operation [3]. In particular, O-RAN's support for intelligent RAN analytics, distributed control, and programmable policy enforcement aligns well with operational needs on future battlefields.

The O-RAN ALLIANCE continues to evolve this vision by releasing specifications for key interfaces such as the O-fronthaul, F1, E2, A1, and O1, which standardize connectivity across RAN components and intelligent controllers [4]. One notable development is the Y1 interface, introduced by the O-RAN ALLIANCE Working Group 3 [5]. The Y1 interface enables the exposure of near-real-time RAN Analytics Information (RAI) from a Y1 Producer (co-located with the near-RT RIC) to authorized Y1 consumers which may be internal applications within the operator's trusted domain or external systems accessing data through a secure exposure function. This interface is envisioned to enable a wide range of third-party analytics services, such as traffic orchestration, V2X coordination, and mission-critical awareness platforms, by delivering near-real-time metrics from the RAN. These capabilities are especially relevant in defense and emergency communications, where accurate and timely RAN insights may drive adaptive spectrum use, network resilience, and coordinated mobility. However, this increased visibility also introduces new attack vectors. A compromised or malicious Y1 consumer could misuse exposed analytics to infer network behavior and strategically disrupt critical operations.

Motivated by this threat, our work investigates the feasibility of RAN analytics-driven efficient interference through the Y1 interface. We demonstrate how an adversary, operating as an authenticated but malicious Y1 consumer, can access RAN metrics streamed over Y1 to infer traffic patterns and coordinate targeted jamming attacks. Specifically, the Y1 consumer forwards real-time analytics to an external software-defined (or GNU Radio) based jammer, which then selectively activates interference based on observed traffic conditions. By leveraging unsupervised learning (via clustering technique) to profile traffic, the jammer intelligently allocates limited transmission resources to maximize disruption, a tactic particularly relevant for covert or energy-constrained military operations.

While prior efforts have explored various machine learning (ML) strategies for O-RAN security enhancement [6]–[9], this work highlights a lesser-discussed threat: *the use of exposed RAN analytics as a side channel for targeted radio interference*. We implement our attack on an over-the-air testbed and compare three strategies, always-on, random, and traffic-aware jamming, demonstrating how analytics-guided attacks can achieve comparable disruption with significantly less transmission effort. Our findings underscore the urgent need to consider threat models involving analytics exposure, especially in sensitive deployments such as defense and tactical communications, where system compromise or data leakage may have high operational costs.

The key contributions of this work are outlined as follows.

• We demonstrate how a threat actor can exploit the O-RAN Y1 interface to passively monitor RAN Analytics Information (RAI) and gain situational awareness of network activity. Using this visibility, we illustrate the feasibility of selective jamming attacks driven by unsupervised traffic profiling using DBSCAN, enabling context-aware interference without requiring privileged access to the RAN itself.

• We develop a complete O-RAN testbed integrating a compliant Y1 interface, srsRAN-based RAN/Core systems, and GNU Radio based jamming implementations. Our threat model emulates a malicious Y1 consumer relaying RAI to a jammer, which dynamically adjusts its activity based on learned traffic patterns and jamming constraints.

• We conduct two complementary experiments: one under unlimited jamming budget using a fixed traffic pattern, and another under constrained jamming budgets with multi-rate traffic. Results show that threshold-based and DBSCAN-guided jammers can closely mirror the disruptive effect of always-on jamming with significantly less transmission time, and that targeting high-throughput traffic yields the greatest impact per unit of jamming effort.

II. Related Works

As O-RAN adoption grows, the security implications of its open and modular design have drawn increasing attention from both academia and industry. A prominent concern is the expanded attack surface introduced by programmable RAN controllers and third-party applications (xApps), particularly within the Near-real-time RIC environment.

Several prior works have focused on enhancing O-RAN security through machine learning (ML)-based xApps. For example, [6] presents an ML-driven connection management xApp capable of both launching and defending against malicious behavior. A federated deep reinforcement learning (FDRL) framework for detecting jamming attacks is proposed in [7], while other works explore DRL-based resource allocation strategies to counter adversarial interference [8], [9]. Additional approaches include classifier xApps that identify and mitigate jamming by classifying interference types [10], intrusion detection mechanisms for threat identification and mitigation [11], and anomaly detection systems that protect the near-RT RIC from attacks targeting the E2 interface [12]. These studies primarily address adversarial behavior on internal O-RAN interfaces (e.g., E2, A1, and F1), focusing on attacks and defenses involving internal control loops or xApps. To the best of our knowledge, no existing work has explored attack scenarios involving an untrusted or malicious Y1 consumer, a new telemetry exposure point in the O-RAN architecture that grants visibility into real-time RAN analytics.

Beyond direct attack mitigation, recent efforts have examined the security of third-party components in O-RAN systems. For example, Atalay et al. [13] introduce the xApp Repository Function (XRF) to address authentication and authorization challenges when integrating external xApps. Similarly, the work in [14] explores adversarial activity via the Rogue Cell attack, where malicious operators manipulate RAN telemetry to mislead traffic steering decisions. Their proposed attack (APATE) demonstrates how such manipulation can lead to unfair resource allocation, while their LSTM-based detection mechanism (MARRS) offers a potential countermeasure. In parallel, other studies have explored jamming strategies in wireless networks. One such effort presents a systematic learning method for optimal jamming, using reinforcement learning to adapt jamming behavior based on channel conditions [15].

In contrast to these efforts, our work is the first to investigate how a compliant yet malicious Y1 consumer can be exploited to coordinate external analytics-guided jamming. We demonstrate how passive access to RAN Analytics Information (RAI) without breaching the RAN or modifying internal xApps can enable an attacker to infer traffic behavior and drive targeted interference via a coordinated external jammer. This expands the O-RAN threat model to account for telemetry leakage and adversarial use of exposed interfaces, particularly under the evolving Y1 specification.

III. O-RAN Background

We briefly outline the key components of the O-RAN architecture relevant to our work. Readers should refer to [1] for a detailed understanding of O-RAN architecture.

A. RAN Intelligent Controller (RIC)

The RAN Intelligent Controller (RIC) is a central component introduced in O-RAN to support programmability and intelligent control of the radio access network. It comprises two logical components based on operational time scales: the near-real-time RIC and the non-real-time RIC.

• **Near-Real-Time RIC:** Operates within 10 ms to 1 s timescales. It hosts xApps that perform latency-sensitive control and optimization tasks, such as handover management or interference mitigation. It communicates with the underlying RAN nodes via the E2 interface and supports telemetry gathering and control actuation.

• **Non-Real-Time RIC:** Operates on a time scale of greater than 1 s and up to minutes, handling long-term network optimization tasks, such as policy generation and model training, hosted within the Service Management and Orchestration

(SMO) framework. It connects to the near-RT RIC via the A1 interface and to other O-RAN components, such as the O-DU, through the O1 interface.

B. RIC Database and Shared Data Layer

The RIC database serves as a centralized database repository that stores various forms of RAN state information, including Key Performance Measurements (KPMs), RAN events, and custom cell-level and UE-level analytics. xApps running on the near-real-time RIC can read from and write to this database to support data-driven decision-making.

Access to the RIC database is mediated through the Shared Data Layer (SDL), which provides a standardized API for storing and retrieving analytics data. The SDL, as implemented by the O-RAN Software Community (OSC) [16], interfaces with a Redis backend and exposes Redis-compatible functions through a controlled API surface. This abstraction allows xApps to interact with the database using familiar key-value operations without direct access to the database internals.

Importantly, the Y1 Producer also interacts with the SDL to access stored analytics data. When a Y1 Consumer subscribes to specific RAN metrics or analytics, the producer queries the SDL and constructs a response conforming to the Y1 specification, ensuring only authorized and subscribed consumers receive the requested analytics.

C. Y1 Interface

The Y1 interface supports two modes of RAI access: (1) subscription-based notification, where the Y1 Consumer registers interest in specific analytics and receives asynchronous updates; and (2) query-based retrieval, where the Y1 Consumer explicitly requests specific analytics on demand. These operations are defined through three core APIs: 1) *RAI_Subscribe* initiates a subscription to a particular RAI type and target; 2) *RAI_Notify* delivers the analytics payload based on the notification method (periodic or event-driven); and 3) *RAI_Query* responds with RAI data upon request.

The analytics content, as defined under the "RAN performance analytics" type in the Y1 specification, includes a set of parameters such as average RLC throughput, downlink latency, packet loss rate, and their distributions. Importantly, while the specification allows analytics to be per-UE or slice-specific, in this study, we scope our use of RAN analytics to aggregated cell-level metrics to avoid user privacy concerns and ensure generality.

Internally, the Y1 Producer, which resides within the Near-RT RIC, sources these analytics from the RIC Database via the SDL. It then filters and encodes the data based on the subscription query or request filter, returning only the subset of attributes as requested by the Y1 Consumer. The protocol stack for Y1 follows REST over HTTPS (as per the O-RAN specification [5]), utilizing JSON for data interchange. Mandatory security controls, including mutual TLS (mTLS) for authentication and authorization, are enforced to restrict access only to verified and permitted consumers. However, the

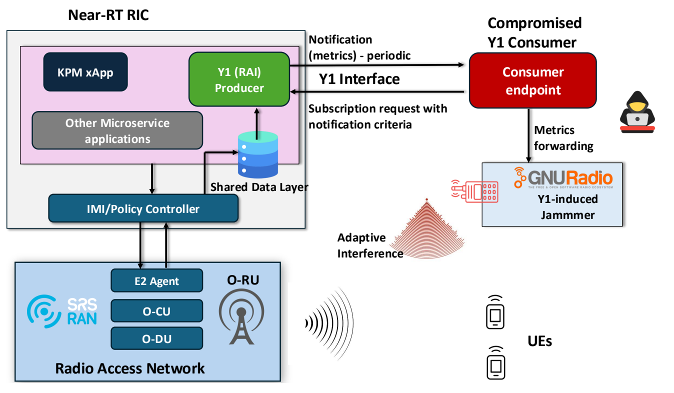

Fig. 1. Threat model. A malicious Y1 Consumer leverages legitimate access to RAN analytics to coordinate adaptive jamming against the RAN.

specification does not define mechanisms for fine-grained auditing or behavioral analysis of consumers post-authentication, leaving room for potential misuse.

This paper focuses on the RAI_Subscription operation with periodic notification trigger, which allows the attacker-controlled consumer to receive a continuous stream of analytics updates at configured intervals (e.g., every 1s or 5s). The flexibility of this subscription mechanism, while beneficial for analytics-driven services, introduces a stealthy reconnaissance channel if abused by malicious or compromised entities.

IV. Threat and System Model

A. Threat Model

As shown in Fig. 1, the RAN exposes telemetry data (e.g., KPMs) to the near-RT RIC via the E2 interface. These metrics are processed by internal xApps (e.g., KPM xApp) and stored within a central RIC Database. The RAI Producer, implemented in compliance with the Y1 interface specification, exposes these analytics to authorized Y1 consumers based on defined subscription criteria. Notifications may be event-triggered or periodic, and are securely exchanged using mutual TLS (mTLS) authentication. The attacker does not attempt to compromise the RIC or RAN infrastructure directly but instead leverages their authorized access to RAN analytics to facilitate external interference operations. Specifically, the Y1 Consumer, while appearing benign and authenticated, is compromised or operated by a malicious insider who forwards the received RAN analytics to a coordinated external jammer. This leads to a novel form of *analytics-driven jamming*, where the interference is intelligently triggered based on accurately observed near-real-time RAN performance data.

The attack model assumes: *1. The Y1 Consumer has valid credentials and is able to establish a mutual TLS connection with the Y1 Producer; 2. The analytics shared are compliant with the consumer's subscription and the producer's export policies; 3. The jammer is physically decoupled from the RIC but receives continuous analytics from the compromised consumer.* This threat highlights the risks associated with exposing aggregated RAN telemetry to third-party consumers,

TABLE I
IMPLEMENTED Y1 INTERFACE API SUMMARY

Endpoint	Method	Description
`/subscriptions/subscribe`	POST	Registers a Consumer to receive RAN analytics
`/subscriptions/unsubscribe`	DELETE	Cancels an existing subscription
`/subscriptions/<id>`	PUT	Updates an existing subscription (partial support)
`/notify` (Consumer-side)	POST	Receives metrics from the Producer

even when such access is governed by standard authentication mechanisms. It underscores a critical tension between openness for innovation and the potential for misuse of authorized visibility within the O-RAN architecture.

In our threat scenario, the Y1 consumer is compromised. While still operating within its access limits (for example, receiving only authorized RAN analytics), it forwards the metrics received to an external malicious agent, a controller responsible for managing a physical jammer. The jammer uses this analytics stream to dynamically adapt its interference behavior to degrade network performance in a stealthy and energy-efficient manner.

B. Y1 Interface Implementation

To emulate a real-world compliant environment, we implemented the Y1 interface as defined in the O-RAN ALLIANCE Y1 specifications (Y1TD, Y1AP, Y1GAP). Our design includes both the Y1 Producer (within the Near-RT RIC) and the Y1 Consumer, ensuring mutual Transport Layer Security (mTLS)-based authentication and structured data exchange [17]. Since the current open-source near-RT RIC platforms, including the O-RAN Software Community (OSC) implementation, do not yet support the Y1 interface, we developed a custom lightweight Y1 Producer using Flask in Python. This component retrieves analytics from a Redis-backed SDL and exposes the RAI to authenticated consumers over a secure mTLS channel. To enable integration with the near-RT RIC platform, we containerized the Y1 Producer as a Docker image which can be deployed as a Kubernetes service within the RIC environment. This mirrors real-world deployment practices and ensures seamless communication with other RIC platform components. The Y1 Consumer, also implemented as a Flask web service, served as the interface through which RAI was forwarded to an external jammer in our threat model.

1) Producer API Design: The Producer exposes a secure RESTful interface, including a subscription endpoint at `/Y1_RAI_Subscriptions/v1/subscriptions/subscribe`, through which authenticated Consumers can submit subscription requests. Each request includes:

• `raiType` and `raiTypeVersion`: defining the category of analytics,

• `notificationCriteria`: specifying the trigger mechanism (periodic or event-based) and the interval,

• `notificationTargetAddress`: indicating the Consumer endpoint for metric delivery.

Upon subscription, the Producer retrieves analytics from the Redis-backed SDL and transmits them to the Consumer based on the subscription parameters. The metrics are JSON-encoded and include; `subscription_id`, `rai_content`, `timestamp`, and `validity_period`.

TABLE II
EXTRACTED RAN ANALYTICS METRICS FOR Y1 INTERFACE

Metric	Description
DL CQI	Average Channel Quality Indicator from UEs, indicating link quality.
DL MCS	Average Modulation and Coding Scheme used for downlink scheduling.
DL Bitrate	Total downlink throughput (in bits/sec) across all active UEs.
DL BLER	Downlink Block Error Rate, computed from transmission error statistics.
DL Latency	Average latency (in seconds) from the PDCP layer.
DL Bytes	Total number of acknowledged downlink bytes from PDCP.
PCI	Physical Cell Identifier of the transmitting cell.
Carrier ID	Logical carrier index, useful in multi-carrier environments.
Number of RACH	Count of random access procedure attempts (Msg1 preambles).

2) Consumer API Implementation: The Y1 Consumer is implemented as a secure Flask-based HTTPS service that initiates a subscription to the Y1 Producer using mTLS with X.509 certificate-based authentication, in compliance with O-RAN security guidelines. Upon successful subscription, the Consumer exposes a notification endpoint that periodically receives the RAI pushed by the Producer. All incoming RAI messages are parsed and stored in memory for subsequent use. To emulate an adversarial scenario, the Consumer includes a malicious forwarding module which establishes a TCP socket with the jammer system and continuously streams the latest received RAI metrics.

We leverage the KPM xApp within the Near-RT RIC to collect performance metrics from the RAN over the E2 interface. These metrics are aggregated periodically and stored in a Redis database, where they are accessed by the Y1 Producer to subsequently serve subscribed Y1 consumers. Table II summarizes the nine metrics extracted and their relevance to RAN behavior and performance.

V. Y1-AIDED JAMMING STRATEGIES

A. Y1-Threshold Jammer

The Y1-Threshold jammer is a lightweight yet effective strategy that reacts to real-time RAN metrics streamed via the Y1 interface. Specifically, it applies a simple thresholding rule on selected features (e.g., CQI, bitrate, or BLER) to decide when to transmit. This design is particularly useful for constrained systems where compute or memory overhead must be minimized. Let $\mathbf{x}_t = \{\mathrm{CQI}_t, \mathrm{MCS}_t, \mathrm{Bitrate}_t, \mathrm{BLER}_t\}$ represent the observed downlink analytics vector at time t. The jammer compares selected features against a threshold:

$$\text{JAM} \quad \text{if } \mathrm{Bitrate}_t \geq \theta; \quad \text{NO JAM} \quad \text{otherwise}, \tag{1}$$

where θ is a predefined threshold selected based on offline traffic profiling. For example, if the traffic is expected to operate consistently at 4 Mbps during active periods, a threshold $\theta = 1$ Kbps may be sufficient to distinguish active from idle states. This threshold-based mechanism enables the jammer to concentrate energy on periods of legitimate traffic, reducing overall transmission time and enhancing stealth.

B. Clustering-based Jammer

We design and implement a clustering-based jamming strategy in which the jammer passively monitors RAN analytics streamed by a subscribed and authenticated Y1 consumer.

Specifically, during an observation phase, it collects traffic feature vectors and trains a clustering model offline. This model is then used for real-time classification: at run-time, the jammer analyzes incoming analytics samples, infers their traffic class, and selectively activates interference accordingly. This approach enables a stealthy and adaptive jamming strategy that targets only high-value traffic patterns, improving jamming efficiency and minimizing the risk of detection.

Offline Training (Clustering Phase) in DBSCAN is applied to determine the optimal centroids of valid clusters, denoted by c_j^*, which are then used during *Online Testing* to identify whether the network is jammed. Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ be the set of KPMs collected from a real testbed, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i-th traffic feature vector. In our case, each vector $\mathbf{x}_i$ is defined as a set $\{\mathrm{CQI}, \mathrm{MCS}, \mathrm{Bitrate}, \mathrm{BLER}\}$. The standard score normalization is given by $\hat{\mathbf{x}}_i = (\mathbf{x}_i - \mu)/\sigma, \quad \forall i \in \{1, \dots, n\}$, where μ and σ are the mean and standard deviation vectors computed per feature.

We apply the unsupervised learning algorithm DBSCAN, as described in [18], to cluster the feature vectors x_i, by requiring that the n-th nearest neighbor of a point lies within a specified distance ε. Points for which this distance exceeds ε are considered noise or outliers, and do not belong to any cluster. The DBSCAN algorithm can be expressed as follows:

$$\mathtt{DBSCAN}(\hat{\mathcal{X}}, \varepsilon, \mathtt{minPts}) \xrightarrow{\text{Output}} \mathcal{L} = \{l_1, \dots, l_n\}, \quad l_i \in \{-1, 0, \dots, k-1\}, \quad (2)$$

where $\hat{\mathcal{X}}$ denotes the normalized KPMs, and `minPts` is the minimum number of points required to form a dense region. Each l_i assigns a point to a cluster, and $l_i = -1$ indicates noise. The optimal centroids of valid clusters (excluding noise) are computed as: $\mathbf{c}_j^* = \frac{1}{|\mathcal{C}_j|} \sum_{\hat{\mathbf{x}}_i \in \mathcal{C}_j} \hat{\mathbf{x}}_i, \quad \forall j \in \{0, \dots, k-1\}$, where $|\mathcal{C}_j| \geq \mathtt{minPts}$ is the number of points assigned to cluster j.

Online Testing (Live Classification): At runtime, the jammer receives a new vector $\mathbf{x}_t$ and performs the following procedures to identify jamming traffic. First, the feature vector is normalized using the `scaler.transform(·)` function, resulting in $\hat{\mathbf{x}}_t$. Next, the optimal centroids c_j^*, obtained from the *Offline Training*, are used to assign $\hat{\mathbf{x}}_t$ to the nearest cluster $\mathcal{I}$ according to: $j^* = \arg\min_j \|\hat{\mathbf{x}}_t - \mathbf{c}_j\|_2$. A jamming decision is then made based on the cluster index:

$$\text{JAM} \quad \text{if } j^* \notin \mathcal{I}; \quad \text{NO JAM} \quad \text{if } j^* \in \mathcal{I}, \quad (3)$$

where $\mathcal{I}$ is the index set of clusters designated for jamming based on the strategy (e.g., low or high traffic targeting).

VI. Testbed Development and Performance Evaluation

A. Experimental Testbed

Our experimental testbed, depicted in Fig. 2, comprises four physical systems configured to emulate a realistic O-RAN environment. These systems represent: (i) the RAN and Core Network (BS/EPC), (ii) User Equipment (UE), (iii) Near-RT RIC, and (iv) an external jammer. Three of the four systems are equipped with USRP B210 SDRs for over-the-air (OTA) transmission and reception.

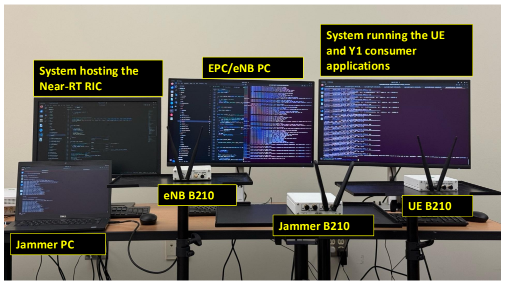
System hosting the Near-RT RIC
EPC/eNB PC
System running the UE and Y1 consumer applications
eNB B210
UE B210
Jammer B210
Jammer PC

Fig. 2. This is the OTA experimental setup, which consists of Near-RT RIC, BS, UE, and the jammer using USRP B210s as RF frontends.

The RAN is based on the srsRAN 4G stack (v21.10) [19], with the eNodeB and EPC co-located on a high-performance workstation (Intel Core i9-14900K @ 5.7GHz, NVIDIA RTX 4090 GPU, 64 GB RAM, Ubuntu 22.04). The UE is hosted on a separate system with identical hardware and software configuration, also equipped with a USRP B210 radio frontend. The malicious Y1 Consumer, implemented as a Flask-based API application, is co-located on the UE system to receive analytics from the Y1 producer and relay it to the jammer.

The Near-RT RIC based on Open Software Community (OSC) RIC runs on a dedicated Alienware Aurora R16 (Intel i9-14900KF @ 5.9GHz, NVIDIA RTX 4090, 64 GB RAM, Ubuntu 22.04) which host the Y1 RAI Producer application and a custom-developed KPM xApp that collects downlink RAN performance metrics via the E2 interface. These metrics are published to a Redis-backed SDL, from which the Y1 Producer exposes RAN analytics to authenticated third-party consumers via mTLS-secured API endpoints, adhering to the O-RAN WG3 Y1 specification. In our setup, the notification period which defines how often the Y1 Producer sends analytics updates to consumers was fixed at 1 second.

To emulate the threat model, we designate a separate laptop (Dell Latitude 7490, Intel Core i7-8650U @ 4.2GHz, 8 GB RAM, Ubuntu 22.04) as the external jammer platform. The jammer is implemented using GNU Radio and supports multiple jamming strategies, including: **Y1-aided Jamming** – *Threshold and clustering-based jamming techniques* presented in Section V, and two baseline jamming techniques: **Always-on Jamming** – the Jammer transmits continuously across the entire experiment, regardless of traffic activity, and **Random Jamming** – randomly activates the jammer in discrete bursts over time, without awareness of network conditions.

We used conventional DSP blocks, including a signal source, throttle, and a USRP sink block. Jamming control is achieved by programmatically adjusting the amplitude of the waveform (set to 1.0 for active jamming and 0.0 otherwise), and dynamically toggling the USRP antenna gain between 31 and 0. While amplitude controls waveform power, the antenna gain setting further amplifies or suppresses transmission. These controls are driven by the jamming logic (threshold or

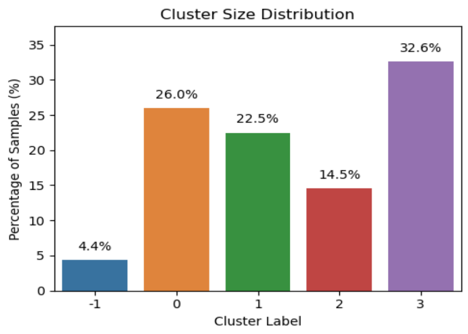

Fig. 3. Distribution of samples across DBSCAN-generated clusters. Noise samples are denoted by label -1.

clustering-based) in real time, enabling the jammer to activate or deactivate based on incoming analytics data.

The threshold condition and DBSCAN algorithm are implemented directly within the GNU Radio script, alongside a TCP socket-based module that enables live streaming of RAN analytics from the malicious Y1 Consumer to the jammer. Depending on the selected jamming strategy, the jammer can either (i) apply a simple threshold rule, activating transmission whenever observed metrics (e.g., bitrate or BLER) exceed a predefined level, or (ii) perform unsupervised traffic classification using DBSCAN to identify and react to specific traffic patterns. In both cases, the jammer selectively transmits during predicted active periods enabling dynamic, context-aware interference that reacts to evolving traffic patterns in the network. All traffic scenarios in our experiments in both Part A and Part B are generated using UDP-based `iperf3` traffic initiated from the base station to the UE, simulating downlink communication.

B. Experimental Scenarios

To evaluate and compare jamming strategies under realistic and repeatable conditions, we design two controlled experimental scenarios, each representing distinct traffic behaviors and jamming constraints.

a) Part A: Fixed Traffic Scenario (Single Traffic Class, Unlimited Jamming Budget): This scenario models a UE with a consistent downlink demand, transmitting at a fixed bitrate of *4 Mbps*. The traffic scheduler randomly alternates between active transmission periods and idle intervals. A fixed random seed is used to maintain deterministic traffic behavior across jammer evaluations. This setup enables fair comparison between jammers with no constraints on jamming duration. The total traffic session spans approximately **270 seconds**, with *75.4% active traffic* and *24.6% idle time*.

This scenario is used to compare three jamming strategies: a conventional *Always-on jammer*, a *Random jammer*, and a simple yet effective *Y1-threshold jammer*. The goal is to demonstrate that under unlimited jamming budgets, a threshold-based jammer, enabled by Y1 analytics can match the interference impact of an always-on approach while significantly reducing transmission overhead by aligning jamming activity with actual traffic periods.

b) Part B: Multi-Rate Traffic Scenario with Budget-Constrained Jamming: To evaluate jammer selectivity and intelligence under limited activity budgets, we create a diverse traffic profile that includes four traffic classes: *4 Mbps (35.5%)*, *2 Mbps (24.5%)*, *500 Kbps (24.5%)*, and *Idle periods (15.5%)*, across a total duration of **220 seconds**. This scenario is designed to demonstrate how a Y1-aided jammer can achieve greater disruption by selectively targeting high-impact traffic classes under constrained jamming budgets.

The Y1-aided jammer first collects RAN analytics from the malicious Y1 consumer and applies unsupervised clustering (DBSCAN) offline to learn traffic patterns. During runtime, the jammer uses this traffic profile to activate jamming only when specific clusters are detected. We evaluate multiple jamming strategies derived from this cluster-aware model, including jammers that target high traffic, medium traffic, and low traffic clusters, and compare their effectiveness against a baseline random jammer across varying jamming budgets (10%, 15%, 20%, 25% of total session time). This setup allows us to investigate the disruption-efficiency tradeoff and highlight how targeting higher-throughput traffic yields significantly greater impact per unit transmission time.

C. Clustering-Based Traffic Profiling for Adaptive Jamming

To support live inference in the Part B experiments, we trained the DBSCAN model using $n = 227$ samples of RAN analytics, each characterized by a 4-dimensional feature vector: `[CQI, MCS, Bitrate, BLER]`. After parameter tuning, the optimal clustering was achieved with a neighborhood radius of $\varepsilon = 0.30$ and a minimum point count of $\texttt{minPts} = 10$. This configuration yielded four distinct clusters and a small noise group, as shown in Fig. 3.

Manual inspection of the feature centroids revealed semantic groupings: Cluster 3 corresponded to high traffic (4 Mbps), Cluster 0 to medium traffic (500 Kbps), Cluster 2 to low traffic (100 Kbps), and Cluster 1 to idle states (near-zero activity). These labels were used to guide the jammer's live decisions in Part B, enabling selective interference based on estimated traffic levels.

D. Network Performance Metrics

To evaluate the impact of jamming on RAN performance, we focus on three key downlink performance indicators measured at the base station: **signal-to-noise ratio (SNR)**, **block error rate (BLER)**, and **throughput (DL bitrate)**. SNR reflects link quality, BLER quantifies transmission reliability, and throughput measures the effective data delivery rate under interference. Additionally, we introduce the **Bitrate Drop Percentage**, which quantifies the relative throughput degradation compared to the no-jamming baseline, it highlights the efficiency of a jammer in degrading the link quality. All metrics are averaged across active UEs (i.e., those with non-zero traffic) and visualized via cumulative distribution function (CDF) plots and tabular comparisons to evaluate interference effectiveness across strategies and jamming budgets.

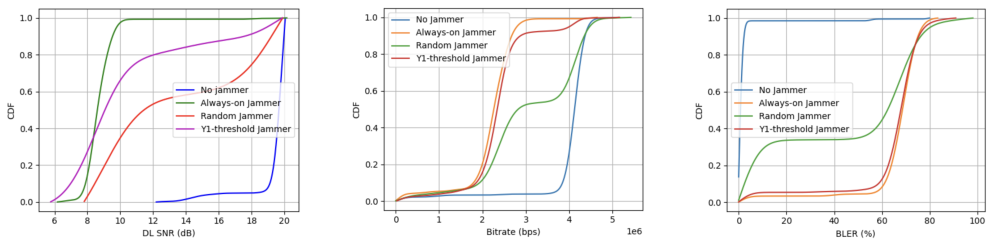

Fig. 4. Part A: Network Performance Metrics

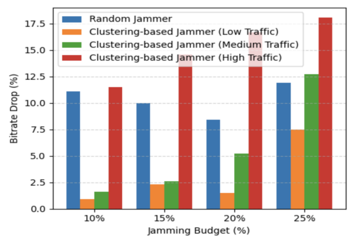

Fig. 5. Bitrate drop v.s. jamming budget

TABLE III
PART A JAMMER COMPARISON (SINGLE TRAFFIC CLASS)

Strategy	BLER (%)	SNR (dB)	Bitrate (bps)	Bitrate Drop (%)	Active Time (%)
No Jammer	0.72	19.6	3949274.9	0	0
Always-on Jammer	64.27	8.74	2154627.8	45.4	100
Random Jammer	34.54	13.88	3025942.7	23.4	56
Y1-Threshold Jammer	61.52	11.03	2348195.3	40.5	73

E. Network Performance Results

1) Part A Result - Evaluation under Fixed Traffic and unlimited Jamming Budget: Table III summarizes the performance of different jamming strategies under a single-rate traffic profile. As expected, the *Always-on jammer* introduces the most disruption, achieving the highest BLER (64.27%) and the largest bitrate reduction (45.4%) relative to the no-jamming case, but at the cost of continuous transmission (100% active time). In contrast, the *Y1-Threshold jammer*, guided by real-time traffic awareness, achieves nearly the same BLER (61.52%) with only 73% transmission activity closely aligned with the true traffic activity level of 75.4% as defined in the traffic scenario. This confirms the effectiveness of using Y1-based jamming to selectively jam only during meaningful transmission windows, avoiding unnecessary interference during idle periods.

Although the SNR of the Y1-threshold jammer (11.03 dB) is slightly higher than that of the Always-on jammer (8.74 dB), this is due to its deactivation during idle periods when no jamming is required, allowing clear channel conditions. The *Random jammer*, while active for 56% of the time, delivers less disruption (BLER of 34.54%, bitrate drop of 23.4%), highlighting the benefit of targeted jamming over probabilistic activation. This behavior is also reflected in the CDF plots shown in Fig. 4, where the Y1-threshold jammer closely mirrors the Always-on jammer in both BLER and bitrate metrics. An exception is observed in the SNR plot, where the threshold jammer achieves slightly higher SNR values due to its inactivity during idle periods, allowing for cleaner signal conditions compared to the Always-on jammer. In contrast, the Random jammer deviates more significantly across all metrics, highlighting its less effective and less targeted jamming behavior. These results demonstrate that even a simple Y1-threshold based jamming strategy can achieve near-optimal interference with significantly reduced energy and transmission effort, validating the threat potential of analytics-driven jammers.

2) Part B: Adaptive Jamming with Budget Constraints: Fig.5 and Table IV evaluate the disruption potential of various Y1-aided jamming strategies under constrained jamming budgets (10% to 25% of total session time). This experiment was designed to demonstrate how an intelligent jammer, equipped with prior traffic profiling capabilities, can maximize disruption while consuming the same jamming budget as a less strategic adversary. The random jammer, which lacks context-awareness, consistently achieves moderate bitrate degradation across all budgets. Despite its unpredictable activation pattern, it often outperforms jammers that target lower-rate traffic (i.e., 500 Kbps or 2 Mbps), which tend to carry less data and thus contribute less to overall throughput reduction when disrupted compared to targeting higher-rate traffic.

In contrast, as shown in Fig.5, the High Traffic jammer, guided by unsupervised traffic clustering, consistently delivers the highest disruption efficiency. It achieves a maximum bitrate drop of **18.1%** at 25% jamming budget, substantially higher than the random and lower traffic jammers, by focusing its transmission solely on the highest-bandwidth flow (4 Mbps). This confirms that targeting high-throughput periods maximizes disruption per unit of transmission time, validating the benefits of profiling-aware jamming.

Interestingly, the BLER values for High Traffic jamming are not always the highest. This counterintuitive trend likely results from rapid link adaptation in response to aggressive jamming during high MCS usage, which forces the network to lower its modulation and coding scheme to preserve reliability. As a result, throughput suffers significantly while average BLER remains controlled.

Overall, these results highlight the advantage of Y1-guided traffic profiling in enabling strategic, cost-effective jamming. When transmission budget is limited as would be the case in energy-constrained or covert scenarios intelligent selection of jamming windows can amplify disruption impact without requiring full-time interference.

F. Discussion

In this work, we implemented and demonstrated a new attack surface that emerges from the exposure of RAN analytics via the O-RAN Y1 interface. While our experimental framework employed a set of carefully selected RAN metrics shown in Table II that are not explicitly listed in the current Y1 specification, we emphasize that our choice aligns with the broader goal of simulating meaningful and realistic telemetry exchange between the RAN and authorized consumers. These metrics reflect aggregate downlink behavior, avoiding per-UE granularity, and were selected based on their prevalence

TABLE IV
PART B: JAMMING EFFECTIVENESS ACROSS TRAFFIC TYPES AND BUDGETS

Strategy	**10% Budget**			**15% Budget**			**20% Budget**			**25% Budget**		
	Bitrate	Drop (%)	BLER	Bitrate	Drop (%)	BLER	Bitrate	Drop (%)	BLER	Bitrate	Drop (%)	BLER
No Jammer	2418944	–	0.06	2418944	–	0.06	2418944	–	0.06	2386517	–	0.03
Random Jammer	2151102	11.1	5.86	2177414	10.0	12.34	2215156	8.4	15.09	2102078	11.9	14.61
Clustering-Based Jammer (Low Traffic)	2398083	0.9	7.90	2363917	2.3	12.04	2383796	1.5	13.98	2208256	7.5	21.78
Clustering-Based Jammer (Medium Traffic)	2380269	1.6	6.47	2355697	2.6	12.78	2294049	5.2	17.17	2082571	12.7	20.80
Clustering-Based Jammer (High Traffic)	**2141635**	**11.5**	8.33	**2065406**	**14.6**	10.61	**2013937**	**16.7**	20.71	**1954988**	**18.1**	18.47

in typical baseband telemetry. Moreover, these metrics are representative of the types of network characteristics captured by analytics listed in the YI specification, such as average packet loss, throughput, and packet delay rate. Importantly, our methodology and attack model are agnostic to the exact analytic fields and could be extended to support the analytics explicitly defined in the Y1 specification without altering the overall flow or inference mechanism.

Our findings underscore a key insight: the presence of any sufficiently informative RAN telemetry can serve as a potent enabler for adversarial profiling. Even limited exposure to periodic or aggregated statistics allows a well-positioned attacker to learn temporal traffic patterns, and selectively trigger interference with minimal footprint.

VII. Conclusion and Future Work

This work demonstrates that jamming attacks guided by RAN analytics from the O-RAN Y1 interface can be highly effective and efficient. Using an OTA testbed, two Y1-aided strategies, a threshold-based jammer and a DBSCAN-based jammer, were evaluated against always-on and random jammers. Results show that even lightweight Y1-based jammers can match the disruption of always-on attacks while cutting transmission overhead by over 25%. Clustering-based jammers further improve efficiency, achieving up to 18% throughput degradation with only 25% transmission time. These highlight the risks of exposing analytics via open interfaces like Y1, especially in military contexts, motivating future research on Y1 access control defenses.

VIII. Acknowledgment

This work has been supported by the Public Wireless Supply Chain Innovation Fund (PWSCIF) under Federal Award ID Number 51-60-IF007, NSF award 2120411, 2145631, and the NSF and Office of the Under Secretary of Defense Research and Engineering ITE 2326898 and 2515378, as part of the NSF Convergence Accelerator Track G: Securely Operating Through 5G Infrastructure Program.

References

[1] N. D. Tripathi and V. K. Shah, *Fundamentals of O-RAN*. John Wiley & Sons, 2025.

[2] A. Gatherer, C. Sengupta, S. Sen, and J. H. Reed, "Dual-use commercial and military communications on a single platform using ran domain specific language," 2024. [Online]. Available: https://arxiv.org/abs/2412.00983

[3] S. Gopal, W. Garey, R. Rouil, and S. Kowdley, "Towards adversary-resilient interference management in o-ran: Requirements and gap analysis," National Institute of Standards and Technology, Gaithersburg, MD, NIST Interagency/Internal Report (NISTIR), 2025. [Online]. Available: https://doi.org/10.6028/NIST.IR.8560

[4] O.-R. ALLIANCE, "O-ran specifications," 2023, accessed: 2025-05-29. [Online]. Available: https://www.o-ran.org/specifications

[5] W. G. 3, "O-ran y1 interface: General aspects and principles," O-RAN ALLIANCE Working Group 3, Technical Specification O-RAN.WG3.Y1GAP-R004-v01.01, 2024. [Online]. Available: https://orandownloadsweb.azurewebsites.net/specifications

[6] R. Balakrishnan, M. Arvinte, N. Himayat, H. Nikopour, and H. Moustafa, "Enhancing o-ran security: Evasion attacks and robust defenses for graph reinforcement learning-based connection management," 2024. [Online]. Available: https://arxiv.org/abs/2405.03891

[7] Z. A. E. Houda, H. Moudoud, and B. Brik, "Federated deep reinforcement learning for efficient jamming attack mitigation in o-ran," *IEEE Trans. Veh. Technol.*, vol. 73, no. 7, pp. 9334–9343, 2024.

[8] Y. A. Ergu and V.-L. Nguyen, "Radar: Robust drl-based resource allocation against adversarial attacks in intelligent o-ran," *IEEE Trans. Green Commun. Networking*, pp. 1–1, 2025.

[9] Y. A. Ergu, V.-L. Nguyen, R.-H. Hwang, Y.-D. Lin, C.-Y. Cho, H.-K. Yang, H. Shin, and T. Q. Duong, "Efficient adversarial attacks against drl-based resource allocation in intelligent o-ran for v2x," *IEEE Trans. Veh. Technol.*, vol. 74, no. 1, pp. 1674–1686, 2025.

[10] A. Chiejina, B. Kim, K. Chowhdury, and V. K. Shah, "System-level analysis of adversarial attacks and defenses on intelligence in o-ran based cellular networks," in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2024, p. 237–247. [Online]. Available: https://doi.org/10.1145/3643833.3656119

[11] J. Moore, A. S. Abdalla, P. Khanal, and V. Marojevic, "Integrated llm-based intrusion detection with secure slicing xapp for securing o-ran-enabled wireless network deployments," 2025. [Online]. Available: https://arxiv.org/abs/2504.00341

[12] C.-F. Hung, C.-H. Tseng, and S.-M. Cheng, "Anomaly detection for mitigating xapp and e2 interface threats in o-ran near-rt ric," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 1682–1694, 2025.

[13] T. O. Atalay, S. Maitra, D. Stojadinovic, A. Stavrou, and H. Wang, "Securing 5g openran with a scalable authorization framework for xapps," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[14] E. Aizikovich, D. Mimran, E. Grolman, Y. Elovici, and A. Shabtai, "Rogue cell: Adversarial attack and defense in untrusted o-ran setup exploiting the traffic steering xapp," 2025. [Online]. Available: https://arxiv.org/abs/2505.01816

[15] S. Amuru, C. Tekin, M. van der Schaar, and R. M. Buehrer, "A systematic learning method for optimal jamming," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 2822–2827.

[16] O.-R. S. Docs, "Welcome to o-ran shared data layer (sdl) in c++ documentation," *O-RAN Software Community, RIC SDL Documentation*, 2023.

[17] O-RAN ALLIANCE Security Work Group (WG11), "O-ran security test specifications," O-RAN ALLIANCE, Tech. Rep., 2024, accessed: 2025-05-30. [Online]. Available: https://specifications.o-ran.org/specifications

[18] M. F. Ivarsen, J.-P. St-Maurice, G. C. Hussey, D. R. Huyghebaert, and M. D. Gillies, "Point-cloud clustering and tracking algorithm for radar interferometry," *Phys. Rev. E*, vol. 110, p. 045207, Oct 2024. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.110.045207

[19] S. R. Systems, "srsran: Open-source lte/5g ran software suite," *GitHub Repository*, 2024, accessed: 2024-12-09. [Online]. Available: https://github.com/srsran/srsRAN