The Cost of Securing O-RAN

Joshua Groen
Institute for the Wireless IOT
Northeastern University
Boston, MA
groen.j@northeastern.edu

Brian Kim
Institute for the Wireless IOT
Northeastern University
Boston, MA
br.kim@northeastern.edu

Kaushik Chowdhury
Institute for the Wireless IOT
Northeastern University
Boston, MA
k.chowdhury@northeastern.edu

***Abstract*—A promising vision for the emerging next generation of cellular networks is one that embraces openness, intelligence, virtualization, and distributed computing. The Open Radio Access Network (O-RAN) framework is making significant strides toward these goals and is already seeing prototype deployments in academia and industry. While there is general consensus that this technology may disrupt the status quo by eliminating vendor lock-ins, there are serious questions about the security implications in such dis-aggregated networks. Indeed, securing data and controlling interfaces must be a core consideration in the design of O-RAN and cost/benefit tradeoffs need to be rigorously analyzed, given the short time-scales of wireless operation. In this paper, we undertake the first systematic study on the impact of encryption on a critical O-RAN interface (called 'E2') connecting the base station to a near-real time radio intelligence controller using an implementation on the Colosseum radio frequency (RF) emulator. The contributions of this paper include quantitative measurements of added latency and CPU utilization due to encryption on the E2 interface that could impact data acquisition and machine learning models. In our experiments we found encryption adds $\leq 50\mu s$ of delay and CPU utilization limits throughput to approximately 500 Mbps. We also include a theoretical model to extend this study to other O-RAN implementations beyond the emulation environments of the Colosseum.**

***Index Terms*—Security, 5G, O-RAN, emulation, encryption**

I. Introduction

The Open Radio Access Network (O-RAN) framework aims to transform 5G and beyond cellular radio access networks (RAN) into open, intelligent, virtualized, and fully interoperable RANs. O-RAN deployment defines dis-aggregated and virtualized components connected through open and standardized interfaces [1]. This paradigm change is a potential enabler of data-driven optimization, closed-loop control, and automation [2], breaking vendor-locked closed networking architectures. Alongside traditional telecommunication operators who are active members of the O-RAN Alliance, the US Department of Defense is actively promoting open interfaces in both the RAN and the 5G Core that will allow for more competition and innovation [3].

• **Need for data-relaying within O-RAN:** Recently, several studies have described ways to optimize the open framework through machine learning (ML), leveraging standardized APIs that allow this functionality. [4] demonstrates a software-defined radio (SDR) based testbed built on srsRAN. The authors deploy an O-RAN compliant E2 Agent and show how data can be passed across this interface. They also deploy two basic xApps, customizable microservices, to demonstrate the capability of the near-real time (RT) RAN intelligent controller (RIC) to both gather metrics from the distributed base station (gNB) and make changes to physical resource assignment. SCOPE [5] is an open-source framework with plug-and-play containers for instantiating O-RAN infrastructure including the gNB and user equipment (UE). It includes a data collection module and a set of open APIs to allow users to control network element functionalities in real time. The building blocks provided in SCOPE are extended in [6], where the authors demonstrate three different xApps that utilize ML to provide closed-loop network control. All these capabilities are integrated and discussed in [7]. The "OpenRAN Gym" provides an open toolbox enabling end-to-end design, data collection, and testing. This framework is deployed in Colosseum, the world largest radio frequency (RF) emulator, allowing users to create interesting scenarios at-scale that utilize large amounts of data at the near-RT RIC. In a recently conducted survey [8], the authors identify four critical features that are important for O-RAN deployment: end-to-end security, deterministic latency, physical layer real-time control, and testing of ML-based RAN control applications.

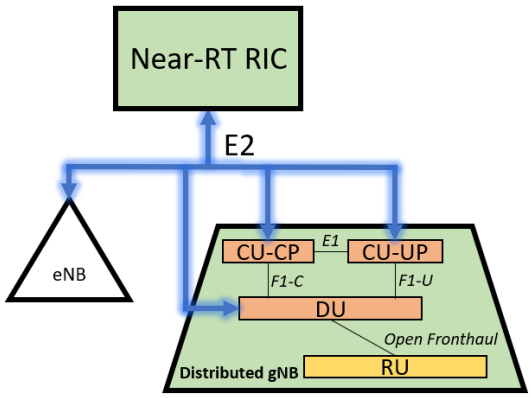


Fig. 1: E2 interface connects near-RT RIC to eNB and nearly every component in the next generation distributed gNB.

• **Challenges in data-relaying:** The introduction of open interfaces that carry data between dis-aggregated components introduces new threats [9], [10]. In fact, [8], [10] point out that one of the primary classes of threats arises from improper or missing ciphering of the data sent across these open interfaces.

978-1-5386-7462-8/23/$31.00 ©2023 IEEE

In this paper, we study an interface that enables intelligent and data-driven optimization, called the E2 interface, which exists between the distributed gNB elements and the near-RT RIC. The O-RAN Alliance recognized this new threat vector and published guidance in [11] to ensure that the E2 interface supports confidentiality, integrity, replay protection, and data origin authentication.

• **Motivation of our study:** While there is general consensus that security is essential to the deployment of 5G O-RAN infrastructure [3], [8]–[12], to our knowledge there has been no systematic study of the impact of securing the E2 interface in an O-RAN implementation. It is vital that an informed and risk-based approach is taken to adequately address security risks in O-RAN, while recognizing that any method for enhancing security, such as adding encryption, comes at a performance cost [12]. To extract actionable insights, we thoroughly test and analyze the effects of adding encryption to the E2 interface using the OpenRAN Gym framework built on SCOPE and ColO-RAN [5]–[7].

Our main contributions are as follows:

- We perform the first ever experimental analysis of the effects of adding O-RAN compliant encryption to the O-RAN E2 interface using Colosseum. [Section IV-A]
- We validate at-scale a theoretical framework for calculating total network delay for O-RAN based distributed functional units. [Section V-A]
- We extend these results by developing a general framework for understanding the cost of encryption for future O-RAN system deployments enabling researchers and engineers to build security from the start. [Section V-E]

The rest of the paper is organized as follows. Section II gives a brief overview of key O-RAN principles. Section III describes our emulation environment and our experimental procedures are detailed in Section IV. We analyze our results in Section V and conclude in Section VI.

II. O-RAN Background

We first review the implications of the four foundational principles that O-RAN is built on: dis-aggregation; intelligent control with the RICs; virtualization; and open interfaces using the excellent tutorial paper [2].

A. O-RAN Principles and Framework

- *Dis-aggregation:* O-RAN dis-aggregation splits base stations into multiple functional units including: the Central Unit (CU), a Distributed Unit (DU), and a Radio Unit (RU). The CU is split further into the Control Plane (CP) and the User Plane (UP). This logical split allows different functions to be performed at different locations and on different platforms across the network. These functional units can be seen in Fig. 1.
- *RAN Intelligent Control:* The main function of the RIC is to use Key Performance Metric (KPM) data and leverage ML algorithms to determine and apply control policies and actions. While there are two RICs in the O-RAN framework, the non-RT RIC and the near-RT RIC, we focus on the near-RT RIC in this paper as it controls and optimizes resources via fine-grained data collection and actions over the E2 interface on a time scale between $10ms$ and $1s$. As Fig. 1 demonstrates, the near-RT RIC interfaces with nearly every functional unit of the distributed gNB and legacy base stations (eNBs). The near-RT RIC also hosts xApps that can be used to perform radio resource management [2].
- *Virtualization:* All the of components shown in Fig. 1 can be deployed as virtualized infrastructure. This virtualization enables a decoupling between hardware and software components, standardization of hardware, sharing of hardware among different tenants, and automated deployment of RAN functions. While decoupling hardware and software creates an open environment for faster development, it also raises encryption requirements for data traversing the E2 interface. Thus, it is important to fully understand the capabilities of the distributed hardware and expected computational load to ensure the right distribution of resources.
- *Open Interfaces:* The open interfaces are one of the keys to overcome the traditional RAN black box approach as they expose network parameters to the RICs and enable data analytic and ML-enabled control. The major interfaces include: the O1 interface, which is the primary interface with the non-RT-RIC responsible for enabling operations and maintenance; the A1 interface which connects the two RICs and is used for deploying policy-based guidance; the Open Fronthaul which connects a DU to one or multiple RUs inside the same gNB [13]; and the E2 interface, which is the key interface that connects the near-RT RIC to the RAN (see Fig. 1). 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 distributed gNB. A comprehensive discussion of these and additional interfaces can be found in [2], [4], [8], [11]–[13].

III. O-RAN Implementation in Colosseum

Colosseum is the world's largest wireless network emulator with hardware in-the-loop [14]. It 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 (SRN). Each SRN consists of a 48-core Intel Xeon E5-2650 CPU with an NVIDIA Tesla k40m GPU connected to an USRP X310 SDR. Users can deploy custom protocol stacks by deploying Linux Containers (LXC) to the bare-metal SRNs.

The Colosseum environment also provides several RF scenarios created through its Massive Channel Emulator (MCHEM) and managed by simple APIs. This enables users to test custom protocol stacks in a wide range of both artificially constructed (such as a fixed path loss) and realistic RF scenarios built from field observations.

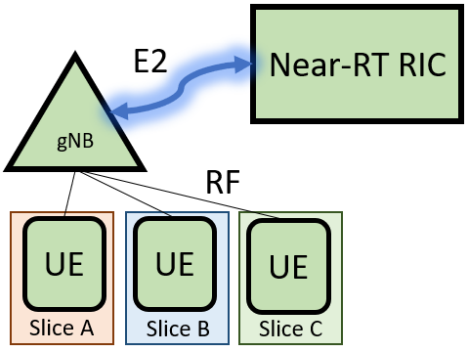


Fig. 2: O-RAN system consisting of three UEs, a gNB, and the near-RT RIC. Each component is implemented in an LXC on top of an SRN within Colosseum.

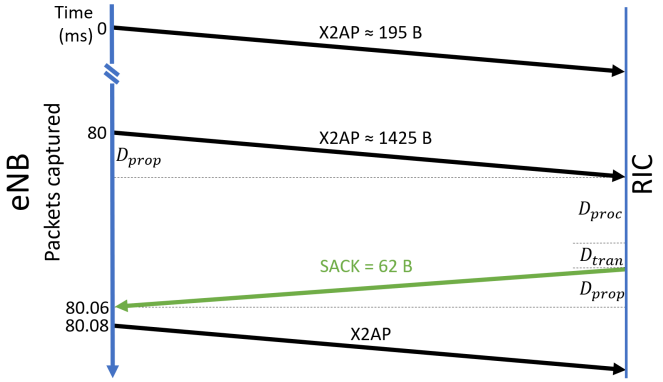


Fig. 3: E2 traffic displays a common pattern of one small X2AP packet then one large X2AP packet followed by a SACK as seen in this flow graph.

We utilize the srsRAN based SCOPE [5] framework to implement a softwarized RAN for both the gNB and multiple 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 [6] framework for the near-RT RIC implementation. ColO-RAN is a minimal version of the O-RAN Software Community near-RT RIC (Bronze release). The ColO-RAN LXC includes a minimal near-RT RIC in the form of several Docker containers to provide the functions of the near-RT RIC. In particular, it provides the E2 interface functionality to connect to the RAN nodes for data collection and control, and a sample xApp that collects basic KPMs from the gNB.

To enable encryption, we add the strongSwan open source IPsec-based VPN to both the SCOPE and ColO-RAN LXCs. The full IPsec configuration is described in paragraph IV-B. We also add several simple scripts to automate data collection on E2 interface performance.

IV. Experiment Overview

A. System Overview

Our experimental system (see Fig. 2) is composed of five blocks: three UEs, a gNB, and the near-RT RIC. Each block is implemented inside an LXC on a separate SRN. The UEs are connected to the gNB over an emulated RF channel where each UE is assigned an unique slice representing the three main use cases for 5G: enhanced Mobile Brodband (eMMB), Ultra Reliable Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC) [15]. Each slice has its own traffic pattern, physical resource block assignment, and scheduling policy. The gNB is connected to the near-RT RIC over a wired 10 Gbps backbone network. We implement the sample KPM monitoring xApp from [6] that periodically polls the gNB for 6 KPMs for each UE. This generates $\leq 200Kbps$ of traffic on the E2 interface.

We capture all traffic traversing the E2 interface at the gNB for over 20 minutes. Stream Control Transmission Protocol (SCTP) is used as the transport layer protocol for all traffic. Fig. 3 illustrates a typical example of the captured traffic. First, the gNB sends a small amount of data using the X2 Application Protocol (X2AP). Next, the gNB sends a large amount of data using X2AP over SCTP. Finally, the near-RT RIC responds with a fixed size, 62 Byte, Selective Acknowledgment (SACK). This pattern is consistent because SCTP specifies a SACK should be generated for every second packet received [16]. While we are able to capture the elapsed time between packet captures, we do not know when the gNB starts processing or transmitting a packet. However, we can observe the delay between the transmission of the large X2AP packet and the reception of the SACK at the gNB. For these reasons, we first study the effect of encryption on the SACK.

B. Securing the E2 Interface

After establishing a baseline performance without encryption, we add O-RAN compliant encryption as specified in [11] to the E2 interface. We implement IPsec with Encapsulating Security Payload (ESP) in tunnel mode. Tunnel mode creates a new IP header for each packet and protects the integrity of both the data and original IP header for each packet [17]. We use AES-256 for encryption and SHA2-256 for the authentication hash function. AES-256 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 [18]. SHA2-256 uses eight 32-bit words and performs 64 transformation rounds to compute a 256 bit hash [19]. 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 [11]. With this configuration, IPsec adds at least 57 Bytes of overhead to each packet. However, because both AES-256 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 in Section IV-A, poll the gNB for the same KPMs, and again capture the traffic traversing the E2 interface at the gNB. Fig. 4 shows the distribution of delay times for both un-encrypted (plain text or PT) and

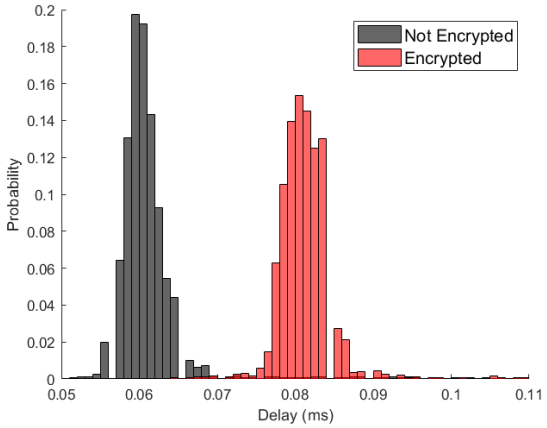


Fig. 4: SACK delay distribution with and without encryption.

encrypted (cypher text or CT) SACKs. It is immediately clear that encryption adds, on average, approximately $22\mu s$ of delay.

V. Analysis and Further Results

We immediately observe there is some cost to add encryption to the E2 link. It is essential for O-RAN researchers, engineers, and system architects to fully understand both the cause and the impact of any extra overhead.

A. Types of Delay

In packet switched networks there are four primary sources of delay at each node along the path: *queuing* delay, *propagation* delay, *transmission* delay, and *nodal processing* delay [20]. Specifically, the total delay can be expressed as

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

- *Queuing Delay:* In our test environment there is virtually no competing traffic, so it is safe to assume $D_{que} = 0$. In more complex networks, this may not be true. However, even in congested networks, the queuing delay will be essentially constant with or without encryption.
- *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 $2 \times 10^8 m/s$ [20]. For our environment, we assume a length of $100m$ giving $D_{prop} = 0.05\mu s$. 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 I lists the calculated transmission delays, with and without encryption, based on the average packet length of the three types of packets we observe.
- *Processing Delay:* Typically the processing delay is defined as the time required for intermediate nodes to examine the packet header and determine where to direct the packet, though it can also include other factors such as checking for bit-level errors [20]. For this analysis, it makes sense to include the encryption delay in the processing delay because it is essential to pass the payload to lower or higher network layers.

Packet Type	Plain Text	Cypher Text
SACK	$0.0496\mu s$	$0.1104\mu s$
Short X2AP	$0.1560\mu s$	$0.2040\mu s$
Long X2AP	$1.140\mu s$	$1.188\mu s$

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

B. SACK Analysis

The SACK packet is ideal for an initial analysis because the packet size is fixed ($PT = 62B$, $CT = 138B$) and the total delay we observe in our experiments is very tightly grouped as seen in Fig. 4. The total average delay we observe for the PT SACK is 61.12 μs, while the CT SACK is 82.64 μs. From Fig. 3, we can see that the total delay for a SACK can be expressed as $D_{total}^{SACK} = 2 \times D_{prop} + D_{trans} + D_{proc}$. Given the total delay, the transmission delays in Table I, and the calculated propagation delay (Section V-A), 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. **With low traffic load ($\leq 200Kbps$) and small packets ($\leq 138B$), encryption has no meaningful impact on E2 interface traffic.**

C. Packet Size Analysis

Section V-B also shows that the processing delay accounts for at least 99.75% of the total delay for the 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 we do not know the exact start times for transmitting the long X2AP packets, 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 48 Byte payload for the ping and increment the size by 5 Bytes until we reach 1486 Bytes. After the ethernet frame (14 Bytes) is added, this fully captures the range of packet sizes we observe in our experiments (62 Bytes to 1500 Bytes). For each step size, we send 10 pings with $250ms$ between each ping. The RTT is expressed as

$$RTT = 2 \times (D_{proc} + D_{trans} + D_{prop}). \quad (2)$$

Given that $2 \times D_{prop} = 0.1\mu s$, $2 \times D_{trans} \leq 2.4\mu s$, and $RTT \geq 50\mu s$, we can approximate equation (2) as $D_{proc} \approx \frac{RTT}{2}$. From the results of the experiment in Fig. 5, we observe that the processing delay increases with packet length

when sending encrypted traffic. However, the processing delay difference between CT and PT is $\triangle D_{proc} \leq 50\mu s$ for all tested packet sizes. We can conclude that **for all packet sizes from 62 B to 1500 B, encryption has minimal impact on E2 traffic**.

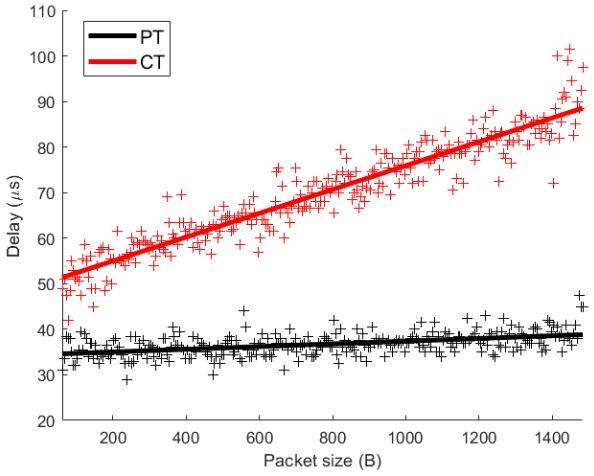


Fig. 5: Processing delay as a function of packet size for PT and CT traffic.

D. Throughput Analysis

Finally, we 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. We start at 25 Mbps and increment the transmission rate. It is seen from Fig. 6 that the maximum encryption rate our system is capable of is approximately 500 Mbps.

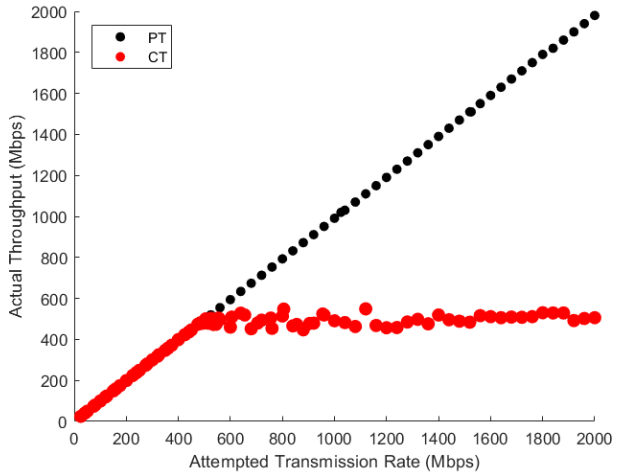
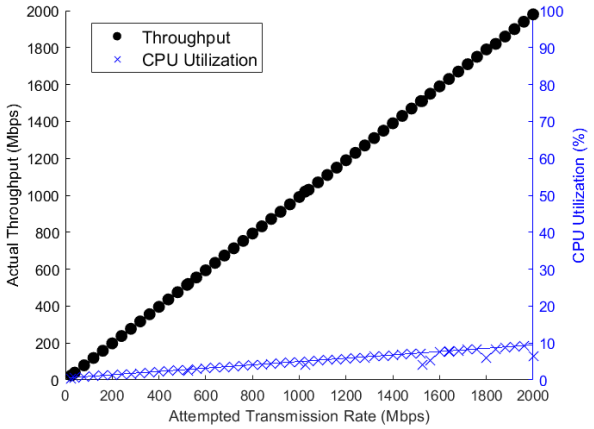
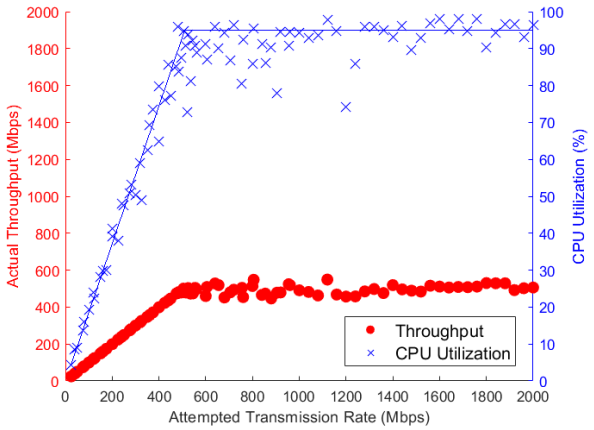


Fig. 6: Measured throughput as a function of attempted transmission rate for PT and CT traffic.

To understand the upper bound of the throughput with encryption seen in Fig. 6, we analyze CPU utilization. We capture CPU utilization on the gNB (sending node) for each attempted transmission rate using iperf3. Fig. 7 shows the results for both PT and CT. While CPU utilization does increase for both types of traffic, we can see that encryption is very CPU intensive. Fig. 7(a) shows that when sending PT, the CPU utilization increases proportional to $0.00365 \times T$ whereas Fig. 7(b) shows the CPU utilization for CT grows proportional to $0.2 \times T$ until it reaches saturation. Therefore, we conclude that encrypting all traffic increases CPU utilization by roughly two orders of magnitude compared to PT traffic. **CPU utilization is the limiting factor for encrypting traffic when the throughput is above 500 Mbps.**



(a) PT CPU utilization grows proportional to $0.00365 \times T$.



(b) CT CPU utilization grows proportional to $0.2 \times T$ until it reaches the maximum utilization allowable. After this threshold, throughput and CPU utilization remain constant.

Fig. 7: Actual throughput and CPU utilization as functions of attempted transmission rate for PT and CT traffic.

E. Cost of Security Framework

These results verify that processing delay is the dominant factor in our test system. To precisely extend these results to other systems, the specific system parameters would have to be known. However, it is possible to generalize these results and use them as a guide when designing future O-RAN systems.

Generalizing Table I, we can conclude that for any system operating over GigabitEthernet, or faster, $\triangle D_{trans} << \triangle D_{proc}$. From the calculations in Section V-A, we can see that for any network where the total distance between the base station and near-RT RIC is on the order of tens of kilometers or less, $\triangle D_{prop} << \triangle D_{proc}$. Section V-A holds for any system because intermediate nodes do not perform decryption, so $\triangle D_{que} \approx 0$. Therefore, for most O-RAN systems, the total overhead *cost of encryption*, $\triangle D_{total}$ can be approximated as

$$\triangle D_{total} \approx \triangle D_{proc}. \quad (3)$$

Even in the rare case where these assumptions do not hold for a particular O-RAN system, the added latency due to encryption will still primarily affect the processing delay. Equation (3) provides a useful framework for system designers and architects. Sufficent processing power is the key tradeoff required to enable encryption. Any dis-aggregated gNB component must have enough CPU resources to manage all of its explicit functions. However, as the total traffic over the E2 interface (and other encrypted interfaces) increases, the CPU resources needed for encryption alone will increase. 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 the E2 interface. In Colosseum, we must limit E2 traffic to 500 Mbps or less. More modern CPUs and encryption algorithms may increase that limit significantly. 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 budget.

VI. Conclusion and Future Work

5G is a critical strategic technology that offers higher performance and additional data-driven intelligent capabilities [3]. O-RAN enables these capabilities primarily through its open interfaces which expose telemetry and its RICs which are capable of hosting powerful ML-driven xApps to customize radio resource management. It is imperative that future O-RAN deployments provide sufficient protection to this E2 link in accordance with [11]. We conducted a thorough experimental and theoretical analysis and have shown that, for the E2 interface, the **cost of securing O-RAN is low**. However, system designers must ensure O-RAN dis-aggregated nodes have sufficient computing resources to handle both their explicit function and encryption overhead for the expected traffic load. Further, they must take steps to ensure that the actual traffic load does not exceed the system budget.

Significant work remains to fully understand how to secure O-RAN and any associated costs that must be planned for. There may remain significant hidden costs for securing other interfaces or for ensuring secure xApp development and deployment. Future work should examine other interfaces, such as the Open Fronthaul (see Fig. 1), which must operate with significantly lower latency and higher throughput. Additional work also needs to be done to ensure security in a multi-vendor xApp environment. [3], [10] call for a zero-trust approach to building out the O-RAN system. This framework offers a good starting point, but significant work needs to be done to actually implement these principles and thoroughly understand the Cost of Securing O-RAN.

Acknowledgment

Special thanks to Leonardo Bonati and Michele Polese for all the tips on implementing an O-RAN system in Colosseum.

References

[1] O-RAN Working Group 1, "O-RAN Architecture Description 5.00," ORAN.WG1.O-RAN-Architecture-Description-v05.00, Tech. Rep., July 2021.
[2] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *arXiv preprint arXiv:2202.01032*, 2022.
[3] "5G Strategy Implementation Plan," Department of Defense, Tech. Rep., December 2020.
[4] 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.
[5] 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.
[6] 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.
[7] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "OpenRAN 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] J. Boswell and S. Poretsky, "Security considerations of open ran," *Stockholm: Ericsson*, 2020.
[13] O-RAN Working Group 4, "O-RAN Fronthaul Control, User and Synchronization Plane Specification 7.0," ORAN-WG4.CUS.0-v07.00, Tech. Rep., July 2021.
[14] 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.
[15] V. K. Choyi, A. Abdel-Hamid, Y. Shah, S. Ferdi, and A. Brusilovsky, "Network slice selection, assignment and routing within 5G Networks," in *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, 2016, pp. 1–7.
[16] R. Stewart, "Stream control transmission protocol (RFC 4960)," Tech. Rep., 2007.
[17] S. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W. Ritchey, and S. R. Sharma, "Guide to IPsec VPNs:." *NIST Special Publication*, 2005.
[18] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray, "Advanced Encryption Standard (AES)," 2001-11-26 2001.
[19] Q. Dang, "Secure hash standard," 2015-08-04 2015.
[20] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach, Seventh Eddition*, 2017.