This paper has been accepted for publication on IEEE Transactions on Mobile Computing, DOI 10.1109 / TMC.2024.3393430.

Securing O-RAN Open Interfaces

Joshua Groen, Salvatore D'Oro, Utku Demir, Leonardo Bonati, Davide Villa,
Michele Polese, Tommaso Melodia, Kaushik Chowdhury

Abstract—The next generation of cellular networks will be characterized by openness, intelligence, virtualization, and distributed computing. The Open Radio Access Network (Open RAN) framework represents a significant leap toward realizing these ideals, with prototype deployments taking place in both academic and industrial domains. While it holds the potential to disrupt the established vendor lock-ins, Open RAN's disaggregated nature raises critical security concerns. Safeguarding data and securing interfaces must be integral to Open RAN's design, demanding meticulous analysis of cost/benefit tradeoffs.

In this paper, we embark on the first comprehensive investigation into the impact of encryption on two pivotal Open RAN interfaces: the E2 interface, connecting the base station with a near-real-time RAN Intelligent Controller, and the Open Fronthaul, connecting the Radio Unit to the Distributed Unit. Our study leverages a full-stack O-RAN ALLIANCE compliant implementation within the Colosseum network emulator and a production-ready Open RAN and 5G-compliant private cellular network. This research contributes quantitative insights into the latency introduced and throughput reduction stemming from using various encryption protocols. Furthermore, we present four fundamental principles for constructing security by design within Open RAN systems, offering a roadmap for navigating the intricate landscape of Open RAN security.

Index Terms—Security, 5G, O-RAN, Emulation, Encryption.

1 Introduction

The Open Radio Access Network (Open RAN) paradigm and its embodiment in the O-RAN ALLIANCE specifications [1] aim to transform the 5G (and beyond) ecosystem via open, intelligent, virtualized, and fully inter-operable RANs. The O-RAN architecture separates itself from legacy blackbox and monolithic RAN architectures by adopting a more flexible approach where base stations, e.g., Next Generation Node Bases (gNBs), are converted into disaggregated and virtualized components that are connected through open and standardized interfaces [2]. For example, Fig. 1 shows the Open Fronthaul connecting the RU and DU and the E2 interface connecting the DU and CU to the near-RT RIC. This paradigm shift is a potential enabler of data-driven optimization, closed-loop control, and automation [3], thus making it possible to break the stagnant vendor lock-in of closed networking architectures used in 4G legacy

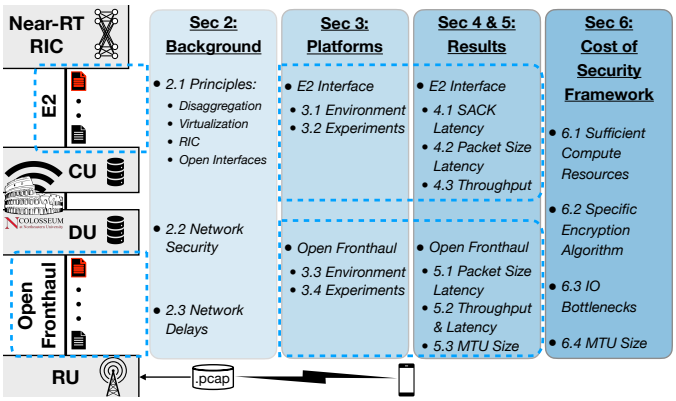

Fig. 1: Overview of the study, focusing on securing the O-RAN architecture's open interfaces.

systems. The introduction of open interfaces to interconnect disaggregated RAN components paves the way to intelligent, flexible, and scalable cellular architectures. However, this effectively opens up the network and allow access to its elements via software, extending the threat surface and exposing the network to a variety of vulnerabilities and attacks [4, 5].

In fact, one of the major threats to O-RAN open interfaces arises from improper or missing ciphering of the data sent across them [5, 6]. O-RAN interfaces transport potentially sensitive user data and network telemetry and control, which need protection against data tampering and eavesdropping. Similarly, the introduction of Radio Access Network (RAN) Intelligent Controllers (RICs)—software entities hosting Artificial Intelligence (AI) algorithms executed through applications known as xApps and rApps—renders the network susceptible to various adversarial attacks. These attacks aim to manipulate the AI towards inefficient control policies or decisions that could potentially degrade network performance [7] and allocate RAN resources unfairly [8]. Additionally, adversaries might attempt to bypass authentication procedures, thereby authorizing the execution of malicious software applications [9, 10].

Security is a crucial aspect of any system as it influences its adoption and utilization. The success of O-RAN will inevitably rely upon its security framework, procedures, and defense mechanisms. For this reason, the O-RAN ALLIANCE, industry, government organizations, and academia alike have put significant effort in laying out

The authors are with the Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA. E-mail: {groen.j, s.doro, u.demir, l.bonati, villa.d, m.polese, t.melodia, k.chowdhury}@northeastern.edu.
This article is based upon work partially supported by Qualcomm, Inc. and by the U.S. National Science Foundation under grants CNS-1925601, CNS-2112471, CNS-2117814, and CNS-2120447.

best practices and identifying threats and their countermeasures [4–6, 11–13]. For example, *O-RAN WG11: Security Work Group* has developed an extensive threat analysis for O-RAN systems and recommendations to secure them. More specifically, *O-RAN WG3: The Near-Real-Time RIC and E2 Interface Work Group* is working on identifying threats and guidance to secure the E2 open interface with confidentiality, integrity, replay protection, and data origin authentication mechanisms [12]. In contrast, at the time of writing, the widely adopted Open Fronthaul interface O-RAN standards call for no encryption mechanism because of the high bandwidth and strict latency requirements. Nonetheless, there are works in the literature that suggest using Media Access Control Security (MACsec) for this interface [8, 14]. MACsec can be used to prevent a host of threats to this interface, including: inserting traffic, network intrusion, and man-in-the-middle attacks. However, prior work does not consider an actual implementation of MACsec for the Open Fronthaul nor analyze its overhead cost.

Motivation. Although the examples above demonstrate traction and desire to make O-RAN secure by addressing a variety of threats, to the best of our knowledge there has been no systematic study to identify and measure the cost that security has on O-RAN open interfaces. It is vital that an informed and risk-based approach is taken to adequately address security concerns in O-RAN, while recognizing that any method for enhancing security, such as adding encryption, comes at a performance cost [11]. Our goal in this paper is to (i) understand if such a cost is tolerable (motivating a rapid adoption of security protocols that can be used without impacting performance and normal operations of the network); and (ii) identify which elements contribute the most to such costs informing system architects on how to design security systems for O-RAN that are practical and sustainable.

To derive practical insights, it is essential for security studies to rely on empirical data obtained from the execution of security algorithms on O-RAN hardware and software components. This approach enables accurate measurement of how security mechanisms impact resource utilization, processing latency, and data rates. For this reason, we leverage the broad and public availability of O-RAN testbeds [15–19] to thoroughly test and analyze the effects of encryption on a variety of O-RAN interfaces. We distinguish between two primary categories of interfaces within our study, offering specific examples of each to provide a clear framework for our analysis. This approach allows us to delve into the unique characteristics and requirements of each interface type, aiding in a more precise evaluation of the impact of security measures.

1) **RIC Data Interfaces:** This category comprises interfaces that the RICs utilize for both receiving and transmitting data. The O-RAN architecture features two RICs, executing control loops with a near-real-time scale (10 ms to 1 s) and a non-real-time scale (beyond 1 s). Notable examples of interfaces for the RICs include E2, O1, and A1. For our analysis, we specifically focus on the impact of adding Internet Protocol security (IPsec) to the E2 interface by extending our prior works and findings in [7, 20], as it plays a pivotal role in facilitating the exchange of packets between the near-RT RIC and the Central Units (CUs)/Distributed Units (DUs).

2) **Interfaces Enabling gNB Disaggregation:** The second category encompasses interfaces that are essential for supporting the disaggregation of gNBs. One of the key interfaces within this category is the Open Fronthaul interface for the 7.2 split of the 3GPP stack. This interface—defined by the O-RAN ALLIANCE—serves as a critical link between the Radio Unit (RU) and the DU, managing the transmission of time-sensitive data in substantial volumes, such as In-phase and Quadrature (IQ) samples. For this class, we examine the cost of securing the Open Fronthaul interface with MACsec.

The main contributions of our work are as follows:

We conduct the first-ever experimental analysis of adding O-RAN-compliant encryption to the O-RAN E2 and Open Fronthaul interfaces using Colosseum and a private 5G O-RAN-compliant testbed. Specifically, we extend our previous study [20] that analyzes the impact of adding security to the E2 interface with IPsec and report, for the first time, an experimental analysis of the effects of adding security with MACsec to the O-RAN Open Fronthaul interface.

We showcase the validity of our results through live experimentation that sheds light into the performance of O-RAN interfaces when the proposed security measures are deployed. Additionally, we validate at-scale a theoretical framework for calculating the total network delay when adding security protocols to distributed functional units. We extend these results by developing a general framework for understanding the cost of encryption in O-RAN with the goal of enabling researchers and engineers to build secure-by-design O-RAN systems.

We develop two separate O-RAN emulation environments and will publicly release the set of tools and datasets used to analyze the impact of adding security to Open Interfaces upon acceptance of this paper.

We derive insights and identify four key principles that system designers should be aware of to build future O-RAN systems that are secure by design. These principles are: sufficient compute resources must be provisioned, specific protocol implementations and encryption algorithms matter greatly, user space and kernel space I/O bottlenecks must be addressed, and the network Maximum Transmission Unit (MTU) size should be optimized.

The rest of the paper's organization is shown in Fig. 1. Section 2 gives a brief overview of key O-RAN principles. Section 3 describes our emulation environments. Our experimental procedures and results are detailed in Section 4 for the E2 interface and in Section 5 for the Open Fronthaul interface. We provide additional analysis of our results in Section 6 and conclude the paper in Section 7.

2 Background

The goal of this section is to provide background that will be useful later, when we delve into the details of the security assessment study. We first provide a brief overview of O-RAN shown in Fig. 2, and its foundational principles (Sec. 2.1), and then introduce network security protocols that will be used in this work (Sec. 2.2). Finally, we give an overview of the types of delay in packet switched networks (Sec. 2.3).

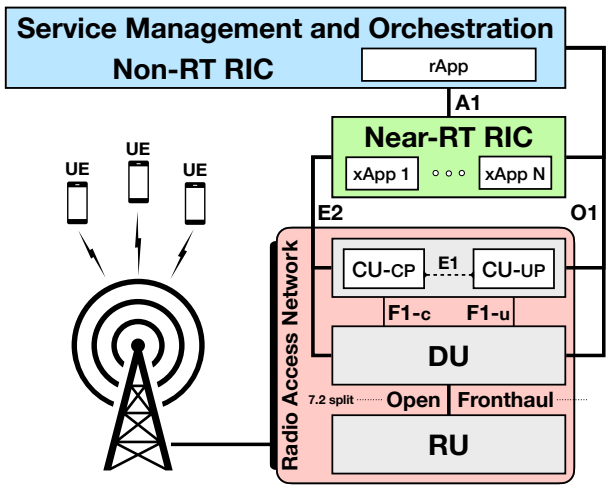

Fig. 2: Overall O-RAN architecture showing the 7.2x split with disaggregated DU, RU, and CU. The Near-RT and Non-RT RICs house Machine Learning (ML) models to provide closed-loop RAN control under different time scales.

2.1 O-RAN Principles and Architecture

Disaggregation and Virtualization: O-RAN embraces disaggregation principles by effectively splitting base stations into multiple functional units, namely the CU, the DU, and the RU. The CU is split further into the Control Plane (C-plane) and the User Plane (U-plane). This logical split is performed via virtualization and softwarization, which allow different functions to be executed at different locations and on different platforms across the network. These functional units, shown in Fig. 2, can be abstracted from the physical infrastructure and deployed as software components (e.g., microservices, containers). This architecture enables a decoupling between hardware and software components, sharing of hardware among different tenants, portability and execution on general-purpose hardware, and automated deployment of RAN functions. It is essential to grasp how virtualization and security considerations impact resource utilization, as this understanding is critical for designing solutions that can accommodate the security overhead effectively without overloading resources and introducing excessive latency.

RIC: O-RAN introduces two RICs, the non-RT RIC and the near-RT RIC. The RICs are both designed to enable intelligent decision-making, network monitoring, and control via AI and ML solutions that are fed with data transferred across O-RAN open interfaces. The near-RT RIC receives data over the E2 interface and handles control loops on a time scale between 10 ms and 1 s, using plug-and-play components called xApps. The non-RT RIC collects data via the O1 interface and operates at time scales higher than 1 second using rApps, and is embedded in the Service Management and Orchestration (SMO) framework. Although both RICs play a vital role in the lifecycle management of O-RAN systems, in this paper, we focus on the near-RT RIC only. The near-RT RIC is the most sensitive to latency and overhead introduced by security mechanisms due to its stringent operational time scale and its tight coordination with controlled gNBs. Indeed, as illustrated in Fig. 2, the near-RT RIC interfaces with the CUs and DUs of the distributed gNBs. On the contrary, since the non-RT RIC operates with latency timescales of 1s or greater, the cost of securing it (in terms of overhead and latency) is incremental and marginal.

Open Interfaces: While decoupling hardware and software creates an open environment for faster development, it also introduces the need for interoperable interfaces. These are some of the key elements necessary to overcome the traditional RAN black-box approach as they expose network parameters to the RICs and enable data analytics and ML-enabled control. The O1 interface is the primary interface with the SMO and the non-RT RIC and is responsible for enabling operations and maintenance. Similarly, the O2 interface connects the SMO to the O-Cloud, the abstraction for the infrastructure supporting O-RAN virtualization. The A1 interface connects the two RICs and is used for deploying policy-based guidance. The E2 interface is the key interface that connects the near-RT RIC to the RAN (see Fig. 2). The E2 interface enables the collection of metrics from the RAN to the near-RT RIC and allows the RIC to control multiple functions in the disaggregated gNB. Finally, the Open Fronthaul connects a DU to one or multiple RUs inside the same gNB [21]. The Open Fronthaul is further broken down into four distinct planes: (C)ontrol, (U)ser, (S)ynchronization, and (M)anagement plane. For example, the U-plane carries the actual user data in the form of IQ samples, while the S-plane carries timing and synchronization messages. The requirements in terms of both traffic type and security vary greatly between the planes. A comprehensive discussion of these and additional interfaces can be found in [1, 6, 11, 12, 21, 22]. It is imperative to secure such interfaces as they might transport sensitive user data and network telemetry. The *O-RAN ALLIANCE WG11: Security Work Group* requires specific security functions to be supported on each interface along with what protocol will provide those functions [23, 24]. Additional interface details are specified by *WG4: Open Fronthaul Interfaces Workgroup* [21, 25] and *WG5: Open F1/W1/E1/X2/Xn Interface Workgroup* [26]. Table 1 lists each interface, security function, and protocol as specified by the O-RAN ALLIANCE.

Interface	Function				Protocol	Overhead (Bytes)
	C	I	A	R		
A1, O2	✓	✓	✓	✓	TLS	≥ 25
O1	✓	✓	✓		TLS	≥ 25
F1- C	✓	✓		✓	TLS	≥ 25
E2	✓	✓	✓	✓	IPsec	≥ 57
F1, W1, E1, X2, Xn	✓	✓	✓		IPsec	≥ 57
Fronthaul- M	✓	✓	✓		SSHv2 or TLS	≥ 28 or ≥ 25
Fronthaul- C,U,S			✓		802.1x	N/A

TABLE 1: O-RAN Interface security functions (C: Confidentiality; I: Integrity: A: Authentication; R: Replay protection) and protocols specified by O-RAN ALLIANCE WGs 4, 5 & 11.

2.2 Network Security Protocols

There are many useful models to describe the set of network security services. Here we focus on the four functions used by *O-RAN ALLIANCE WG 11*: Confidentiality, Integrity, Authentication, and Replay Protection. Confidentiality ensures

the payload of the message cannot be read while the data is in transit. Integrity guarantees the content of the message is not changed in transit. Authentication, in this context, guarantees the endpoints of a conversation are who they say they are. Finally, Replay Protection ensures a pre-recorded message cannot be sent as a new message in the future.

There are a wide range of protocols used to provide these network security functions at various layers of the network protocol stack. In this paper, we consider: Transport Layer Security (TLS), which operates at the transport layer, IPsec, which operates at the network layer, and MACsec, which operates at the data link layer. While each protocol can provide similar general security functions, there are also distinctions among them.

TLS [27] is a widely used transport layer security protocol that provides authentication during the initial handshake process, and offers confidentiality by using a suite of algorithms to encrypt the transport layer payload. TLS also provides integrity and replay protection through the use of a message authentication code to prevent replay attacks. Newer releases of TLS (e.g., TLS 1.3) have faster Security Association (SA) establishment when compared to previous versions of TLS and IPsec. After the initial SA is established, TLS adds about 25-40 Bytes of overhead to each packet.

IPsec works at the network layer and offers several modes of operation. The primary security modes are Authentication Header (AH) [28], which provides integrity, authentication, and replay attack protection; and Encapsulating Security Payload (ESP) [29], which additionally provides encryption. In practice, ESP is almost exclusively used and the O-RAN ALLIANCE has mandated its use. IPsec can also operate in either transport or tunnel mode. Tunnel mode creates a new Internet Protocol (IP) header for each packet and protects the integrity of both the data and the original IP header [30]. On the contrary, transport mode only secures the network layer payload. O-RAN specifications mandate the support of tunnel mode, while support for transport mode is optional. Even though IPsec uses a slightly different handshake to establish SAs compared to TLS, it still offers all the same security functions and utilizes the same cryptographic functions as TLS after the SA is established. When using ESP and tunnel mode, IPsec adds at least 57 Bytes of overhead to each packet. However, in practice this is often higher because the cryptographic functions are block ciphers, requiring a fixed-size input.

MACsec [31], is used for securing point-to-point connections at the data link layer. MACsec offers two modes: encryption on or off. Both modes provide integrity, replay protection, and authentication, but only the encryption on mode offers confidentiality. Unlike TLS and IPsec, the MACsec standard specifies a single cryptographic algorithm, AES128-GCM, though implementations using AES256-GCM are available. MACsec adds a fixed 32-byte header to all packets. However, similar to the other encryption protocols, MACsec uses AES for confidentiality, which is a block cipher. Therefore, when encryption is enabled the overhead may be larger than 32 Bytes. Another security protocol that operates at the data link layer is IEEE 802.1x [32], which provides a standard for port-based authentication on Local Area Networks (LANs). 802.1x only provides periodic authentication (a common default is every 60 minutes) of devices connected over physical ports and no other security functions.

One of the primary differences between all these protocols is the way in which SAs are established. However, this happens infrequently; for TLS and IPsec by default the SAs expire after 8 hours, and for MACsec the SA expires after 1 hour. While this re-authentication time is configurable, the default values used in the majority of applications, on the order of a few hours, are considered secure. It should be noted that very frequent (on the order of seconds) re-authentication greatly reduces throughput. Additionally, all three protocols allow re-authentication before the current SA expires, guaranteeing continuous operation. Because this establishment happens very infrequently, it has a negligible impact on overhead and resource utilization. For this reason, we do not deeply analyze the initial handshake or SA establishment in this article. On the other hand, there are numerous differences in the encryption algorithms used and security services provided which, as we will show, have a significant impact on performance and resource utilization.

2.3 The Latency Cost of Security

To properly evaluate how security impacts network performance, it is important first to understand how the different security mechanisms affect the way data flows through the network. Specifically, since security adds overhead (both in terms of data and procedures), we need to establish a model that captures the effect that security has on total delay. In packet-switched networks, there are four primary sources of delay at each node along the path: *queuing* delay (D_{que}), *propagation* delay (D_{prop}), *transmission* delay (D_{trans}), and *nodal processing* delay (D_{proc}) [33]. The total delay can be expressed as

$$D_{total} = D_{que} + D_{prop} + D_{trans} + D_{proc}. \quad (1)$$

Queuing Delay: This delay considers the duration that packets spend in the processing queue at each interface along the end-to-end path. In our test environment, there is almost no competing traffic, allowing us to safely assume $D_{que} = 0$. In more complex networks, this assumption may not hold.

To analyze queuing delay further, we model the network nodes using an M/M/1 queue with packet arrivals forming a Poisson process. Then the average queuing delay is given by $D_{que} = \frac{1}{} \quad \frac{1}{}$ where is the traffic arrival rate and is the queue service rate. The difference between unsecured or Plain Text (PT) and secured or Cipher Text (CT) delays depends on the overhead added by encryption, denoted as . The value of is protocol-specific, but typically falls within the range of 60 Bytes (480 bits) or less. The difference in queuing delay is given by $D_{que} = \frac{1}{(+)} \quad \frac{1}{}$. When , the approximation $D_{que} \quad 0$ is valid. Even in significantly more congested networks, the queuing delay remains largely unaffected by the presence or absence of encryption. To illustrate, in our network supporting $= 10$ Gbps speeds, this approximation holds (i.e., $D_{que} \quad 1$ µs) when the arrival rate 9.78 Gbps.

Propagation Delay: The propagation delay is strictly a function of the physical length and propagation speed of the link. The propagation speed depends on the link type

but is typically on the order of 2 10^8 m/s [33]. For our environment, we assume a length of 100m giving D_{prop} = 0:05 µs. This will change for other systems but will remain constant regardless of encryption.

Transmission Delay: The transmission delay is a function of the packet size (in bits), L, and the link transmission rate, R, which is defined as $D_{trans} = L/R$. For any given system, R is fixed but L will increase with encryption. Table 2 lists the calculated transmission delays, with and without encryption, based on the average packet length of the three types of packets we observe and describe in detail in Sec. 3.1.1.

Packet Type	Plain Text	Cypher Text
SACK	62 B : 0.0496μs	138 B : 0.1104μs
Short E2AP	195 B : 0.1560μs	255 B : 0.2040μs
Long E2AP	1425 B : 1.140μs	1485 B : 1.188μs

TABLE 2: Calculated transmission delay for 3 types of packets with and without encryption.

Processing Delay: Typically, this is defined as the time required for intermediate nodes to inspect the packet header and determine the appropriate path for the packet, whether at layer 2 for switching or layer 3 for routing. This time can also encompass other factors, such as bit-level error-checking [33]. It's important to note that while the fronthaul network is expected to function as a pure layer-2 network (i.e., no routing involved), the E2 interface can be deployed over a layer-3 network, necessitating routing at certain intermediate nodes.

In our analysis, we include the encryption delay within the processing delay, as the cryptographic functions are integral to passing the payload to lower or higher layers in the network stack. Notably, this encryption delay occurs only at the sending and receiving nodes. Intermediate nodes do not need to engage in cryptographic operations because the Ethernet and IP headers are sent in PT.

3 O-RAN Security Evaluation Platforms

In this section, we describe the interface security evaluation platforms we used in this work. Specifically, the E2 interface implementation is illustrated in Sec. 3.1, and the Open Fronthaul interface implementation is described in Sec. 3.2. The results obtained on both platforms will be presented in Secs. 4 and 5.

3.1 E2 Interface

We extensively use Colosseum [15], the world's largest wireless network emulator with hardware in-the-loop, to evaluate the E2 interface. Colosseum supports experimental research through virtualized protocol stacks, enabling users to test full-protocol solutions at scale, with real hardware devices, in realistic emulated RF environments with complex channel interactions. The key building blocks for deploying full protocol stacks are 128 Standard Radio Nodes (SRNs). Each SRN consists of a 48-core Intel Xeon E5-2650 CPU with an NVIDIA Tesla k40m GPU connected to a USRP X310 Software-defined Radio (SDR). Users can instantiate custom protocol stacks by deploying Linux Containers (LXCs) on the bare-metal SRNs.

We utilize the srsRAN-based SCOPE [17] framework to implement a softwarized RAN for both the gNB and multiple User Equipments (UEs). SCOPE extends srsLTE (now srsRAN) version 20.04 by adding an E2 interface, several open APIs to facilitate run-time reconfiguration of the gNB, and additional data collection tools. We utilize the ColO-RAN [18] framework for the near-RT RIC implementation. ColO-RAN offers a minimal version of the O-RAN Software Community (OSC) near-RT RIC (Bronze release) tailored to execute on Colosseum via an LXC-packaged set of Docker containers. In particular, it provides an E2 interface implementation compliant with O-RAN specifications that can be used to interface with the RAN nodes for data collection and control. ColO-RAN also offers an extensible xApp template that collects basic Key Performance Metrics (KPMs) from the gNB and can send control actions to it.

To secure the E2 interface, we add the strongSwan [34] open source IPsec-based VPN to both the SCOPE and ColO-RAN LXCs. The full IPsec configuration is described in paragraph 3.1.1. We also add several simple scripts to automate data collection on E2 interface performance which we make open source for public use and further research.

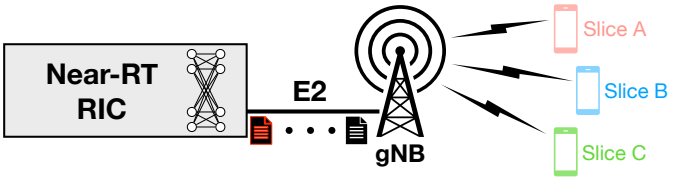

Fig. 3: O-RAN testbed used to study security in the E2 interface, consisting of three UEs, a gNB, and the near-RT RIC. Each component is implemented in an LXC on top of an SRN within Colosseum.

3.1.1 E2 Experimental Setup

Our experimental system (see Fig. 3) is composed of up to twelve blocks: 10 UEs, a gNB, and the near-RT RIC. Each block is implemented through LXC on separate SRNs. The UEs are connected to the gNB over an emulated RF channel where each UE is assigned to one of three unique slices representing the three main use cases for 5G: enhanced Mobile Brodband (eMMB), Ultra Reliable Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC) [35]. Each slice has its own traffic pattern generated from real world 5G traffic traces as described in [36]. First, we use a variety of applications to generate traffic for each network slice. For eMBB, we stream videos, browse the Internet, and transfer large files. For URLLC, we conduct both voice phone calls, video chat, and utilize real time AR applications. For mMTC we capture texts and background traffic from all apps when the phone is not actively being used. This is not the typical example of mMTC traffic, such as IoT applications. However, it does fit nicely in the fundamental definition of mMTC because it is low throughput, latency tolerant communication from numerous applications. Next, we built a traffic generator tool to replay the traffic between the UE and gNB. The traffic generator emulates the original traffic by reading the length field for each packet and sending a random byte string of

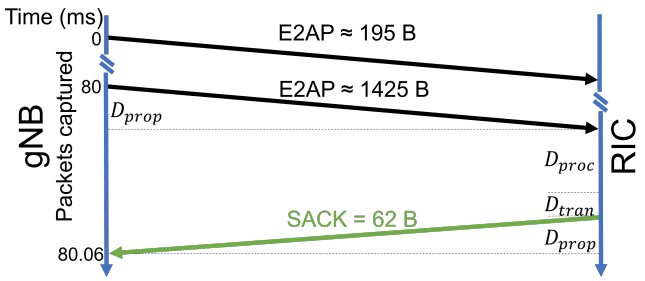

Fig. 4: E2 traffic displays a common pattern of one small E2AP packet then one large E2AP packet followed by a SACK as seen in this flow graph.

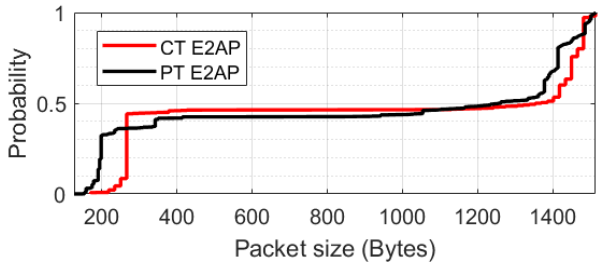

Fig. 5: CDF of E2AP packet length for both PT and CT traffic.

the appropriate length at the time indicated by the packet timestamp. This enables us to replicate the timing, length, and direction of all data sent between the UE and gNB, while completely anonymizing the actual payload within our experimental test bed. Our O-RAN test bed further emulates the channel conditions between the gNB and UE based on measured channel conditions for a real deployed cellular system. This allows us to accurately capture the O-RAN KPIs as if the original communication were taking place in a deployed O-RAN test bed. The gNB is connected to the near-RT RIC over a wired 10 Gbps backbone network. We implement the sample KPM monitoring xApp from [18] that periodically polls the gNB for up to 31 KPMs for each UE. This generates between 200 Kbps of traffic with 3 UEs sending 6 KPMs up to 3.5 Mbps of traffic with 10 UEs sending 31 KPMs on the E2 interface.

We capture all traffic traversing the E2 interface at the gNB for over 20 minutes. The Stream Control Transmission Protocol (SCTP) is used as the transport layer protocol for all traffic. Fig. 4 illustrates a typical example of the captured traffic. First, the gNB sends a small data packet followed by a large data packet using E2 Application Protocol (E2AP) over SCTP. Fig. 5 shows the empirical CDF of the E2AP packet sizes with and without encryption. Finally, the near-RT RIC responds with a fixed-size Selective Acknowledgment (SACK) packet (62 Bytes). This pattern is consistent because SCTP specifies that a SACK should be generated for every second packet received [37]. Any additional processing delay introduced by IPsec to the E2 interface takes place at the kernel level, outside of the srsLTE software stack. Even though we do not have access to the precise time when the gNB starts processing or transmitting a packet, we can observe the delay between the transmission of the large E2AP packet and the reception of the SACK at the gNB. For these reasons, we first study the effect of encryption on the SACK.

After establishing a baseline performance without encryption, we add O-RAN-compliant encryption as specified in [12] to the E2 interface. We implement IPsec with ESP in tunnel mode. For each packet, tunnel mode creates a new IP header, and protects the integrity of both the data and original IP header [30]. We use AES256 for encryption and SHA2-256 for the authentication hash function. AES256 is a high-speed symmetric encryption algorithm that uses a fixed block size of 128 bits and a key size of 256 bits and performs 14 transformation rounds [38]. SHA2-256 uses eight 32-bit words and performs 64 transformation rounds to compute a 256-bit hash [39]. However, only the first 128 bits of the hash are included in the IPsec trailer. IPsec, as configured in our test, provides all the required services listed in the O-RAN specifications for E2 [12]. With this configuration, IPsec adds at least 57 Bytes of overhead to each packet. However, because both AES256 and SHA2-256 require fixed input block sizes, padding may be added causing the overhead to further increase. For example, encrypting the SACK adds 76 Bytes for a total CT SACK length of 138 Bytes. We generate the same UE traffic described earlier, poll the gNB for the same KPMs, and again capture the traffic traversing the E2 interface at the gNB.

3.2 Open Fronthaul

An overview of the Open Fronthaul interface is shown in Fig. 6. While the Open Fronthaul interface can be viewed as a single connection (blue pipe) passing through other networks (white cloud), the O-RAN ALLIANCE WG4 specifies the allowable latency's for downlink (T12) and uplink (T34) separately.

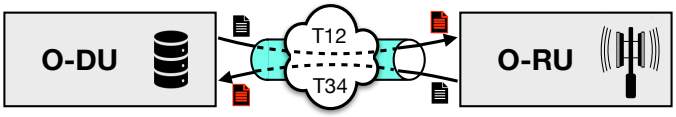

Fig. 6: The Open Fronthaul (shown in blue) connects the O-DU and O-RU, potentially across a wider network (cloud). The O-RAN ALLIANCE WG4 specifies transmission delays for the downlink (T12) and the uplink (T34).

We leverage the NVIDIA Aerial Research Cloud (ARC) platform [40] to capture traffic from a production-ready O-RAN and 5G-compliant system based on a 3GPP 7.2 split [21]. NVIDIA ARC combines the open-source project OpenAirInterface (OAI) [41] for the higher layers of the protocol stack with the NVIDIA Aerial physical implementation, which runs on Graphics Processing Unit (GPU) for inline acceleration (i.e., NVIDIA A100). The GPU in an ARC server is combined with a programmable Network Interface Card (NIC) (in our case, a Mellanox ConnectX-6 Dx) through remote direct memory access (RDMA), bypassing the CPU to transfer packets from the NIC to the GPU itself. This makes it possible to implement a high-speed DU-side termination of the Open Fronthaul interface, capable of block floating point compression to ensure prompt delivery of the IQ samples and control messages to the RU.

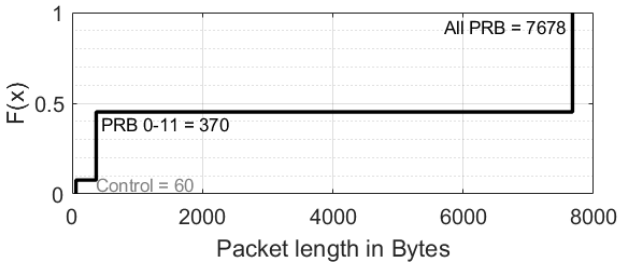

Fig. 7: CDF of Open Fronthaul packet length, highlighting the three types of packets observed: C-plane, U-plane with PRB 0-11, and U-plane with all PRBs.

Specifically, we leverage a private 5G network deployed at Northeastern University (part of the X-Mili project) [19], including 8 ARC nodes with a dedicated Core Network (CN) and fronthaul infrastructure. The fronthaul infrastructure features a Dell S5248F-ON switch, with a Qulsar QG-2 acting as a grandmaster clock. The latter distributes Precision Time Protocol (PTP) and SyncE synchronization to the DU and the RU. In our setup, the RU is a Foxconn 4T4R unit operating in the 3.7 – 3.8 GHz band, and we use Commercial Off-the-Shelf (COTS) 5G UEs from OnePlus (AC Nord 2003) [42].

While the code base for the DU is open and potentially extensible to embed MACsec, the RU comes with a closed-source FPGA-based termination for the fronthaul interface. This prevents us from directly enabling MACsec in our Open Fronthaul environment. Therefore, we adopt a trace-based approach and configure a port of the fronthaul switch to mirror the fronthaul traffic to a server running a packet capture. We built an emulation environment in our lab using two desktop computers with Intel i9-13900K CPUs with NVIDIA Mellanox ConnectX-4 Lx NICs directly connected over a 10 Gbps Ethernet link. We leverage a Python script to re-play the original pcap file captured on the real Open Fronthaul. In this way, we can properly emulate the original Open Fronthaul capture and observe the impact of adding any encryption. The Linux kernel natively supports MACsec. We use the Ubuntu commands provided in documentation [43] to configure MACsec.

3.2.1 *Open Fronthaul Experimental Setup*

We capture all the traffic traversing our Open Fronthaul described in Sec. 3.2 for over 20 minutes for several different traffic loads. In our environment, the Open Fronthaul uses the eCPRI protocol to send both C-plane and U-plane messages. There are two network options to use eCPRI; the first uses the full network stack (UDP over IP), while the second is designed for point-to-point networks and only uses the MAC layer. Our system is the second; our traffic only contains an Ethernet header with the eCPRI header and payload. While eCPRI supports payloads of up to 8192 Bytes, we consistently observe a maximum frame length of 7678 Bytes on the wire as shown in Fig. 7.

Currently, the O-RAN ALLIANCE specifications do not call for any security on the Open Fronthaul (C, U, S) planes. However, there are known vulnerabilities to not securing the Open Fronthaul [9, 10]. For this reason, there is a growing momentum for advocating to secure this vital link using MACsec [8, 14]. MACsec can be configured with or without encryption and we test both modes of operation. As mentioned above, due to hardware limitations our experimental environment does not support MACsec hardware offloading in the NIC directly, so all MACsec operations are performed in software.

4 E2 Experimental Results

4.1 SACK Analysis

Fig. 8 shows the distribution of delay times for both PT and CT SACKs. It is immediately clear that encryption adds approximately 22 µs of delay on average. However, it is essential for O-RAN researchers, engineers, and system architects to fully understand both the cause and the impact of any extra overhead.

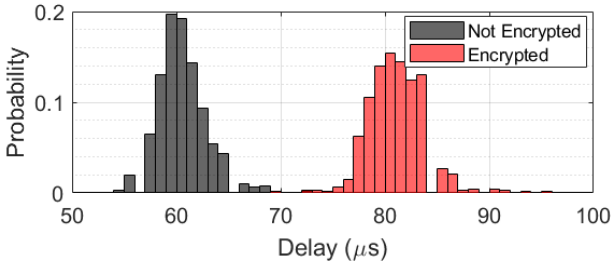

Fig. 8: SACK delay distribution with and without encryption.

The SACK packet is ideal for an initial analysis because the packet size is fixed (PT=62 B, CT=138 B) and the total delay we observe in our experiments is very tightly grouped, as seen in Fig. 8. The total average delay we observe for the PT SACK is 61.12 µs, while the CT SACK is 82.64 µs. From Fig. 4, we can see that the total delay for a SACK can be expressed as $D_{total}^{SACK} = 2 \quad D_{prop} + D_{trans} + D_{proc}$. Given the total delay, the transmission delays in Table 2, and the calculated propagation delay (Sec. 2.3), we can calculate the average processing delay; for the PT SACK it is 60.97 µs while the CT SACK is 82.64 µs. In Colosseum, encryption on the E2 link adds approximately 22 µs of delay to small packets. However, the near-RT RIC is designed to operate on scales from 10 ms to 1 s. **These results show that for the E2 Interface, with low traffic load (200*kbps*) and small packets (138*B*), encryption has no meaningful impact on E2 interface traffic.**

4.2 Packet Size Analysis

Our initial results examining the SACK over the E2 interface (Sec. 4.1) show that the processing delay accounts for at least 99.75% of the total delay for the 62 Byte SACK, regardless of encryption. In other words, the delay for short packets is dominated by the processing delay. Given the calculated propagation and transmission delays, it is likely that processing delay dominates in equation (1) for larger packets as well. However, since it is not possible to know the exact start times for transmitting the long X2AP packets based on the traces (and we do not have any SACK for the Open

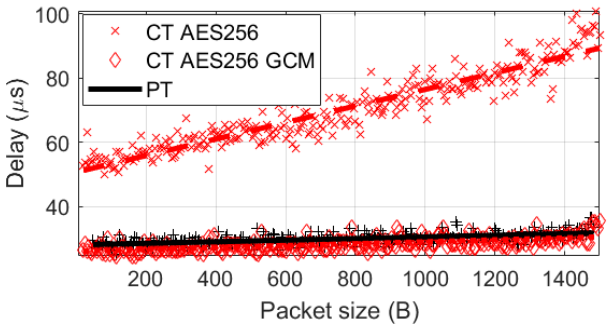

Fig. 9: Processing delay as a function of packet size for PT and CT traffic over the E2 Interface.

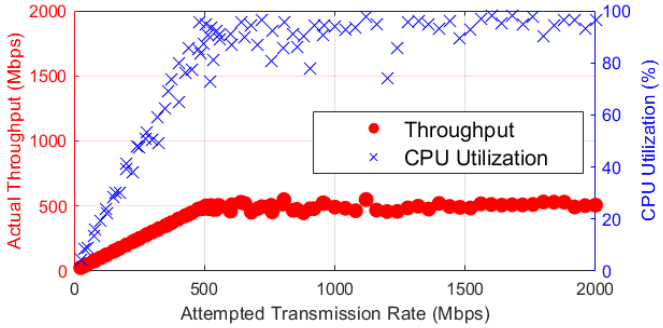

Fig. 10: Measured throughput and CPU utilization (in blue) as a function of attempted transmission rate for CT traffic (IPsec) over the E2 Interface while using AES256-CBC.

Fronthaul) we conduct another experiment to confirm this hypothesis.

To fully quantify the effect of encryption for packets of various lengths, we use ping (ICMP echo) of various lengths to accurately capture the network Round Trip Time (RTT). We start with a small payload for the ping and increment the size regularly. For each step size, we send 100 pings with 250 ms between each ping. Because there are no competing flows for this experiment the queuing delay is zero and the RTT is expressed as

$$RTT = 2 \quad (D_{proc} + D_{trans} + D_{prop}): \qquad (2)$$

Given the calculated propagation and transmission delays, 2 D_{prop} = 0:1 µs, 2 D_{trans} 2:4 µs, and the observed RTT 50 µs, we can approximate equation (2) as D_{proc} $\frac{RTT}{2}$. From the results shown in Fig. 9, we observe that the processing delay increases with packet length when sending encrypted traffic using AES256. However, the processing delay difference between CT and PT is D_{proc} 50 µs for all tested packet sizes. In contrast, when we use AES256-GCM we see no difference in processing delay between the PT and CT traffic. We can conclude that, regardless of the encryption algorithm used, **for all packet sizes from 62 B to 1500 B, encryption has minimal impact on E2 traffic**.

4.3 Throughput Analysis

We also design an experiment to quantify the effect of encryption on the total traffic throughput, T. We use iperf3 to generate traffic at specific bit rates for 10 seconds and regularly increment the transmission rate. We record the actual throughput reported by the receiving node. We also capture CPU utilization on the gNB (sending node) for each attempted transmission rate using iperf3.

From Fig. 10 it can be seen that the maximum encryption rate our system is capable of when using AES256-CBC as the encryption algorithm is 575 Mbps. Fig. 10 also shows the CPU utilization while using IPsec. While CPU utilization does increase for both types of traffic, we can see that encryption is very CPU-intensive. When sending PT, the CPU utilization increases proportional to 0:00365 T [20], whereas Fig. 10 shows the CPU utilization for CT grows proportional to 0:2 T until it reaches saturation. Therefore, we conclude that encrypting all traffic increases CPU utilization by roughly two orders of magnitude compared to PT traffic. In our system, using AES256 **CPU utilization is a limiting factor for encrypting traffic** when the attempted transmission rate is 575 Mbps.

We ran the same experiment with a wide variety of encryption algorithms and key sizes. We observed the same pattern where the CT throughput increases linearly until it plateaus. The maximum throughput varies greatly based on the encryption algorithm used, as seen in Table 3. **The specific algorithm implementation is the most significant factor when implementing IPsec on the E2 interface** with AES-GCM vastly outperforming the other algorithm implementations. Galois/Counter Mode (GCM) simultaneously provides confidentiality (using counter mode) and authentication (using arithmetic in the Galois field GF(2^n)), where n is the key size [44]. These operations can be performed in parallel, offering greater performance than other modes such as CBC, which require chaining of operations.

Encryption algorithm	Throughput
AES128-CBC	505 Mbps
AES256-CBC	512 Mbps
AES128-CCM	573 Mbps
AES256-CCM	573 Mbps
ChaCha20-Poly1305	989 Mbps
AES256-GCM	1370 Mbps

TABLE 3: 30 second throughput for various encryption algorithms. The algorithm and implementation have a greater impact on system performance than key size, with AES-GCM performing the best.

One key observation is that for all implementations there was virtually no difference in performance based on key size (AES128 versus AES256). In addition to the difference in key length, AES128 only uses 10 transformation rounds while AES256 uses 14 rounds [38]. The additional rounds for AES256 consist of the operations: SubBytes, ShiftRows, and MixColumns which are highly optimized for performance and do not add any significant delay. We encourage all system designers to use the longer 256-bit key length as it provides higher security with virtually no impact on performance. While individual systems and encryption algorithms may be different, these experiments show that CPU utilization is a key trade-off.

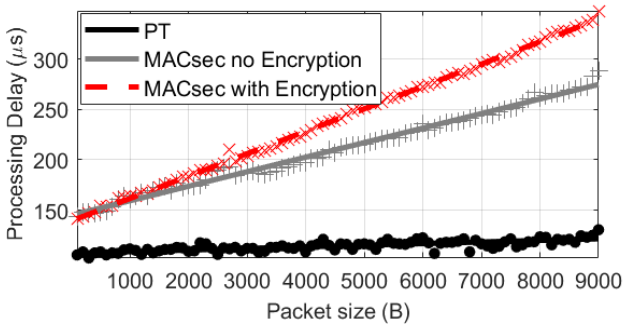

Fig. 11: Processing delay as a function of packet size over the Open Fronthaul.

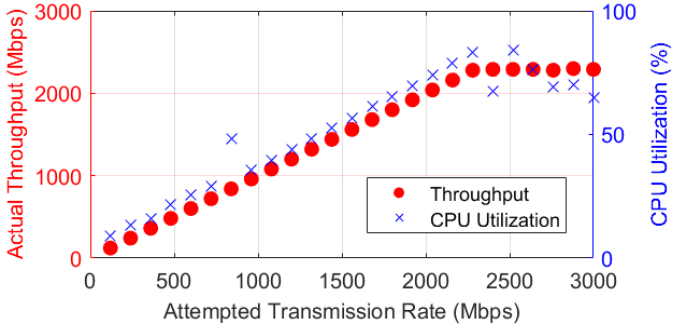

Fig. 12: Actual CT throughput (red) and CPU utilization (blue) for different attempted transmission rates using MACsec.

5 Open Fronthaul Experimental Results

5.1 Packet Size Analysis

We repeat the experiment described in Sec. 4.2 over the Open Fronthaul interface and plot the processing delay without and with MACsec in Fig. 11. It is immediately clear that using MACsec on the Open Fronthaul interface increases delay and has a greater impact on larger packets than smaller packets. Using MACsec, with or without encryption, increases the delay for small packets by 39 µs. For the maximum packet size of 9000 Bytes, the processing delay due to MACsec with no encryption increases to 153 µs. However, the cost of using encryption increases faster than that for MACsec without encryption, reaching a total increase of 218 µs.

O-RU Cat	O-DU Category A	B	C	D	E	F	G	H	I	J	K	L	M	N
O	3000	399	379	359	339	319	299	279	259	239	219	199	179	159
P	2949	348	328	308	288	268	248	228	208	188	168	148	128	108
Q	2929	328	308	288	268	248	228	208	188	168	148	128	108	88
R	2909	308	288	268	248	228	208	188	168	148	128	108	88	68
S	2889	288	268	248	228	208	188	168	148	128	108	88	68	48
T	2869	268	248	228	208	188	168	148	128	108	88	68	48	28
U	2849	248	228	208	188	168	148	128	108	88	68	48	28	8
V	2829	228	208	188	168	148	128	108	88	68	48	28	8	0
W	2809	208	188	168	148	128	108	88	68	48	28	8	0	0
X	2789	188	168	148	128	108	88	68	48	28	8	0	0	0
Y	2769	168	148	128	108	88	68	48	28	8	0	0	0	0
Z	2749	148	128	108	88	68	48	28	8	0	0	0	0	0

TABLE 4: The O-RAN ALLIANCE WG4 specified maximum transmission delays in µs. For our system the RU/DU combinations in the red region support MACsec with encryption, the grey region support MACsec without encryption, and the blue region is supported without any MACsec.

Considering the latency requirements discussed in Sec. 3.2 for the Open Fronthaul and the observed large frame sizes, it is evident that, for certain systems, **the use of MACsec may significantly impact Open Fronthaul traffic**. The O-RAN ALLIANCE specifies the minimum and maximum latency supported by different equipment combinations for RU and DU, as detailed in [21] and reproduced in Table 4. For any system, the chart will exhibit three distinct regions, contingent on the configuration and use of MACsec. The red region of the chart represents RU/DU combinations that support MACsec using encryption in the system we used to evaluate performance. The grey region, signifying MACsec without encryption, broadens the range of RU/DU combinations at our disposal. Finally, the blue region introduces a substantial number of additional combinations that the considered evaluation system can accommodate without using MACsec altogether.

It is important for researchers and system engineers to understand the RU and DU capabilities and requirements and the cost of enabling MACsec with or without encryption before deploying systems. This work demonstrates that there are combinations of RU/DU that support MACsec, though the number of combinations is significantly smaller than with no security. Therefore, security impacts the size of the feasibility region of certain O-RAN deployments, and it is fundamental to understand the security-latency trade-off prior to deployment.

5.2 Throughput Analysis

We repeat the same experiment described in Sec. 4.3 for the Open Fronthaul. The maximum throughput for our local emulation environment is approximately 2300 Mbps, as seen in Fig. 12. Based on these results we again conclude that **CPU utilization is a key factor when adding MACsec.** However, unlike in the case of the E2, the CPU utilization is not above 95%. This suggests there may be other limiting factors, which we discuss further in Sec. 6.3.

We also study the impact of total throughput on delay specifically for the Open Fronthaul. For this experiment, we again use iperf3 to generate increasing traffic load. Simultaneously, we use ping with two different fixed-size packets to evaluate the RTT. We repeat the experiment with small (300 Byte) and large (8172 Byte) packets. We subtract the known transmission and propagation delays. In this experiment, unlike the previous ones, there may be queuing delays at the NIC as the iperf3 and ICMP traffic compete for the same physical interface. The RTT is therefore given by $RTT = 2 (D_{proc} + D_{queue})$. However, in Sec. 2.3 we show that the added queuing delay is negligible when the queue arrival rate is less than 9.78 Gbps. Thus, we can again approximate $RTT \approx 2 D_{proc}$ and solve for the added processing delay.

Fig. 13 shows the results of this experiment and illustrates the impact of throughput on delay. First, we observe that packet size tends to have less impact than the overall transmission rate. Second, we again observe that MACsec with encryption has a higher cost, in terms of increased latency, compared to MACsec without encryption. While both methods achieve a similar maximum throughput, the

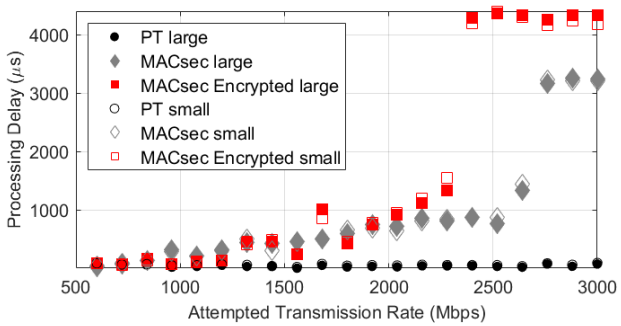

Fig. 13: Open Fronthaul processing delay as a function of attempted transmission rate.

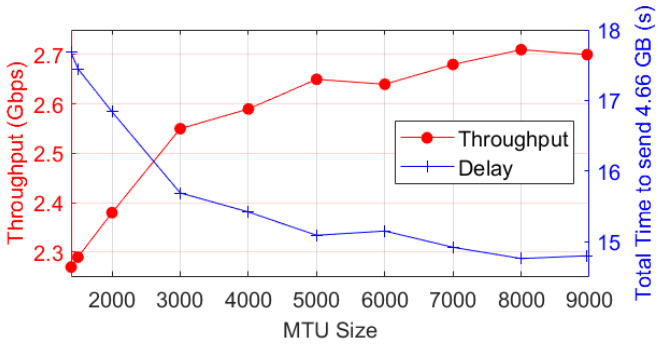

Fig. 14: MTU size impacts throughput (red) and delay (blue) in our deployed system using MACsec with encryption. A larger MTU provides better performance.

maximum latency for MACsec with encryption is about 4.3 ms while the maximum latency for MACsec without encryption is about 3.2 ms. In other words, **the specific MACsec configuration is a key factor in determining Open Fronthaul throughput and latency.**

5.3 MTU Analysis

The MTU of a network path is determined by the minimum MTU supported by any device along that path. This becomes critical when analyzing the Open Fronthaul. Not only could a disaggregated RU and DU be deployed in physically separate locations, but a single entity may not control the entire network path between them.

We designed an experiment to better understand the impact of different MTU sizes on the Open Fronthaul traffic. We send a large file, 4.66 GB, using iperf3 over our emulated Open Fronthaul interface using MACsec with encryption enabled. We chose a large enough file to saturate the link for around 15 seconds based on the maximum observed throughput of approximately 2.5 Gbps. We repeat this experiment while incrementing the MTU size from 1400 Bytes to 9000 Bytes. We record the total time to complete the transmission as well as the average throughput for the duration of the transmission.

Fig. 14 shows the results of our MTU analysis. As the MTU increases, the time to send a large file decreases and the effective throughput increases. Increasing the MTU size from 1400 Bytes to 9000 Bytes improves performance (reduces delay) by approximately 20%. From these results, we conclude that, for the large packet sizes and high data rates observed in the Open Fronthaul, **MTU size is a critical factor.**

6 Cost of Security Key Principles

Although the results presented so far show trends and phenomena that are transferable to any deployment configuration, it is important to point out that the exact values of all these results depend heavily on the specific system resources and network topology considered. In the following, we make an effort to extend our analysis to a more general configuration. Specifically, we present four key principles from lessons that we learned to enable researchers and system architectures to build security by design in O-RAN systems.

6.1 Sufficient Compute Resources

Sufficient processing power is one of the key trade-offs required to enable encryption. Generalizing Table 2, we can conclude that for any system operating over Gigabit Ethernet, or faster, $D_{trans} << D_{proc}$. From the calculations in Sec. 2.3, we can see that for any network where the total distance between the base station and near-RT RIC is in the order of tens of kilometers or less, $D_{prop} << D_{proc}$. We are confident that any additional queuing delays due to encryption, D_{que} 0 for nearly all conditions as shown in Sec. 2.3. Therefore, for most O-RAN systems, the total delay *cost of encryption*, D_{total} can be approximated as D_{total} D_{proc}.

Even if these assumptions are not universally applicable to a specific O-RAN system, adding encryption will impact the processing delay. Any disaggregated gNB component must have enough CPU resources to manage all of its explicit functions. While 3GPP standards allow for the use of secure gateways to deal with encryption, O-RAN standards do not. This means that as the total traffic over the E2 and Open Fronthaul interface (and other encrypted interfaces) increases, the CPU resources needed for encryption alone will increase. This could potentially impact the scalability and costs of gNB components, especially the DU. System designers can choose to add dedicated hardware for the encryption to offload the CPU burden, increase the total compute resources, or set strict limits on the amount of traffic that can be sent over O-RAN open interface. Therefore, **system engineers must understand the amount of traffic expected across a given interface and include the overhead of encryption for that level of traffic in their compute and hardware acceleration budget**.

6.2 Specific Encryption Algorithms

The fundamental protection provided by IPsec is through the encryption algorithm used to secure the payload. A complete discussion of all possible encryption algorithms (for example StrongSwan, supports 49 different encryption algorithms [45]) is beyond the scope of this paper. Some of these only perform encryption and require a separate integrity hashing algorithm, while others are Authenticated Encryption with Associated Data (AEAD) algorithms that do not require separate integrity hashing algorithms. Even

limiting the scope of algorithms to those that are currently known to be secure, there is a huge array of options. However, as seen in Fig. 9, choosing the specific encryption algorithm is incredibly important.

In contrast, MACsec uses a standard encryption algorithm but offers two modes: plain text payload or encrypted payload. As seen in Sec. 5.1 and 5.2 this distinction makes a significant difference in the processing delay. System engineers should make careful decisions about which method to use for what traffic. For example, the U-plane traffic is already encrypted by the PDCP layer between the UE and CU. In other words, the U-plane traffic crossing the Open Fronthaul already has confidentiality. Using MACsec without encryption for U-plane traffic will provide the other necessary security functions (authentication, integrity, and replay protection) at a lower cost. On the other hand, the synchronization and management plane may not provide confidentiality at other layers, requiring the use of MACsec with encryption.

While the O-RAN ALLIANCE guidance requires similar security functions to be provided for most interfaces, **the specific security protocol and encryption algorithm used must be carefully chosen to meet both security and system performance requirements.**

6.3 I/O Bottlenecks

In Fig. 12 we see that the actual throughput reaches a plateau while the CPU utilization is only around 80%. Also, as discussed in Sec. 4.3, while we observe similar patterns with all encryption algorithms, for some the CPU utilization plateaus around only 50%. While this observation does not diminish the importance of optimizing the specific protocol and encryption algorithm used, it does indicate another factor plays a key role; namely the I/O operations between kernel space and user space in Linux.

Network packet processing for encrypted traffic in the Linux kernel can be significantly slow due to context switching associated with system calls and transitional copy operations in packet traversal through all network layers [46]. Specifically for StrongSwan, there is a significant bottleneck from user/kernel space context switching. Prior work [46] achieves roughly a 3.5x increase in throughput and a 2.5x decrease in latency by removing or optimizing these I/O operations. Other methods of improving I/O bottlenecks include offloading workload from the CPU to the NIC, reducing the number of interrupts generated by incoming traffic, optimizing the basic Linux Kernel network stack, and moving network functions entirely to user space such as with DPDK [47]. **It is vital for O-RAN system designers, working with distributed and compute constrained devices, to understand the existing network stack I/O bottlenecks and properly optimize each component.**

These first three principles are often highly dependent on each other. For example, our Open Fronthaul system using the Mellanox ConnectX-6 does not support hardware acceleration of MACsec. However, it does support hardware acceleration of IPsec when using AES-GCM. It also supports TLS data-path offloading and keeping all the TLS functions in the U-plane. A comprehensive systems engineering approach is essential for optimizing the interplay among these considerations.

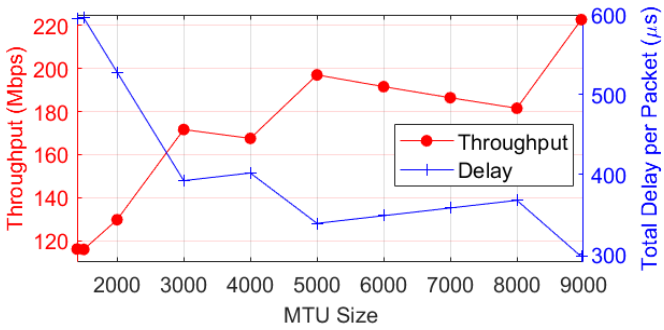

Fig. 15: Modeled impact of MTU size on throughput (red) and delay (blue) using MACsec with encryption. A larger MTU provides better performance.

6.4 MTU size

We clearly see that a larger MTU provides better performance in our emulation environments. Much of this improvement comes from the reduction in overhead by using less total packets. For example, with a payload of exactly 8000 Bytes, the amount of overhead with an MTU size of 4000 Bytes will be double the overhead compared to a network with an MTU of 8000 Bytes. The experiments we perform involve transferring a large, fixed amount of data. While this is a good approximation to the Open Fronthaul and provides measurable metrics, it does not exactly match the nature of the Open Fronthaul interface. Instead, we observe that the Open Fronthaul sends fixed packet sizes (7678 Bytes) at regular intervals.

We built a model based on our observations to understand the impact of MTU size on regular, fixed-size packets. We use 8192 Bytes as the payload size because it is the maximum eCPRI payload size. We calculate the total overhead based on the number of packets the payload must be fragmented into and calculate the delay of each packet based on packet size using Eq. 1 and Fig. 11. We calculate the throughput as the total bits sent over the total delay. The results of the model are seen in Fig. 15.

The general trends of our model closely match our experimental results. However, there is one key difference. Our experiment shows continuous improvement in performance with increasing MTU size. For a fixed file size, this intuitively makes sense as it minimizes the total amount of overhead compared to the total file size. In contrast, our model shows that setting the MTU to 5000 Bytes gives better performance than 8000 Bytes. In other words, for fixed size packets transmitted with regular frequency, if fragmentation must happen, it is best to split the payload as evenly as possible. This is because the processing delay increases significantly with packet size, as shown in Fig. 11, while the other delays are essentially constant. Splitting a fixed amount of data into two equal chunks results in lower processing delay than one large and one small chunk.

In either case, the best option is to ensure the entire network path has an MTU of 9000 Bytes. However, it is likely that a single entity will not control the entire network path between the RU and DU in future O-RAN systems. In this case, **system engineers must determine the maximum MTU of the network path and carefully select the optimal**

MTU size for the distributed RU and DU.

7 Conclusions

5G stands as a critical strategic technology, offering enhanced performance and data-driven intelligent capabilities [13]. O-RAN, driven by its open interfaces and adaptable RICs, plays a pivotal role in harnessing these capabilities. It empowers the customization of radio resource management through powerful ML-driven xApps/rApps while exposing valuable telemetry. Given the paramount importance of safeguarding O-RAN's open links, this article conducts a comprehensive experimental and theoretical analysis of the E2 and Open Fronthaul interfaces, aligning with O-RAN specifications [12, 21, 23, 24, 26], using the world's largest emulator, Colosseum [15] and a private, 5G and O-RAN compliant network.

We implement various security protocols (IPsec for E2, MACsec for Open Fronthaul) and employ diverse encryption techniques to assess their impact on critical network performance metrics. Our analysis covers key parameters like processing delay and throughput, accompanied by detailed quantitative insights. Notably, we find that the cost of securing O-RAN for the E2 interface remains low, while MACsec is more likely to exert a significant impact on the Open Fronthaul interface. We conclude that system designers must ensure O-RAN disaggregated nodes possess sufficient computing resources, make judicious protocol and encryption algorithm selections, optimize I/O bottlenecks, and manage local MTU settings with an understanding of the end-to-end network MTU.

Although significant progress has been made, there is still substantial work to fully comprehend O-RAN security and the associated costs. Ensuring security in a multi-vendor xApp environment is a largely open problem with possible hidden expenses. One potential solution is adopting a zero-trust approach [5, 13], which presents a promising framework but has yet to be implemented. Continued efforts are required to shape the future of secure O-RAN systems.

References

[1] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[2] O-RAN Working Group 1, "O-RAN Architecture Description 5.00," ORAN.WG1.O-RAN-Architecture-Description-v05.00, Tech. Rep., July 2021.

[3] J. Thaliath, S. Niknam, S. Singh, R. Banerji, N. Saxena, H. S. Dhillon, J. H. Reed, A. K. Bashir, A. Bhat, and A. Roy, "Predictive Closed-Loop Service Automation in O-RAN Based Network Slicing," *IEEE Communications Standards Magazine*, vol. 6, no. 3, pp. 8–14, Sep. 2022.

[4] C. Shen, Y. Xiao, Y. Ma, J. Chen, C.-M. Chiang, S. Chen, and Y. Pan, "Security Threat Analysis and Treatment Strategy for ORAN," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2022, pp. 417–422.

[5] K. Ramezanpour and J. Jagannath, "Intelligent zero trust architecture for 5G/6G networks: Principles, challenges, and the role of machine learning in the context of O-RAN," *Computer Networks*, p. 109358, 2022.

[6] A. S. Abdalla, P. S. Upadhyaya, V. K. Shah, and V. Marojevic, "Toward Next Generation Open Radio Access Networks–What O-RAN Can and Cannot Do!" *IEEE Network*, 2022.

[7] J. Groen, S. D'Oro, U. Demir, L. Bonati, M. Polese, T. Melodia, and K. Chowdhury, "Implementing and Evaluating Security in O-RAN: Interfaces, Intelligence, and Platforms," *arXiv preprint arXiv:2304.11125*, 2023.

[8] J. Y. Cho and A. Sergeev, "Secure Open Fronthaul Interface for 5G Networks," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ser. ARES 21. New York, NY, USA: Association for Computing Machinery, 2021.

[9] W. Tiberti, E. Di Fina, A. Marotta, and D. Cassioli, "Impact of Man-in-the-Middle Attacks to the O-RAN Inter-Controllers Interface," in *2022 IEEE Future Networks World Forum (FNWF)*, 2022, pp. 367–372.

[10] S.-H. Liao, C.-W. Lin, F. A. Bimo, and R.-G. Cheng, "Development of C-Plane DoS Attacker for O-RAN FHI," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, ser. MobiCom '22, 2022, p. 850–852.

[11] J. Boswell and S. Poretsky, "Security considerations of Open RAN," *Stockholm: Ericsson*, 2020.

[12] O-RAN Working Group 3, "Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles," ORAN.WG3.E2GAP-v02.02, Tech. Rep., July 2022.

[13] D. of Defense, "5G Strategy Implementation Plan," Department of Defense, Tech. Rep., December 2020, accessed: 2024-02-15. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1118833.pdf

[14] D. Dik and M. S. Berger, "Transport security considerations for the open-ran fronthaul," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 253–258.

[15] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder *et al.*, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 2021, pp. 105–113.

[16] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open-RAN Gym: An Open Toolbox for Data Collection and Experimentation with AI in O-RAN," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 518–523.

[17] L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "SCOPE: An open and softwarized prototyping platform for NextG systems," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 415–426.

[18] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, 2022.

[19] D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, R. S. da Silva, A. Kelkar, C. Dick, S. Basagni, J. M. Jornet, T. Melodia, M. Polese, and D. Koutsonikolas, "An Open, Programmable, Multi-vendor 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface," 2023.

[20] J. Groen, B. Kim, and K. Chowdhury, "The Cost of Securing O-RAN," in *IEEE International Conference on Communications (ICC)*, 2023.

[21] O-RAN Working Group 4, "O-RAN Fronthaul Control, User and Synchronization Plane Specification v12," O-RAN.WG4.CUS.0-R003-v12.00, Tech. Rep., June 2023.

[22] P. S. Upadhyaya, A. S. Abdalla, V. Marojevic, J. H. Reed, and V. K. Shah, "Prototyping Next-Generation O-RAN Research Testbeds with SDRs," *arXiv preprint arXiv:2205.13178*, 2022.

[23] O-RAN Working Group 11, "Security Requirements Specifications," O-RAN.WG11.Security-Requirements-Specification.O-R003-v06.00, Tech. Rep., June 2023.

[24] ——, "Security Protocols Specifications," ORAN.WG11.Security-Protocols-Specifications-v04.00, Tech. Rep., July 2022.

[25] O-RAN Working Group 4, "O-RAN Management Plane Specification 12.0," O-RAN.WG4.MP.0-R003-v12.00, Tech. Rep., June 2023.

[26] O-RAN Working Group 5, "Transport specification," ORAN.WG5.Transport.0-v01.00, Tech. Rep., March 2020.

[27] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Aug 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8446.txt

[28] "IP Authentication Header," Dec 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4303.txt

[29] "IP Encapsulating Security Payload (ESP)," Dec 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4303.txt

[30] S. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W. Ritchey, and S. R. Sharma, "Guide to IPsec VPNs:." *NIST Special Publication*,