POLITECNICO DI TORINO
Repository ISTITUZIONALE

TNED: Trusted Network Environment for Devices

Original
TNED: Trusted Network Environment for Devices / Ciravegna, F., Bruno, G., Elorza Forcada, M.A., Pastor, A., Lioy, A.. - In: IEEE ACCESS. - ISSN 2169-3536. - 14:(2026), pp. 149964-149982. [10.1109/ACCESS.2026.3737900]

Availability:
This version is available at: 11583/3015878 since: 2026-10-06T08:28:35Z

Publisher:
IEEE

Published
DOI:10.1109/ACCESS.2026.3737900

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

07 November 2026

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 September 2026, accepted 23 September 2026, date of publication 28 September 2026,
date of current version 2 October 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3737900

RESEARCH ARTICLE

TNED: Trusted Network Environment for Devices

FLAVIO CIRAVEGNA[1], GIACOMO BRUNO[1], MATTIN ANTARTIKO ELORZA FORCADA[2], ANTONIO PASTOR[2], AND ANTONIO LIOY[1]

[1]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Turin, Italy
[2]CDO Unit, Telefónica Innovación Digital, 28050 Madrid, Spain

Corresponding author: Flavio Ciravegna (flavio.ciravegna@polito.it)

This work was supported in part by the Secure Platform for ICT Systems Rooted at the Silicon Manufacturing Process (SPIRS) Project through European Union's Horizon 2020 Research and Innovation Program under Grant 952622, and in part by the Project Security and Rights in CyberSpace (SERICS) through the National Recovery and Resilience Plan (NRRP) Italian Ministry of University and Research (MUR) Program funded by European Union—NextGenerationEU under Grant PE00000014.

ABSTRACT Advances in 5G connectivity, edge computing, and cloud-native platforms have driven network environments to new levels of scale and dynamism, challenging conventional security mechanisms for secure communication. For this reason, we introduce the Trusted Network Environment for Devices (TNED). The TNED is an architecture designed to enhance security through centralised management and hardware-based integrity verification. A key component of the TNED is the Centrally Controlled IPsec (CCIPS), an SDN-based framework that streamlines IPsec deployment by adopting an IKE-less model. This approach eliminates peer-to-peer handshakes and reduces tunnel setup complexity by pre-computing and distributing cryptographic material from a central controller. To further protect sensitive keys and configurations, the TNED architecture integrates Trusted Execution Environments (TEEs) and supports Remote Attestation. The implementation leverages two heterogeneous TEE frameworks, Keystone and Enarx, to demonstrate portability and cross-platform trust enforcement.

INDEX TERMS Trusted execution environment, IPsec, trusted computing, confidential computing, remote attestation, SDN, NFV.

I. INTRODUCTION

The rapid evolution of modern networking technologies, driven by the widespread adoption of 5G, edge computing, and cloud-native infrastructures, has significantly increased the scale, complexity, and dynamism of communication networks. These technological advancements have fundamentally reshaped how services are protected, deployed, and orchestrated across distributed infrastructures. In such a dynamic environment, the traditional static approaches to network security have proven to be increasingly inadequate, especially in ensuring data confidentiality, integrity, and authenticity.

IP Security (IPsec) is a well-established protocol suite for securing IP communications at the network layer. It enables encryption and authentication mechanisms that support secure communication channels across various scenarios, from site-to-site Virtual Private Networks (VPNs) to tunnels for mobile or cloud-based services. Nevertheless, conventional IPsec deployments depend on the Internet Key Exchange (IKE) protocol for negotiating Security Associations (SAs) and managing key exchanges. While IKE provides strong security assurances, it also introduces considerable overhead due to its complex handshake procedures, which involve multiple interactions between communicating peers. These characteristics make it less suitable for contexts involving constrained devices.

To address these limitations, we introduce the TNED, an architecture designed to provide secure communication through centralised management and hardware-based trust. At its core lies the Centrally Controlled IPsec (CCIPS), a novel approach for managing IPsec configurations in a scalable and efficient way. CCIPS leverages the principles of Software Defined Networking (SDN) and Network Function Virtualization (NFV) to achieve a clean separation between control and data planes, enabling centralised programmability and on-demand provisioning of Network Security Functions (NSFs). Unlike traditional IPsec implementations, which rely on decentralised negotiation mechanisms, CCIPS shifts security management to a centralised entity (i.e., the

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

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

SDN controller). This design reduces operational complexity, mitigates misconfigurations, and enhances the visibility and consistency of security enforcement across large-scale environments.

A key aspect of CCIPS is its adoption of an IKE-less approach for SA establishment. In this model, the central controller generates and provides the cryptographic material required for the tunnel setup, thereby eliminating the need for direct negotiation between endpoints. This not only simplifies the process of establishing secure communications but also reduces computational overhead on the destination host, making it particularly suitable for constrained or virtualised edge nodes.

To secure this centralised model and protect sensitive assets, the TNED integrates TEEs as a core architectural component. In fact, TEEs provide hardware-enforced isolation for executing critical operations, ensuring the protection of sensitive assets such as cryptographic keys, credentials, and configuration data. This provides hardware-enforced isolation for critical operations, ensuring their integrity even in the presence of a compromised host. Combined with Remote Attestation (RA) mechanisms, this Confidential Computing (CC) foundation ensures trust by enabling system validation and anchoring secure communications to a verifiable root of trust.

Building on the preliminary results [1], [2], this paper advances the state of the art with the following novel contributions:

- A complete formalisation of the TNED architecture within the CC paradigm.
- A full implementation of the CCIPS Agent across two heterogeneous, open-source TEE frameworks (Keystone on RISC-V and Enarx on x86), demonstrating cross-platform portability.
- An experimental validation comprising per-operation benchmarks, end-to-end tunnel establishment scalability in mesh topologies, throughput analysis, resource consumption profiling (CPU, memory, binary size), and RA overhead assessment.
- The presentation of a compromise handling workflow that specifies detection, key revocation, node isolation, and auditing procedures upon failed verification.

The remainder of this article is structured as follows. Section II provides the necessary background on IPsec, TEEs, and related technologies. Section III introduces the threat model considered for the proposed architecture. Section IV discusses related research efforts and highlights the novelty of our contribution. Section V describes the TNED and its purpose. Section VI details the architecture and design of the CCIPS framework and Section VII describes the related workflows. Section VIII outlines the practical implementation of the CCIPS Controller and Agent components, while Section IX validates the system through experimental analysis. Finally, Section X concludes the paper and outlines potential directions for future research.

II. BACKGROUND

This section introduces the core technologies that form the basis of our work. It begins with an overview of IPsec, which provides secure communication at the network layer, and then presents the concept of TEEs, focusing on the Keystone and Enarx frameworks.

A. IPsec

IP Security (IPsec) is a widely used security protocol that provides a suite of cryptographic services for protecting network traffic at the IP layer [3]. It allows the creation of VPNs over untrusted networks (*tunnel mode*) or end-to-end secure packet flows (*transport mode*).

At its core, IPsec relies on two key protocols to enforce security, namely Authentication Header (AH) and Encapsulating Security Payload (ESP). AH ensures data integrity, authentication, and optional protection against replay attacks. ESP extends the properties of AH by providing confidentiality.

A fundamental element of IPsec is the SA, a unidirectional logical connection between two IPsec hosts which defines the cryptographic parameters for each traffic flow. These SAs are managed through a dedicated database, the Security Association Database (SAD). The SAD stores active SAs along with their cryptographic parameters, such as encryption and authentication algorithms, keys, and lifetimes. Yet, IPsec relies on multiple databases, not just the SAD. The Security Policy Database (SPD) dictates how packets should be handled, determining whether they should be discarded, bypassed, or protected. Additionally, the Peer Authorization Database (PAD) associates SPD and SAD entries. It ensures proper authentication before an SA is established, mapping each peer to its identity, authentication method, and authorisation data.

To establish an IPsec connection, peers must agree on cryptographic parameters and authenticate each other. This process is typically managed by the IKE protocol, which automates key management and SA negotiation. IKE ensures that encryption keys are securely exchanged and refreshed periodically. The latest version, IKEv2, introduces improvements in efficiency, security, and robustness [4].

B. TRUSTED EXECUTION ENVIRONMENT

A Trusted Execution Environment (TEE) is a secure area within the processor which guarantees that sensitive data can be stored and processed in an isolated environment [5]. Its primary purpose is to protect against threats originating from the Rich Operating System (Rich OS), a versatile execution environment which runs various applications. In this scenario, a specific program requiring protection is defined as a Trusted Application (TA). A TA is an authorised software that performs sensitive operations, possibly on confidential data, executed inside the TEE [6]. According to GlobalPlatform, it must be isolated from both the Rich OS and other TAs [5].

This ensures that if one TA becomes corrupted, it cannot compromise the integrity and confidentiality of others.

Notably, TEEs are the concept on which CC relies upon. In fact, the objective of CC is to protect data in use by leveraging hardware-based TEEs [7], [8], [9].

C. KEYSTONE ENCLAVE

Keystone is an open-source framework developed to address the limitations of vendor-specific TEEs, which often impose rigid security models, hardware dependencies, limited resource management, and closed-source restrictions [10]. Designed to run on RISC-V architectures [11], its primary goal is to provide a flexible and secure execution environment, enabling developers to tailor the security model to their specific hardware and deployment needs using a modular set of components.

At the core of Keystone lies the concept of *Enclave*, a secure execution environment isolated from the untrusted operating system and other enclaves. Each enclave includes a Enclave Application (EAPP) and the Runtime (RT). The latter, running in Supervisor-mode, operates similarly to a lightweight kernel, handling tasks such as virtual memory management, system calls, and exception handling.

The highest-privileged component within Keystone is the Security Monitor (SM), an M-mode software layer designed to minimise the Trusted Computing Base (TCB) while performing essential security operations. Leveraging RISC-V architecture features, the SM is responsible for enclave lifecycle management and enforces memory isolation through the Physical Memory Protection (PMP) mechanism.

D. ENARX

Enarx is an open-source framework designed to run workloads within TEE instances across different platforms. Unlike vendor-specific TEEs that are tightly coupled to particular hardware architectures, Enarx adopts a CPU-agnostic approach, allowing applications to run on various TEE implementations, such as Intel SGX and AMD SEV [12].

Enarx introduces the concept of *Keep*, a TEE instance that provides a secure execution environment where applications remain isolated from the host system, including the hypervisor, kernel, and user-space processes. Each Keep leverages hardware-based memory encryption and attestation mechanisms to guarantee the integrity and confidentiality of workloads. The Enarx security model restricts the TCB to only three components: the CPU and its firmware, the application workload, and the open-source Enarx middleware. Enarx provides a runtime based on WebAssembly (Wasm). Wasm is a portable target for compilation which abstracts platform-specific details and allows applications to be written in various languages, including Rust, C, C++, Python, and more.

III. THREAT MODEL

Our scenario considers a heterogeneous network infrastructure in which different devices rely on IPsec for secure communication. These devices may be deployed in untrusted networks, where adversaries may attempt to compromise their configuration or capture cryptographic parameters. Since these network devices often run general-purpose operating systems and can be remotely managed, an attacker could target vulnerabilities in both the software stack and the communication interfaces. Unlike traditional IPsec solutions, which assume that the OS and network stack remain uncompromised, our model acknowledges that an adversary could manipulate the system at multiple levels.

We distinguish between the protection boundary offered by the TEE and the host system components that remain outside it. The TEE, including the TA and its protected memory, is considered trusted. The cryptographic material and security configuration stored within the TA are therefore assumed to remain confidential and to be protected, at rest and in transit, against software adversaries operating outside the TEE.

Thus, we consider the following adversaries:

- *Network Adversary* (N_{adv}): the attacker can exploit weaknesses in network protocols to intercept, modify, or inject malicious IPsec configurations. Furthermore, it can read and write data packets passing through the network interfaces. N_{adv} is capable of performing both passive attacks, such as traffic analysis or eavesdropping, and active attacks (downgrade attacks, replay previously captured SAs, injection, etc.).
- *Software Adversary* (S_{adv}): the attacker can compromise the Rich OS and software components running outside the TEE. However, S_{adv} cannot access the enclave's private memory region. Consequently, S_{adv} cannot extract the TA's private decryption key, forge attestation credentials, or tamper with the trusted reference databases stored inside the enclave. While S_{adv} may read plaintext keys from kernel SAD structures after they have been installed for packet processing, any modification to these entries triggers the TA's verification mechanism.

We also assume that a *Control Plane Adversary* (CP_{adv}), capable of compromising an SDN controller, is out of scope for *this specific work*. Indeed, we have already proposed a solution to secure the control plane [1]. Therefore, in this paper, we focus on adversaries targeting the *data plane*.

Despite the strong guarantees provided by TEEs and RA, certain vulnerabilities remain inherent to the architecture. One of the most relevant is the Time-of-Check to Time-of-Use (TOCTOU) attack. Runtime RA occurs periodically and, for this reason, a capable s_{adv} could theoretically exploit a vulnerability to extract or manipulate IPsec parameters in the window between two consecutive attestation rounds. Furthermore, as previously stated, once the TA provisions the plaintext keys into the untrusted host kernel's SAD and SPD for data plane processing, those keys exit the TEE boundary and become susceptible to kernel-level extraction.

Other complex attacks not covered by this architecture target the application control flow. RA solutions that aim to attest the control flow are referred to as Control Flow Attestation (CFA) [13]. CFA, however, usually suffers from

TABLE 1. Comparison with most related research.

Work	TEE	Open-source TEE	Target Network Function (NF)	Remote Attestation	Centralised Management
SENG [14]	Intel SGX	×	Gateway (Firewall)	✓	×
S-NFV [15]	Intel SGX	×	Snort NF	✓	×
Wang et al. [16]	Intel SGX	×	IDS	Partial	×
OFTinSGX [17]	Intel SGX	×	OpenFlow	✓	✓
EndBox [18]	Intel SGX	×	OpenVPN	✓	✓
TruGW [19]	ARM TrustZone	×	Gateway (Firewall)	✓	×
Al Atiiq et al. [20]	AMD SEV-SNP	open-source firmware	Snort NF	×	×
Jacquin et al. [21]	-	-	OpenFlow	✓	✓
López-Millán et al. [22]	-	-	IPsec	×	✓
TNED	Keystone, Enarx	✓	IPsec	✓	✓

path explosion, often requires introducing custom hardware extensions, or significantly increases the TCB size, which is a critical limitation, especially in constrained environments.

Finally, physical attacks and microarchitectural side-channels are out of scope.

IV. RELATED WORK

Most of the literature on the use of TEEs in SDN considers Intel SGX as the core technology. SENG [14] demonstrated the use of Intel SGX to build a network gateway, with the goal of allowing firewalls to attribute traffic to specific applications. Similarly, TruGW [19] leveraged ARM TrustZone to isolate routing and firewall logic from vulnerable services. Within the NFV domain, research has focused on protecting Virtual Network Functions (VNFs) from compromised host environments. For example, Shih et al. proposed S-NFV [15] to isolate sensitive VNF states using Intel SGX. In Trusted Click [23], the authors extended the Click modular router and integrated it with Intel SGX to perform secure packet processing. Wang et al. [16] further enhanced this by proposing a modular VNF architecture, based on multiple enclaves, to ensure scalability and support dynamic reconfiguration. Paladi et al. [24] and Medina et al. [17] used Intel SGX to protect OpenFlow tables. FeIDo [25], instead, uses Intel SGX to secure authentication tokens derived from electronic IDs. EndBox [18] tries to overcome the limitations of [26] and [27], providing a framework based on OpenVPN and Intel SGX that allows the execution of middlebox functions directly on client machines.

Recent studies have expanded beyond Intel SGX to evaluate other TEEs, such as AMD SEV. Atiiq and Risdianto [20] benchmarked the performance overhead of running unmodified NFs, such as Snort, within SEV-SNP. In TrustedGateway (TruGW) [19], the authors leveraged ARM TrustZone to isolate routing and firewall logic from vulnerable auxiliary services.

In other works, the authors used trusted computing concepts in an SDN scenario without using TEEs as trust anchors. Jacquin et al. [21], for instance, leveraged the Trusted Platform Module (TPM) and Linux Integrity Measurement Architecture (IMA) to include measurement of OpenFlow rules in the RA process.

Beyond hardware-based network security in SDN scenarios, recent literature has proposed holistic abstraction models and end-to-end TEE design paradigms to manage trust across complex and distributed ecosystems. Specifically, MEMCA [28] is an architecture that formalises trust as a coefficient embedded into a system's throughput model and enforces it through periodic checksum verification across a tiered cloud-to-edge hierarchy. Building upon this concept, Ağca et al. [29], [30] propose Trusted Distributed AI (TDAI), a methodology for distributed AI systems that distinguishes centralised, decentralised, and hybrid architectures, and relies on TEE-based trusted agents to manage the system's interaction with its environment across untrusted networks. TDAI and holistic abstraction models such as MEMCA target the algorithmic and system layers, and their deployment inherently relies on robust, scalable, and trusted network channels to connect distributed nodes. Our proposed TNED architecture aims to address this gap by providing the foundational trusted connectivity layer.

Driven by the need to overcome the I/O bottlenecks and host-level vulnerabilities of CPU-bound TEEs, recent literature has also focused on confidential networking using SmartNICs and Data Processing Units (DPUs) [31], [32], [33]. Several architectures propose shifting the security perimeter directly to the network interface. For instance, S-NIC [34] introduces hardware-level virtualisation of on-NIC resources to provide strong isolation among multi-tenant network functions while defending against side channels. Similarly, HELIOS [35] leverages SmartNICs to establish physically isolated communication channels for container environments, offloading L3-L7 security inspections to the hardware. In the context of verifiable trust, TNIC [36] builds a formally verified silicon root-of-trust at the network interface level, bypassing host CPU TEEs to support Byzantine fault-tolerant distributed systems. Other approaches explore DPU-enhanced architectures to offload deep packet inspection tasks, increasing network security situation awareness and throughput [37].

To reconcile the cryptographic guarantees of TEEs with the requirements of high-speed packet processing, recent works investigated user-space enclave network processing via kernel-bypass frameworks such as Data Plane Development Kit (DPDK) [38], [39], [40], [41], [42]. Crossing the isolation boundary of a hardware enclave via traditional system calls introduces significant latency. For this reason, early designs such as SafeBricks [39] partitioned the network stack on

top of the NetBricks [38] runtime. In SafeBricks, DPDK is executed in the untrusted host space and packets are exchanged with the Intel SGX enclave over shared-memory queues. In this way, it achieved transition-free I/O while minimising the TCB. Conversely, rkt-io [40] ported DPDK directly inside the enclave alongside a Linux Kernel library. With this approach, device Direct Memory Access (DMA) queues are mapped to untrusted host memory and are called directly from the TEE. As a result, rkt-io eliminated host-mediated system call indirection and mitigated Iago interface attacks. However, this required an explicit copy between the DMA buffer and the protected enclave memory.

In the domain of Confidential Virtual Machines (VMs) (e.g., AMD SEV-SNP), where hardware memory encryption prevents external devices from performing direct DMA into private guest memory, FOLIO [41] introduced a secure DPDK extension. It leverages pre-allocated shadow packet buffer pools to bypass dynamic bounce-buffer allocation overheads. It aims to achieve near-native throughput on commodity hardware without relying on emerging trusted I/O peripherals. Furthermore, comparative performance analyses [42] evaluated DPDK-based P4 pipelines across Intel SGX and AMD SEV-SNP, demonstrating the architectural trade-offs between extern models and whole virtualisation in secure VMs.

In the context of IPsec, other works on integrating IPsec in SDN networks are available. However, none of them considers any TEE solution in their design. In addition, they either target the IKE protocol or OpenFlow switches, and use non-standardised interfaces. For instance, SDIG [43] proposes an IPsec gateway that acts as a middleware that exchanges IKE packets and encrypts/decrypts IP packets. Similarly, in [44] the authors present a modular SDN-based solution that orchestrates and separates IKE and packet encryption.

Instead, in [45] the authors proposed a solution that has been standardised in RFC-9061, which defines interfaces to manage SAs by leveraging the SDN paradigm. Interestingly, in this work, two cases have been defined: the traditional IKE case and the IKE-less case, where the SDN Controller sends the IPsec configuration directly to the network node. Later, in 2023, López-Millán et al. [22] validated the RFC-9061. In the same year, different algorithms have been proposed to perform the rekey process in the IKE-less approach [46]. Afterwards, within the same framework, it has been proposed to generate IPsec keys using the Diffie–Hellman algorithm [47].

Building on [22] and [45], we adopt the same specifications to implement an IKE-less model in our framework, while leveraging TEEs to protect sensitive operations and data.

Unlike previous work, the TNED provides implementations for multiple TEEs, specifically Keystone Enclave and Enarx, that are not explored in this research area. In particular, the proposed TNED approach extends prior research that explores the integration of TEEs and attestation mechanisms within network security architectures [1], [2].

In [1], we investigated the IKE-less approach for IPsec management, proposing a centralised SA deployment through a trusted control plane. This approach eliminates the need for traditional IKE exchanges, simplifying policy management in the data plane while maintaining security guarantees. Additionally, our previous demonstration [2] showcased the feasibility of incorporating attestation mechanisms into a centralised IPsec management model. This proof of concept illustrated how TEEs could be employed to enhance the security of IPsec configurations by providing isolation and integrity verification. However, as a short demo paper, it focused primarily on architectural design. This paper moves beyond the previous limitations by providing a comprehensive formalisation of the CCIPS architecture using the CC paradigm, a complete implementation of the framework across heterogeneous TEEs, and a performance and security validation.

Table 1 summarises the differences between the TNED and existing solutions found in the literature. The novelty of TNED lies in combining SDN concepts with multiple hardware-based, open source trust anchors. Furthermore, Keystone Enclave and Enarx have been rarely explored in this research area.

While most works in the literature focus on firewalls, OpenFlow configuration, or other NFs, the TNED specifically addresses the security challenges of IPsec.

V. TNED CONCEPT AND ARCHITECTURE

The Trusted Network Environment for Devices (TNED) is a framework designed to enhance security and trust in networked environments. Its primary objective is to establish a secure execution environment for managing critical operations, protecting sensitive data, and ensuring the integrity of communications. The TNED is based on the SDN [48] and NFV [49] paradigms and operates as an intermediary between network infrastructure and end devices. It provides security functionalities such as secure communication channels, RA, and policy enforcement. To achieve this, the TNED leverages a combination of TEEs and cryptographic mechanisms.

A. CONTROL PLANE

A crucial element in managing TNED operations is the Operation, Administration, and Management (OAM) concept. OAM enables network operators to perform resource provisioning, network performance monitoring, and issue troubleshooting. In addition, this OAM system extends its capabilities to include the provisioning and orchestration of both trusted and untrusted applications.

The OAM system comprises the following components:

- *CCIPS Controller*: described in Section VI-B.
- *Trust Monitor (TM)*: the TM [50] is responsible for continuously verifying the integrity of TNED software and hardware components. It features the TM Core, a central entity that coordinates the execution of RA processes. Designed with a modular architecture, the TM supports multiple attestation technologies through

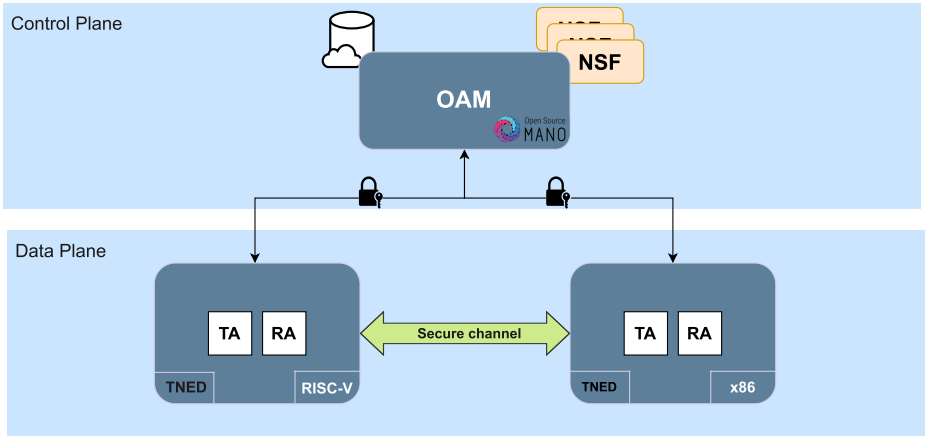

FIGURE 1. TNED architecture.

Adapters, which act as intermediaries between the TM Core and specific attestation mechanisms. These Adapters facilitate interoperability by translating attestation results into a format compatible with the TM Core.
- *Trust Controller (TC)*: manages the deployment, monitoring, and life-cycle of trusted applications on TNED platforms. It also supports attestation procedures and ensures compliance with security policies.

This OAM solution is built on Network Functions Virtualization Management and Orchestration (NFV MANO) and cloud-native technologies [51], which allows for the control and monitoring of the behaviour of each NSF deployed within the TNED architecture. In this scenario, a NSF provides modular security capabilities for specific needs, such as encryption and firewall functionalities.

B. DATA PLANE

Each NSF can be deployed as a virtualised service or directly attached to the TNED design. Notably, a NSF consists of 2 components:

- *Untrusted Application (UA)*, in charge of processing the non-critical operations.
- *Trusted Application (TA)*, executed in the TEE and thus manages sensitive operations and data (e.g., keys, certificates, security configuration, etc.).

In addition, each NSF is equipped with an Orchestration Agent (OA). Running as a daemon or service on each node, the OA manages both TAs and UAs to ensure controlled execution and secure interactions. It facilitates communication between the control plane (Fig. 2) and the node hosting the TEE. This is achieved by exposing the TEE platform's functionalities via an open API integrated with the common Virtual Infrastructure Manager (VIM) abstraction. This abstraction provides two distinct interfaces, one connecting to Open Source MANO (OSM) [52] and another to the OA.

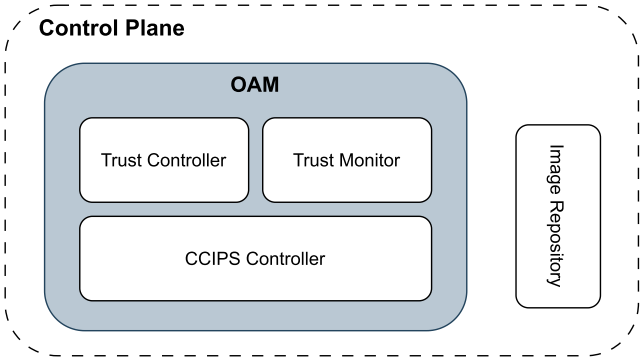

FIGURE 2. TNED control plane.

VI. CENTRALLY CONTROLLED IPsec

The concept of Centrally Controlled IPsec (CCIPS) stems from the growing need for secure connectivity within 5G networks. It acts as a NSF deployed in (at least) two nodes to ensure secure communication. As a core component of the TNED, the CCIPS enables centralised management of SAs to enhance both security and scalability in dynamic environments. Specifically, it leverages the Interface to Network Security Functions (I2NSF) to establish standardised interfaces for deploying NSFs [53].

A. IKE-LESS APPROACH

Traditional IPsec deployments rely on the IKE protocol for SA establishment. Although IKE offers robust authentication and key exchange, its overhead and complexity can be prohibitive, especially in large-scale networks with resource-constrained devices. To overcome these limitations, the I2NSF framework provides a dedicated standard for IPsec [45], [54]. This standard defines two YANG models: the IKE model (Fig. 3), which uses IKE to negotiate key material, and the IKE-less model (Fig. 4). Traditional IKE-based configurations require each NSF to independently manage

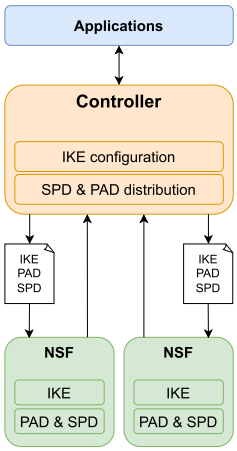

FIGURE 3. I2NSF IKE architecture.

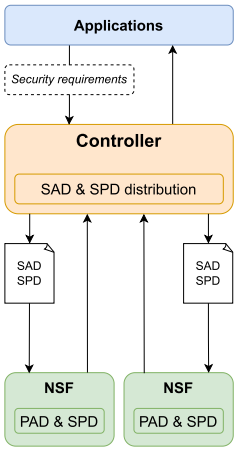

FIGURE 4. I2NSF IKE-less architecture.

the SA agreement process, resulting in increased data-plane overhead. In contrast, the IKE-less model eliminates the need for peer-to-peer key negotiations by enabling a centralised controller to provision the required SAs. Two different modes are defined for the IKE-less case: *proactive* and *reactive*. In the proactive mode, both IPsec policies and SAs are provided together. With the reactive mode, instead, the Controller provides at first the SPD entries, then the node has to ask for the respective SAs.

However, this approach also introduces certain limitations. The SDN controller may become a single point of failure, as its compromise or unavailability can disrupt the secure connectivity. Furthermore, centralisation can raise concerns related to scalability and controller workload. Nevertheless, these drawbacks are considered acceptable in our scenario. In fact, this approach allows for fine-grained configuration of SPD and SAD entries without invoking the IKE protocol, thereby mitigating the resource constraints typically encountered in lower-end devices.

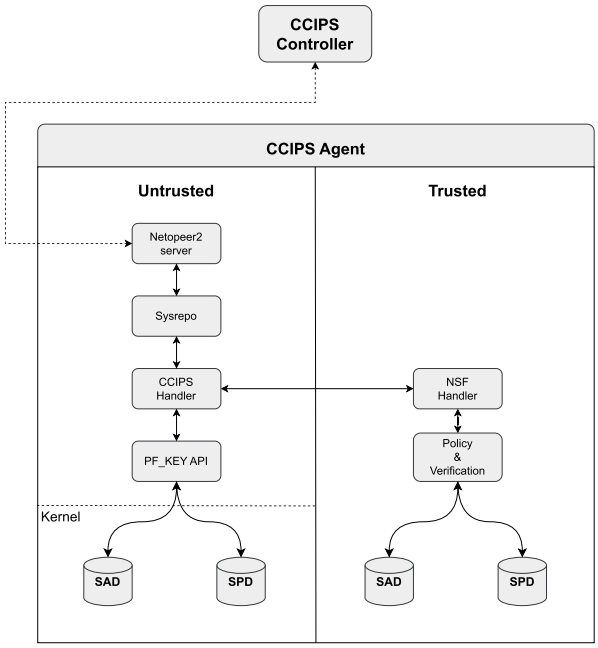

FIGURE 5. CCIPS agent architecture.

B. CCIPS CONTROLLER

The CCIPS Controller serves as the central management point for the IPsec tunnel configuration within the CCIPS framework. It receives security and configuration requests from the OAM system and translates these requirements into specific configurations in accordance with the IKE-less data model.

To achieve this, the Controller exposes a Northbound Interface (NBI) via REST APIs, enabling authorised applications to initiate, monitor, and manage the lifecycle of IPsec tunnels. On the Southbound Interface (SBI), the Controller communicates securely with CCIPS Agents through the Network Configuration Protocol (NETCONF) [55], a network management protocol designed for configuring network devices. NETCONF leverages the YANG modelling language [56] to structure and exchange configuration data between clients and server.

Each NETCONF session is required to ensure data integrity, confidentiality, authentication, and replay protection. For this reason, NETCONF mandates the use of a secure transport protocol. Hence, either SSH or TLS must be employed. In this implementation, an SSH-based secure channel has been chosen. This not only ensures the reliable delivery of configuration parameters but also allows agents to forward IKE-less notifications used to manage the lifecycle of SPD and SAD entries.

C. CCIPS AGENT

The CCIPS Agent (Fig. 5) is a NSF which is in charge of establishing one side of the IPsec channel. Running on a

network device in the data plane, it receives the configuration parameters from the CCIPS Controller and deploys the requested IPsec channel. As mentioned before, it consists of two applications: the UA and the TA.

1) Untrusted Application (UA)

The UA runs in the Rich OS. It is primarily responsible for receiving the IPsec configuration from the CCIPS Controller via NETCONF and for sending the notifications defined in the IKE-less YANG model.

The main component of the UA is the CCIPS Handler. This process listens for changes in the IPsec configuration. Whenever this state is updated, the information needs to be processed so that it can be handled by the kernel and the TA. A communication channel is thus established with the TA to transmit the encrypted configuration. In this way, it can be securely decrypted for the installation. It can also request actions such as checking whether the kernel's current IPsec configuration is valid. Finally, the CCIPS Handler can install and remove IPsec policies and SAs by directly interacting with the Kernel. It also handles the events generated by the kernel so that it can directly notify the CCIPS Controller.

2) Trusted Application (TA)

The TA runs in an enclave inside the TEE. It mainly manages and validates SAD and SPD entries.

The first main component, the NSF Handler, parses JSON requests from the UA that include an identifier for the task to be performed. Those requests are then forwarded to the Policy and Verification component.

The Policy and Verification module manages the different requests received from the NSF Handler. Specifically, it is in charge of performing the following tasks:

- Adding new SAD and SPD entries in the trusted databases. During this process, it must decrypt the cryptographic material sent by the CCIPS Controller.
- Delete the SAD and SPD entries once they are no longer used.
- Verification of the policies and SAs that have been stored in the trusted databases against the ones deployed into the kernel. Importantly, this operation is performed both periodically and on configuration changes.

It is important to note that this procedure ensures that the trusted SPD and SAD databases are updated before their counterparts in the kernel.

VII. WORKFLOWS

This section describes the main operational workflows of the CCIPS framework, focusing on how secure IPsec tunnels are configured, maintained, and validated. It includes the sequence of actions involved in the setup of SAs, the secure distribution of cryptographic material, and the coordination among Controller, Agents, and trusted components. Additionally, it outlines the procedures for tunnel re-keying, trusted boot, and runtime attestation, which collectively ensure and secure communication across the system.

A. IPSEC TUNNEL CONFIGURATION

The tunnel configuration process, illustrated in Fig. 6, begins when the CCIPS Controller receives a request from the TC. This request includes various parameters, such as the management addresses of the CCIPS Agents, the networks that need secure communication (depending on the operational mode, such as gateway-to-gateway scenarios), the IP addresses of both endpoints, and specific security requirements (including cryptographic algorithms and the threshold for the rekeying process).

1) KEY GENERATION

Upon receiving this request, the CCIPS Controller parses the provided information and generates the required configuration material in accordance with the I2NSF IKE-less model. As part of this process, the controller also produces the IPsec cryptographic key material, ensuring compliance with the requested security algorithms. The CCIPS Controller encrypts the key material using platform-specific public keys received from the agents after the first attestation phase. This mechanism ensures that IPsec parameters are accessible only to the intended TA and protected from a potentially compromised host. The generation of these cryptographic keys is tailored to the specific TEE framework. In Enarx, the public key is generated within the Keep and is certified by the Steward (see Fig. 9) via an X.509 certificate. The Steward acts as a trusted certification authority that issues the certificate only after successfully authenticating the underlying hardware's TEE capabilities and verifying runtime integrity. For the Keystone agent, the key generation process follows the Device Identifier Composition Engine (DICE) architecture [57], [58]. Specifically, the enclave generates a temporary key pair, which is then certified by an identity key derived from the unique measurement of the enclave and the SM.

2) CONFIGURATION OF THE AGENTS

The configuration is structured into an XML descriptor conforming to the NETCONF standard. This descriptor adheres to the expected structure defined by the YANG model. The CCIPS Controller then establishes authentication with the CCIPS Agents over a secure SSH channel before transmitting the configuration descriptors.

Once authenticated, the CCIPS Agents use Sysrepo to parse the received configuration, ensuring its compliance with the I2NSF IKE-less model. After successful validation, applications that have subscribed to changes within the relevant XPath locations, such as those for SAD and SPD entries, receive notifications about the updated configuration in the datastore.

In this case, the CCIPS Handler is responsible for processing the notifications received from Sysrepo. It translates them into its internal data structures, which are then handled by the TA and the PF_KEY Management API [59]. Before applying the security policies to the kernel, the SAD and SPD

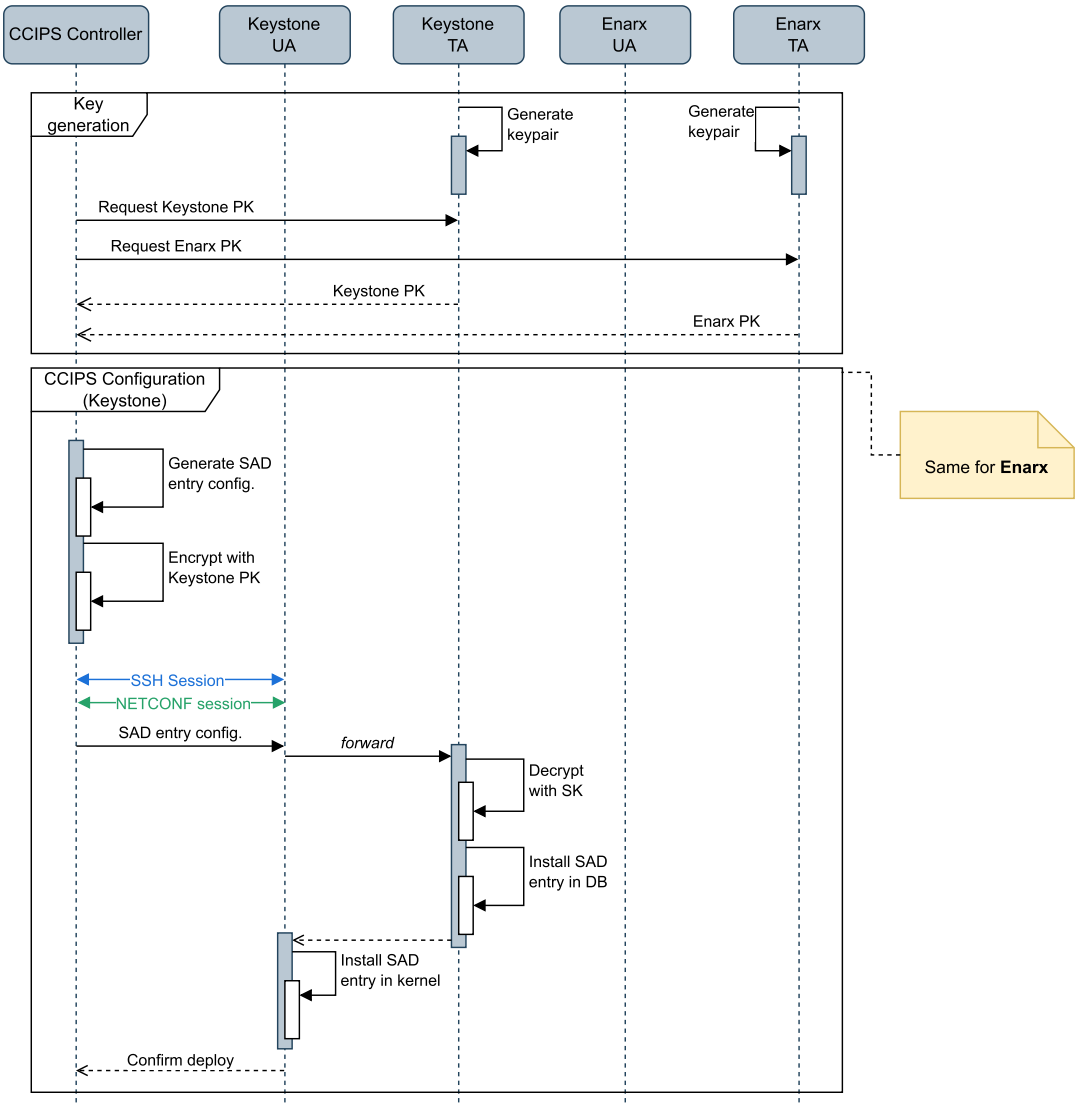

FIGURE 6. SAD configuration process.

entries are forwarded to the NSF Handler within the TA. Here, the cryptographic keys and other sensitive information are decrypted and securely stored in the TA's database. The NSF Handler then forwards the decrypted SAD and SPD entries to the policy verification process, ensuring their integrity before returning them to the CCIPS Handler. With the validated entries in place, the CCIPS Handler applies the IPsec rules to the kernel via the PF_KEY API, thereby establishing one end of the secure tunnel.

Finally, after successfully installing SAD and SPD entries on both CCIPS Agents, the CCIPS Controller sends a status update to the TC, confirming the completion of the rekeying process.

B. RE-KEYING PROCESS

In the context of I2NSF IKE-less, the RFC-9061 [54] specifies a re-keying procedure managed by the Controller upon request from the nodes. This process is essential, as the Agents require new SAD entries before their expiration to ensure uninterrupted IPsec tunnel operation.

Fig. 7 illustrates the key steps involved in this procedure. The CCIPS Controller establishes a NETCONF session with the Netopeer2 [60] service and subscribes to Sysrepo's datastore change notifications. At this point, the NSF can securely send a notification to the Controller. In particular, this allows it to receive `SADB_EXPIRE` notifications (as defined in the YANG data model) whenever a SAD entry approaches expiration.

When a soft expiration condition for a SAD entry is met (e.g., exceeding its lifetime, byte count, or packet threshold), the kernel triggers a notification via the PF_KEY Management API. The CCIPS Handler, monitoring these events, intercepts the notification, extracts the relevant SAD entry details, and generates a corresponding NETCONF notification via Sysrepo, which is then transmitted to the Controller. Upon receiving the notification, the Controller

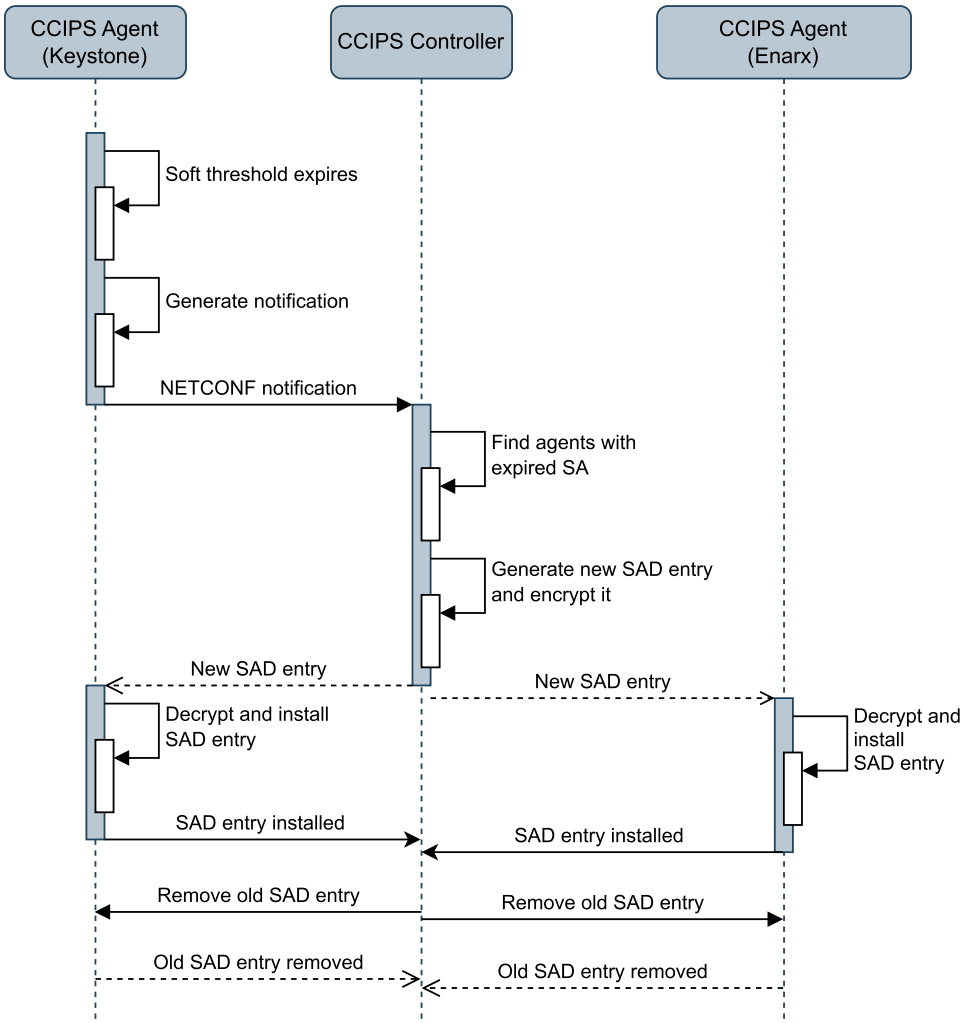

FIGURE 7. Re-keying process.

identifies the corresponding Agents responsible for the expiring SAD entry. Subsequently, the CCIPS Handler decrypts the cryptographic material and installs the new entries, first within the TEE and then in the kernel.

Once the updated security policies are successfully applied, the Controller initiates the removal of obsolete SAD entries from both Agents. This is accomplished by issuing a delete operation via Sysrepo, ensuring that the outdated entries are removed from both the kernel and the TA, thereby maintaining consistency and security within the system.

C. COMPROMISE HANDLING AND EXIT STRATEGY

The TNED implements a response mechanism, orchestrated by the control plane, to handle both the failed verification of the IPsec configuration and failed attestation. The remediation workflow, illustrated in Fig. 8, transitions through the following states:

1) *Detection:* Trust verification happens at two points in the CCIPS lifecycle. Before the CCIPS TA is launched, a one-time CCIPS host attestation must succeed. Once the TA is launched and the agent is executing, the CCIPS Agent continuously performs runtime attestation and SAD/SPD verification, reporting a mismatch in a SAD or SPD entry to the TC; concurrently, the SM monitors the host and TA, alerting the TM if it is found compromised, which in turn flags the node and reports the failure to the TC.
2) *Response:* The corrective action depends on when the failure occurs. If the initial CCIPS host attestation fails, the TC simply withdraws the CCIPS TA deployment from the node. If a failure is instead detected while the CCIPS Agent is already executing (i.e., a runtime attestation failure or a SAD/SPD mismatch), the TC places the node into a *quarantine* state: it disables the CCIPS Agent's operations and enforces network access control to isolate the node from other networks, and then takes a trigger-specific action. If the trigger was a SAD/SPD mismatch, it instructs the CCIPS Controller to revoke the compromised IPsec keys, which sends a delete operation via NETCONF to flush the tainted

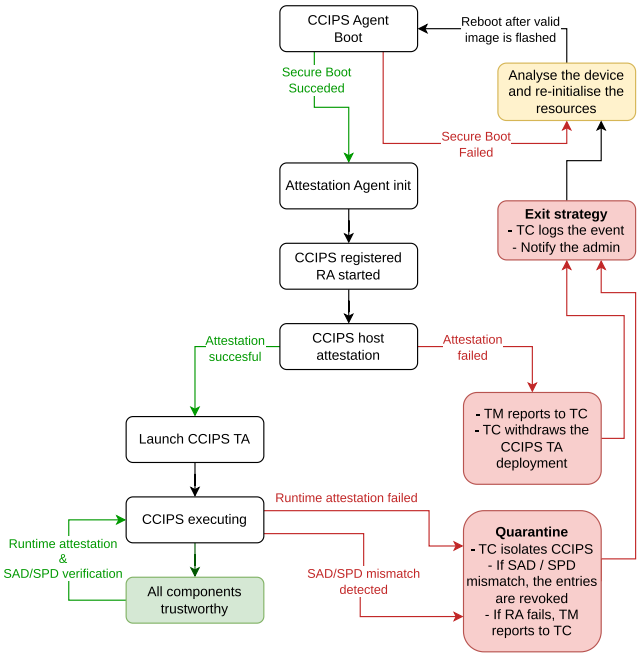

FIGURE 8. TNED verification lifecycle.

SAD entries across all peered nodes; if the trigger was a runtime attestation failure, the TM reports the failure to the TC as part of this same quarantine action.

3) *Auditing:* In all runtime-failure cases, the TC logs the event by recording the exact policy mismatch and/or the failed attestation status, and notifies the administrator as part of the *exit strategy*. A failure detected during Secure Boot instead bypasses this step entirely, since the TC is not yet involved at boot time, and proceeds directly to recovery.
4) *Recovery:* Whether the node reached this state via a failed Secure Boot or via the exit strategy above, it is analysed and its resources are re-initialised. The CCIPS Agent remains isolated until an administrator applies this manual recovery strategy to re-establish the device's security posture, after which the node reboots once a valid image has been flashed.

D. ATTESTATION

RA is a cornerstone of the TNED security model, enabling continuous verification of the integrity and authenticity of hardware and software components. This applies to both the Rich OS and the TEE. Trust is established at boot time through Secure and Measured Boot mechanisms, which ensure that each system component is loaded correctly and has not been tampered with. During operation, runtime attestation is used to monitor that components remain in a reliable state. As shown in Fig. 9, we focus on two TEE technologies: Keystone, which provides an open and customizable trusted execution environment for RISC-V platforms, and Enarx, which enables confidential workload execution on Intel SGX-enabled systems. In both cases, Linux IMA is used to attest the Rich OS at runtime.

1) ATTACKS MITIGATED BY REMOTE ATTESTATION

The integration of RA mitigates different software-level threats in the CCIPS workflow. In our scenario, RA actively prevents:

- *Malicious Agent Deployment:* It ensures that IPsec cryptographic keys are never provisioned to an unauthorised or modified CCIPS Agent binary. In addition, it ensures that the Agents are deployed only on hosts with configurations consistent with the reference values.
- *Man-in-the-Middle (MitM)* Thanks to the DICE specifications and RA, the CCIPS Controller binds the key distribution exclusively to the attested public key of the enclave, ensuring that configuration payloads cannot be successfully intercepted by a compromised host operating system.
- *Runtime Tampering:* Periodic RA detects if the host memory footprint is corrupted or manipulated during execution. For instance, it reports any unauthorised access to files. In the Keystone TEE, runtime RA has also been extended to measure the TAs.

2) KEYSTONE

We extended the TM architecture to support Keystone-based RA [61]. The Keystone framework was enhanced to enable attestation of TAs at both load-time and run-time, in accordance with the DICE specifications [57]. To integrate Keystone within the TM workflow, we introduced a dedicated Keystone Adapter that interfaces with the Keystone Verifier and thus with the TM Core.

In brief, the attestation procedure begins with retrieving and validating the certificate chain associated with the Local Attestation Key (LAK) of the target enclave. If not available on the Verifier, this chain is obtained from the SM through the Attestation Agent using Keystone-specific attestation commands. At this point, the system proceeds with periodic run-time attestation, during which the Verifier issues attestation requests to the SM. The latter computes a cryptographic hash over the enclave's protected memory regions and signs the resulting attestation report using the enclave's LAK. The Verifier then validates the signature and compares the reported measurement with trusted reference values to detect any tampering or unauthorised modification during execution. Finally, the attestation results are published.

3) ENARX

As in the Keystone scenario, the RA logic for the Enarx framework [62] has also been included.

The process begins when the TC retrieves the workload image from the Software Image Repository (Fig. 2) and deploys it to the target node. This workload contains both the TA and the UA. Once the deployment is complete, the TC registers the workload in the TM through the Enarx Adapter, thereby enabling its attestation.

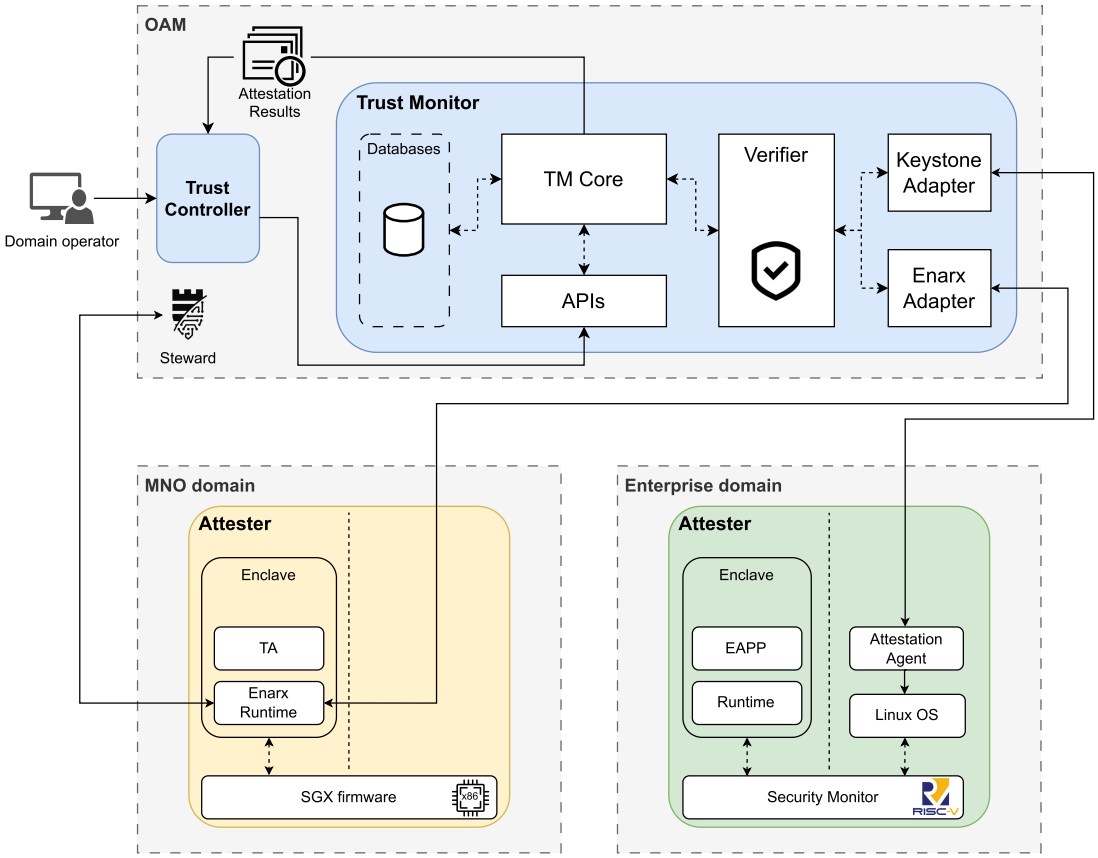

FIGURE 9. Attestation of trusted applications in the TNED.

The attestation phase starts with verification of the underlying trusted platform. In particular, the Steward first authenticates the Intel SGX capabilities of the host and verifies the integrity of the Enarx runtime. If these checks succeed, it issues a certificate that attests to the trustworthiness of the Enarx keep. Following this, the Enarx runtime initiates the attestation of the TA by sending its measurement together with the Steward's certificate to the Enarx Adapter. The Enarx Adapter validates the certificate to confirm the trust of both the hardware and runtime layers before verifying the TA measurement executed within the Enarx keep. Finally, the TM publishes the attestation results, making the trust status of the Enarx node available to other system components.

VIII. IMPLEMENTATION

A. CCIPS CONTROLLER IMPLEMENTATION

The CCIPS architecture is designed around a REST API, serving as the central component for its NBI while managing secure communication sessions with Agents through the NETCONF protocol. The CCIPS Controller has been developed using Go in order to implement these functionalities. In particular, it uses the Gorilla Mux library (v1.8.0)[1] for HTTP request handling and the Go NETCONF client library[2] to manage the interactions with the CCIPS Agents.

In addition, to improve scalability and enhance interoperability within SDN environments, the CCIPS Controller integrates its core functionalities in the Open Network Operating System (ONOS) SDN controller.[3]

B. CCIPS AGENT IMPLEMENTATION

The CCIPS Agent adopts a modular architecture, with the build process handled by *CMake*, which compiles both TA and UA.

The UA implements a NETCONF server to communicate with the CCIPS Controller. This server relies on *libnetconf2*[4] for managing NETCONF sessions, *libyang*[5] for data modeling, and *Sysrepo*[6] that serves as configuration datastore.

The TA is designed for portability and has been implemented for the two distinct TEE frameworks:

- *Enarx*: The TA is developed in Rust and compiled to a WebAssembly binary. This module is executed within

[1] https://github.com/gorilla/mux

[2] https://github.com/openshift-telco/go-netconf-client
[3] https://opennetworking.org/onos
[4] https://github.com/CESNET/libnetconf2
[5] https://github.com/CESNET/libyang
[6] https://www.sysrepo.org/

an Enarx Keep, which is deployed and managed via a Docker container.
- *Keystone*: A separate build target compiles the TA for the Keystone framework. The process generates a Keystone package containing the application and its runtime, which is then deployed within an enclave running on a RISC-V environment.

The core IPsec management logic is implemented in C language to maximise portability and maintainability, and is shared across both the implementations.

IX. VALIDATION

This section presents the validation of the proposed framework, focusing on the performance overhead introduced by the main functionalities of the CCIPS Agent (Section VI-C). While RA is a core concept of this architecture, evaluations of these procedures and their associated latencies have already been reported in [50], [61], and [62]. All validation tests for the framework were conducted on a machine equipped with an Intel Core Ultra 7 155H CPU and 32 GB of RAM. To minimise the impact of network latency, our experiments were conducted on a local network.

A. EXPERIMENTAL SETUP

As mentioned before, the performance evaluation was conducted using two different testbeds:
- *Keystone*: The CCIPS Agent was deployed on a RISC-V virtual machine emulated using QEMU, configured with 2 GB of memory, 1 virtual CPU, and a Linux kernel version 5.19.0. The virtual machine was equipped with the Keystone Enclave TEE. As this environment relies on software emulation rather than physical hardware, all Keystone performance metrics presented in this evaluation should be considered synthetic estimates.
- *Enarx*: The CCIPS Agent was also tested on the Enarx environment to evaluate the performance across different TEE implementations and validate the framework's portability. The deployment followed a containerised approach with two Docker containers. The first container executed the Enarx environment (version 0.6.3) with the TA of the CCIPS Agent. Meanwhile, the corresponding UA was running on the other container.

In both cases, the CCIPS Controller was running on another Ubuntu machine in the same local network, with Linux kernel version 7.1.10, an Intel Core Ultra 7 CPU, and 32 GB of RAM.

B. PER-OPERATION PERFORMANCE EVALUATION

The evaluation measured the execution time of six core CCIPS Agent operations: adding, verifying, and deleting trusted SAD and SPD entries. In our scenario, these operations represent the fundamental building blocks of the IPsec management system. For each operation, 20 measurements were collected from 20 independent runs to ensure statistical reliability. All reported values correspond to the average and standard deviation over the considered runs. Fig. 10 and 12 summarise the per-operation latencies for the Keystone and Enarx environments, respectively. The error bars in each column represent the standard deviation.

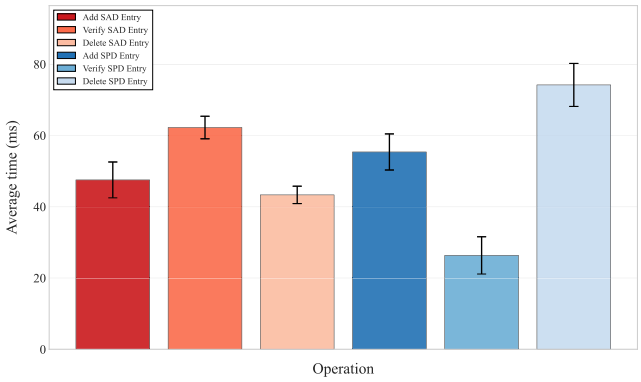

FIGURE 10. Operations performed in Keystone.

1) KEYSTONE

On the Keystone platform, the average execution time of operations ranged from 26.3 ms to 74.2 ms (Fig. 10). The fastest operation was the verification of SPD entries, with an average time of 26.3 ms, while deleting an SPD entry incurred the highest latency (74.2 ms). Operations involving SAs showed consistent performance, with average execution times between 43.3 ms and 62.2 ms.

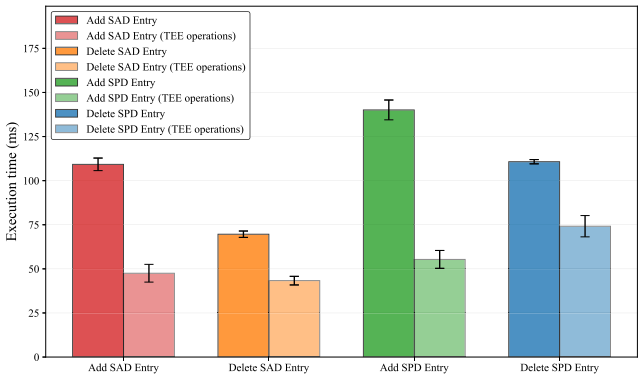

FIGURE 11. Overall and trusted operations in Keystone.

In addition, we measured the time spent in trusted execution (i.e., inside the enclave) compared to the total duration of each operation. This breakdown is illustrated in Fig. 11. The *TEE operations* columns correspond to enclave-side tasks, such as saving the SPD/SAD entries in the trusted databases, verifying their integrity, and performing validation. The *overall* columns cover both the operations executed in the TA and UA (installation/removal of entries in the Linux kernel and other IPsec-related actions).

As shown, the execution time inside the Keystone TEE accounts for approximately half of the overall latency, though the exact proportion varies across different operations.

2) ENARX

As depicted in Fig. 12, the Enarx platform demonstrated stable performance, with average execution times across all evaluated operations ranging from approximately 20 ms to 40 ms. Specifically, the fastest operation was the verification of an SAD entry (22.71 ms), while the most time-consuming task was the addition of an SPD entry (40.04 ms). Other operations, such as deleting SAD entries (24.22 ms) and deleting SPD entries (30.10 ms), also completed within this latency range.

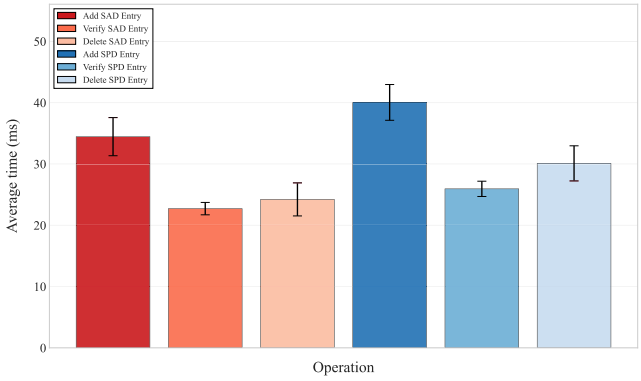

FIGURE 12. Operations performed in Enarx.

C. SCALABILITY AND END-TO-END LATENCY

To evaluate the scalability of the framework, we measured the end-to-end tunnel establishment time using the IKE-less proactive mode (i.e., providing SAs and IPsec policies simultaneously). Specifically, we benchmarked the time required to provision multiple tunnels concurrently, defined as the elapsed time from when an Agent sends the request until the IPsec configuration is installed in the Kernel. Fig. 13 illustrates the mean end-to-end tunnel establishment latency. The measurement captures the complete lifecycle: from the request being sent to the Controller, through provisioning with NETCONF, to enforcement in the Agent's kernel.

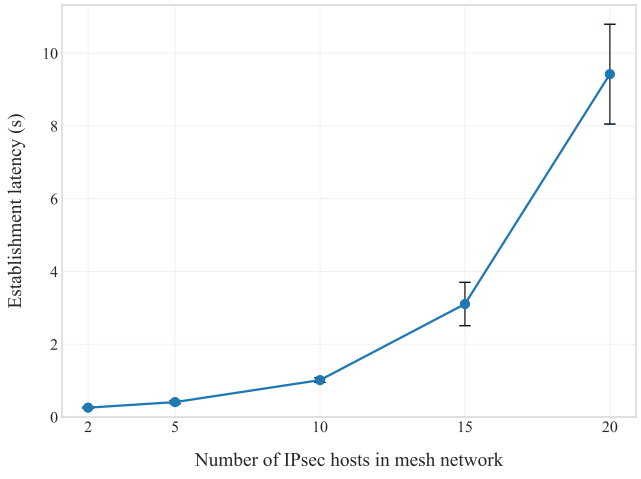

FIGURE 13. End-to-end establishment latency based on the number of concurrent tunnels.

The graph shows the mean time required to fully establish the tunnel as a function of the number of hosts. The Y-axis represents the mean tunnel establishment time, whereas the X-axis represents the number of hosts. The considered network configuration is a *mesh topology*, in which every pair of hosts is connected by an IPsec tunnel. Therefore, a network comprising n hosts requires $\frac{n\cdot(n-1)}{2}$ bidirectional tunnels, corresponding to $n\cdot(n-1)$ unidirectional SAs. Hence, the number of tunnels and SAs grows quadratically with the number of hosts, resulting in $O(n^2)$ complexity. For example, for 20 network nodes, 380 SAs are required.

Establishing a single tunnel takes approximately 260 ms to become fully operational end-to-end. Configuring 5 IPsec hosts requires, on average, a similar time (400 ms), while 10 and 20 hosts need 1.0 and 3.2 s, respectively. As the number of hosts scales to 20, the mean latency grows to roughly 9.4 s. As a result, the largest overhead is due to the computations performed in the SDN Controller. This can be motivated by the fact that the Controller must manage dozens of concurrent SSH sessions to open NETCONF channels with the Agents, and the number of needed SAs grows quadratically.

Finally, we must consider the local network environment used for this evaluation. Since the tests were conducted on a local network, these measurements isolate the computational overhead of the CCIPS architecture from additional network delays associated with Wide Area Networks (WANs). In a real-world deployment, the network would introduce additional delays in the Round-Trip Time (RTT), possibly increasing the total end-to-end establishment time.

1) COMPARISON WITH OTHER SDN-BASED IKE-LESS IPsec FRAMEWORKS

As reported in Table 2, the results obtained in this evaluation are consistent with those of López-Millán et al. [22]. Our scalability evaluation (Section IX-C) measured also the time to provide and install the IPsec configuration on the nodes. Instead, in [45] and [47], the authors measured only the Controller's performance. Since the time RTT and the time to install the configuration are negligible compared to the total elapsed time, we assume that both metrics are comparable. Even though their hardware specifications differ from ours and their testbed was deployed on Kubernetes, the latency for configuring $n = 20$ nodes in a mesh network and using the IKE-less proactive mode was similar (∼9 s). In [47], Parra-Espín et al. used the same testbed as [22] and obtained a slightly lower latency (∼6.5 s) for the same

TABLE 2. Comparison of end-to-end tunnel establishment latency with other SDN-based IKE-less IPsec frameworks ($n = 20$ nodes, mesh topology).

Work	Testbed	Latency	Complexity
TNED (this work)	QEMU/native	∼9.4 s	$O(n^2)$
López-Millán et al. [22]	Kubernetes	∼9 s	$O(n^2)$
Parra-Espín et al. [47]	Kubernetes	∼6.5 s	$O(n^2)$

number of nodes. However, in all the mentioned works, the same $O(n^2)$ time complexity was observed.

D. THROUGHPUT

The charts in Fig. 14 show the results of the tests performed to evaluate the packet processing performance of the established IPsec tunnels. For this evaluation, Keystone has been used as a test environment due to its limited resources. In each test case, we measured the data plane throughput using *iperf*, a well-known network benchmarking tool. The throughput, expressed in Mbits/sec, has been tested for both User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) traffic across varying packet sizes, ranging from 64 to 4096 B.

For each packet size, the evaluation compares the AES-256-CBC encryption algorithm against the legacy (and slower) 3DES algorithm, using the unencrypted channel as the baseline. In both scenarios, SHA-1 was used as the integrity algorithm. It is evident that the overall throughput scales with increasing packet size. This behaviour occurs since the IPsec encapsulation applies a fixed per-packet overhead; intuitively, larger packets reduce the ratio of this overhead.

Looking at the algorithms, AES-256-CBC clearly outperforms 3DES across all configurations. The performance can be further enhanced with hardware acceleration.

The results also reveal a distinction between the transport protocols. As expected, UDP achieves higher throughput than TCP across all configurations. However, this happens only after the packet size is bigger than 1 kB: in fact, for smaller packets, the TCP protocol performs slightly better.

E. CCIPS AGENT RESOURCE CONSUMPTION

Since edge and IoT devices are inherently resource-constrained, deploying a multi-component application requires a careful evaluation of its footprint. To address this, we measured the binary storage size, memory footprint, and CPU overhead imposed by the CCIPS Agent. Even in this case, Keystone has been used as a test environment.

The total binary storage size required for the compiled CCIPS Agent and its core dependencies is 687 kB on disk, which remains well within the storage limits of typical edge devices. Fig. 15, instead, tracks the CPU and memory consumption of the CCIPS Agent over a 50 s window, capturing both idle states, verification, and rekey events. In this scenario, the SA have been set up to have a soft lifetime of 30 s and a hard lifetime of 50 s.

The memory usage is quite limited and does not exceed 7 MB.

The periodic CPU spikes (reaching up to 80-90%) correspond to the operations such as SA re-keying and database verification triggered by the TA. The idle CPU overhead, by contrast, is very low, averaging 2%.

F. CCIPS CONTROLLER RESOURCE CONSUMPTION

In this evaluation, we measured the maximum CPU and memory consumption of the CCIPS controller under increasing workloads. Specifically, we scaled the number of concurrent IPsec tunnels from 1 to 50 and monitored the Controller's resource demands during the negotiation and establishment phases. Figure 16 illustrates the results of this benchmark. The left Y-axis reports the maximum CPU usage (where 100% represents a single fully used core), while the right Y-axis reports the maximum memory usage for that specific test case. As the graph shows, resource consumption scales linearly with the number of concurrent tunnels. For a single tunnel, the overhead is quite low, peaking at only 10.00% CPU and 16.09 MB of RAM. As the system scales to 10 concurrent tunnels, the maximum CPU usage rises to 39.90%, while memory utilisation remains at 68.74 MB. When 50 simultaneous tunnels are required, CPU usage peaks at 169.50% (more than a single virtual core), and memory usage reaches 308.62 MB. Overall, the benchmark demonstrates that the memory footprint scales predictably and remains within the hardware constraints of modern cloud nodes, while CPU scaling confirms that the Controller can effectively parallelise concurrent negotiations.

G. REMOTE ATTESTATION OVERHEAD

To evaluate the performance impact of the integration with the Keystone TEE, we measured the execution time of critical IPsec operations within the CCIPS Agent. Specifically, we analysed the latency of adding and deleting SAD and SPD entries, each performed independently, as well as the time required for a full rekeying procedure. For each test case, we measured the time from the first NETCONF notification to the configuration's installation in the kernel. We compared three execution scenarios: native execution (without the TEE), base execution of operations performed within the enclave, and execution with remote attestation enforced. In all the scenarios, we considered 2 hosts, which means that 1 IPsec tunnel needs to be configured.

Figure 17 illustrates the results of this benchmark. The X-axis represents the operation, while the Y-axis reports the execution time in milliseconds. The dotted line represents the native execution, the solid line depicts the base operation inside the TEE without attestation, and the dashed line represents the total time including the attestation overhead.

As shown, performing the operations in the TEE introduces a measurable performance penalty compared to the native execution.

When remote attestation is enabled, we observe a consistent overhead, less than 100 ms across all operations. This delay is due to the cryptographic workload required to measure the states of the enclave and the host, generate the attestation quote, and sign it. While this represents a non-negligible increase for the fastest operations, it is a highly acceptable trade-off for the robust security guarantees

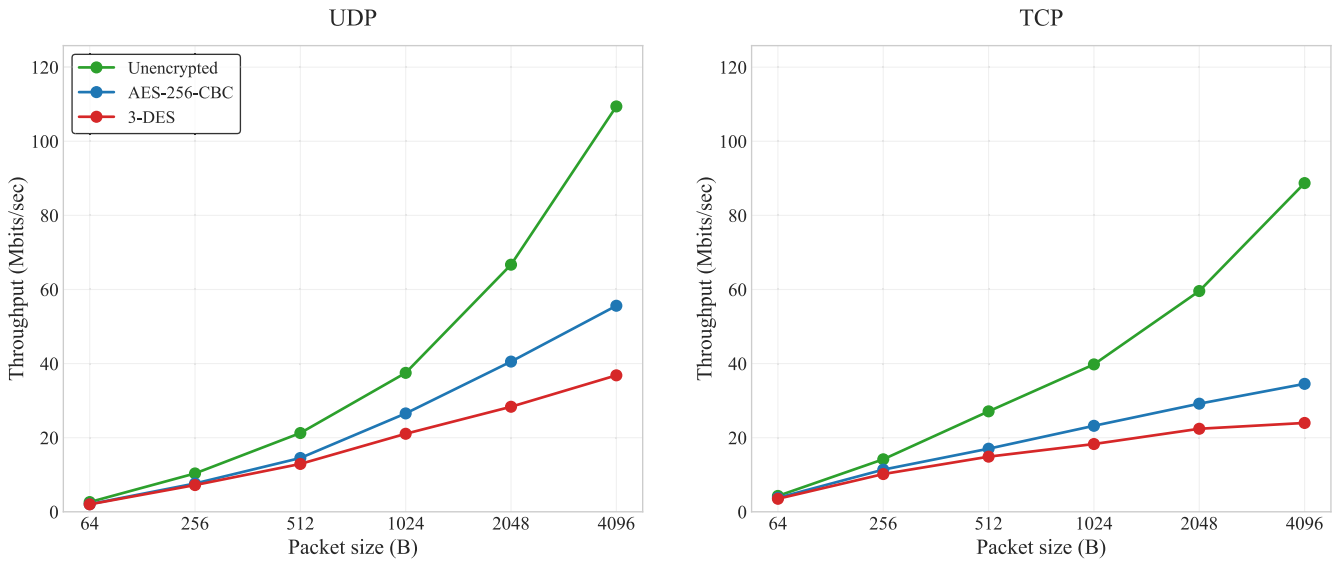

FIGURE 14. TCP and UDP throughput comparison across different packet sizes.

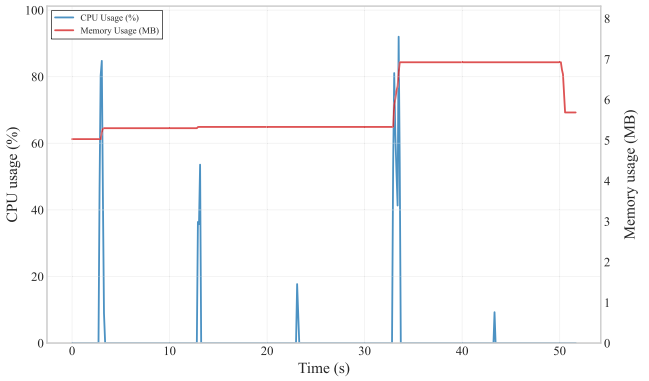

FIGURE 15. CPU and memory usage of the CCIPS agent over time.

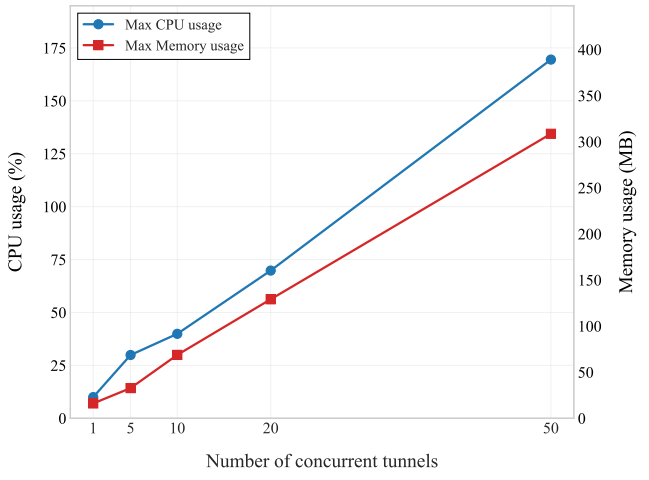

FIGURE 16. Maximum CPU and memory usage of the CCIPS Controller while establishing a different number of concurrent IPsec tunnels.

provided by the TEE. Furthermore, since RA is performed periodically, it rarely occurs alongside the aforementioned operations.

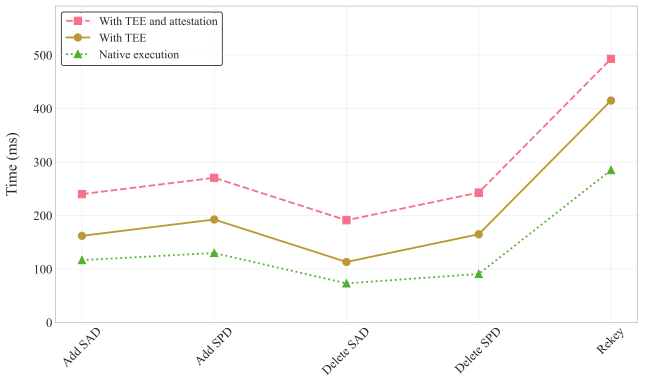

FIGURE 17. Overhead introduced by RA occurring during the various core operations.

H. COMPARISON WITH THE IKE APPROACH

For the sake of completeness, we used the RISC-V environment as an example to evaluate the overhead introduced by an IKE implementation to configure an IPsec tunnel (i.e., 2 hosts). The goal is to compare a common IKE solution with a native IKE-less approach without TEE.

To perform this test, we used the strongSwan v6.0.6 daemon. In brief, strongSwan [63] is a widely adopted implementation of the IKE protocols for establishing SAs and security policies between two peers. Similar to the previous evaluations, the data were collected from 20 independent runs.

Fig. 18 reports the average times gathered for the IKE phases, with related standard deviations. On average, the IKE authentication phase took about 90 ms total, with the IKE_SA_INIT and the IKE_AUTH phases requiring 27.20 ms and 62.45 ms, respectively. The time needed to install security policies (6.15 ms) and SAs (9.30 ms), instead, was less relevant in the overall connection establishment, which was around 238 ms.

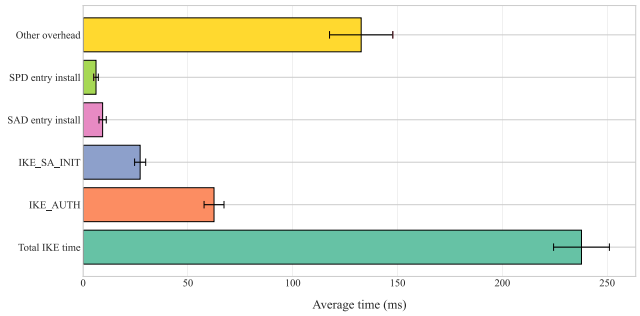

FIGURE 18. Operations performed with IKE.

When considering the total time required to retrieve information from the Controller, the overheads are comparable. As previously reported in Section IX-C, the mean time required to setup a single tunnel takes approximately 260 ms, which is very similar to the IKE approach. As a result, the IKE-less approach represents a viable alternative; it reduces the computational effort on the network node, making it particularly advantageous for constrained devices that lack the resources to use IKEv2.

X. CONCLUSION AND FUTURE WORK

In this paper, we introduced the TNED, a security architecture designed to address the challenges of modern, dynamic networks. By integrating the principles of SDN and NFV with hardware-based trust mechanisms, TNED establishes a secure and centrally managed environment for network services. The architecture's foundation rests on the use of TEEs and RA, which provide verifiable integrity and confidentiality for critical network functions, even when deployed on untrusted infrastructures.

The core concept of the TNED architecture is the CCIPS. This NSF leverages an IKE-less model to eliminate peer-to-peer negotiation, allowing a centralised controller to securely provision and manage SAs. Our implementation protects the entire IPsec configuration lifecycle within isolated enclaves, securing cryptographic material from a compromised host. We successfully validated the CCIPS implementation across two distinct TEE platforms (Keystone and Enarx), demonstrating the TNED portability, with performance overhead suitable for its intended use case.

In the future, we plan to extend this work in several key directions. A crucial next step is to benchmark the CCIPS on a physical RISC-V board, which will provide real-world performance metrics beyond the emulated environment used for the Keystone testbed. We also intend to enrich the TNED ecosystem by developing and integrating additional NSFs, such as firewalls or intrusion detection systems. Another extension would be to move the IPsec data plane inside the TEE, for example, by integrating a user-space packet processing framework such as DPDK or Vector Packet Processing (VPP) within the enclave, thereby keeping keys exclusively in TEE for their entire lifecycle. Alternatively, offloading the IPsec cryptographic operations on dedicated cryptographic engines could allow keys to be accessed directly from the TEE into hardware registers, bypassing the kernel entirely. Both approaches would eliminate the key exposure limitation identified in Section III, at the cost of increased deployment complexity and performance trade-offs that need further investigation. Finally, we plan to evaluate the full TNED architecture at scale, deploying the OAM control plane to manage hundreds of devices and heterogeneous NSFs to test the system's scalability and resilience under high workload.

REFERENCES

[1] F. Ciravegna, G. Bruno, and A. Lioy, "IKE-less IPsec: Enhancing security for IoT device networks," in *Proc. Italian Conf. Cyber Secur. (ITASEC)*, Apr. 2024, pp. 1–13.

[2] G. D'Onghia, F. Ciravegna, G. Bruno, M. A. Elorza Forcada, A. Pastor, and A. Lioy, "Securing 5G: Trusted execution environments for centrally controlled IPsec integrity," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, Jun. 2024, pp. 595–597, doi: 10.23919/IFIPNETWORKING62109.2024.10619852.

[3] K. Seo and S. Kent, *Security Architecture for the Internet Protocol*, document RFC-4301, Dec. 2005, doi: 10.17487/RFC4301.

[4] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, *Internet Key Exchange Protocol Version 2 (IKEv2)*, document RFC-7296, Oct. 2014, doi: 10.17487/RFC7296.

[5] GlobalPlatform. (May 2018). *Introduction to Trusted Execution Environments*. [Online]. Available: https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf

[6] A. Paju, M. O. Javed, J. Nurmi, J. Savimäki, B. McGillion, and B. B. Brumley, "SoK: A systematic review of TEE usage for developing trusted applications," in *Proc. 18th Int. Conf. Availability, Rel. Secur.*, Aug. 2023, pp. 1–15, doi: 10.1145/3600160.3600169.

[7] Confidential Computing Consortium. (Nov. 2022). *A Technical Analysis of Confidential Computing v1.3*. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3unlocked.pdf

[8] M. Li, Y. Yang, G. Chen, M. Yan, and Y. Zhang, "SoK: Understanding design choices and pitfalls of trusted execution environments," in *Proc. 19th ACM Asia Conf. Comput. Commun. Secur.*, Singapore, Jul. 2024, pp. 1600–1616, doi: 10.1145/3634737.3644993.

[9] D. Feng, Y. Qin, W. Feng, W. Li, K. Shang, and H. Ma, "Survey of research on confidential computing," *IET Commun.*, vol. 18, no. 9, pp. 535–556, Apr. 2024, doi: 10.1049/cmu2.12759.

[10] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proc. 15th Eur. Conf. Comput. Syst.*, Apr. 2020, pp. 1–16, doi: 10.1145/3342195.3387532.

[11] A. Waterman, K. Asanovic̀, and J. Hause. (May 2024). *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture (Document Version 20211203)*. ISA Specifications, RISC-V International. [Online]. Available: https://drive.google.com/file/d/17GeetSnT5wW3xNuAHI95-SI1gPGd5sJ/view

[12] *Enarx*. May 25, 2026. Accessed: [Online]. Available: https://enarx.dev/docs/Start/Enarx

[13] Z. Sha, C. Shepherd, A. Rafi, and K. Markantonakis, "Control-flow attestation: Concepts, solutions, and open challenges," *Comput. Secur.*, vol. 150, pp. 1–23, Mar. 2025, doi: 10.1016/j.cose.2024.104254.

[14] F. Schwarz and C. Rossow, "SENG, the SGX-Enforcing network gateway: Authorizing communication from shielded clients," in *Proc. 29th USENIX Security Symp. (USENIX Secur.)*, Aug. 2020, pp. 753–770. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/schwarz

[15] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-NFV: Securing NFV states by using SGX," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 11, 2016, pp. 45–48, doi: 10.1145/2876019.2876032.

[16] J. Wang, Y. Yu, Y. Li, C. Fan, and S. Hao, "Design and implementation of virtual security function based on multiple enclaves," *Future Internet*, vol. 13, no. 1, pp. 1–23, Jan. 2021, doi: 10.3390/fi13010012.

[17] J. Medina, N. Paladiy, and P. Arlosz, "Protecting OpenFlow using Intel SGX," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2019, pp. 1–6, doi: 10.1109/NFV-SDN47374.2019.9039980.

[18] D. Goltzsche, S. Rüsch, M. Nieke, S. Vaucher, N. Weichbrodt, V. Schiavoni, P.-L. Aublin, P. Cosa, C. Fetzer, P. Felber, P. Pietzuch, and R. Kapitza, "EndBox: Scalable middlebox functions using client-side trusted execution," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2018, pp. 386–397, doi: 10.1109/DSN.2018.00048.

[19] F. Schwarz, "TrustedGateway: TEE-assisted routing and firewall enforcement using ARM TrustZone," in *Proc. 25th Int. Symp. Res. Attacks, Intrusions Defenses*, Oct. 2022, pp. 56–71, doi: 10.1145/3545948.3545961.

[20] S. A. Atiiq and A. C. Risdianto, "Demystifying AMD SEV performance penalty for NFV deployment," in *Proc. 13th Int. Conf. Netw., Commun. Comput.*, Dec. 2024, pp. 1–8, doi: 10.1145/3711650.3711651.

[21] L. Jacquin, A. L. Shaw, and C. Dalton, "Towards trusted software-defined networks using a hardware-based integrity measurement architecture," in *Proc. 1st IEEE Conf. Netw. Softwarization (NetSoft)*, Apr. 2015, pp. 1–6, doi: 10.1109/NETSOFT.2015.7116186.

[22] G. López-Millán, R. Marín-López, F. Pereñíguez-García, O. Canovas, and J. A. P. Espín, "Analysis and practical validation of a standard SDN-based framework for IPsec management," *Comput. Standards Interfaces*, vol. 83, pp. 1–13, Jan. 2023, doi: 10.1016/j.csi.2022.103665.

[23] M. Coughlin, E. Keller, and E. Wustrow, "Trusted click: Overcoming security issues of NFV in the cloud," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2017, pp. 31–36, doi: 10.1145/3040992.3040994.

[24] N. Paladi and C. Gehrmann, "TruSDN: Bootstrapping trust in cloud network infrastructure," in *Proc. Secur. Privacy Commun. Netw. (SecureComm)*, Guangzhou, China, Oct. 2016, pp. 104–124, doi: 10.1007/978-3-319-59608-26.

[25] F. Schwarz, K. Do, G. Heide, L. Hanzlik, and C. Rossow, "FeIDo: Recoverable FIDO2 tokens using electronic IDs," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 2581–2594, doi: 10.1145/3548606.3560584.

[26] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "ShieldBox: Secure middleboxes using shielded execution," in *Proc. Symp. SDN Res.*, Mar. 2018, pp. 1–14, doi: 10.1145/3185467.3185469.

[27] H. Duan, C. Wang, X. Yuan, Y. Zhou, Q. Wang, and K. Ren, "LightBox: Full-stack protected stateful middlebox at lightning speed," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2351–2367, doi: 10.1145/3319535.3339814.

[28] M. A. Agca, "A holistic abstraction to ensure trusted scaling and memory speed trusted analytics," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2019, pp. 1428–1434, doi: 10.1109/CSCI49370.2019.00267.

[29] M. A. Agca, S. Faye, and D. Khadraoui, "Trusted distributed artificial intelligence (TDAI)," *IEEE Access*, vol. 11, pp. 113307–113323, 2023, doi: 10.1109/ACCESS.2023.3322568.

[30] M. A. Agca, S. Faye, and D. Khadraoui, "A survey on trusted distributed artificial intelligence," *IEEE Access*, vol. 10, pp. 55308–55337, 2022, doi: 10.1109/ACCESS.2022.3176385.

[31] S. Elizalde, A. AlSabeh, A. Mazloum, S. Choueiri, E. Kfoury, J. Gomez, and J. Crichigno, "A survey on security applications with SmartNICs: Taxonomy, implementations, challenges, and future trends," *J. Netw. Comput. Appl.*, vol. 242, pp. 1–31, Oct. 2025, doi: 10.1016/j.jnca.2025.104257.

[32] U. Andrei-Alexandru and B. Călin, "Reconfigurable SmartNICs: A comprehensive review of FPGA shells and heterogeneous offloading architectures," *Appl. Sci.*, vol. 16, no. 3, pp. 1–35, Feb. 2026, doi: 10.3390/app16031476.

[33] R. Lal, J. B. Anderson, and A. Jackson, "Data processing unit's entry into confidential computing," in *Proc. 12th Int. Workshop Hardw. Architectural Support Secur. Privacy*, Oct. 2023, pp. 56–63, doi: 10.1145/3623652.3623670.

[34] Y. Zhou, M. Wilkening, J. Mickens, and M. Yu, "SmartNIC security isolation in the cloud with S-NIC," in *Proc. 19th Eur. Conf. Comput. Syst.*, Apr. 2024, pp. 851–869, doi: 10.1145/3627703.3650071.

[35] M. You, J. Nam, M. Seo, and S. Shin, "HELIOS: Hardware-assisted high-performance security extension for cloud networking," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2023, pp. 486–501, doi: 10.1145/3620678.3624786.

[36] D. Giantsidi, J. Pritzi, F. Gust, A. Katsarakis, A. Koshiba, and P. Bhatotia, "TNIC: A trusted NIC architecture: A hardware-network substrate for building high-performance trustworthy distributed systems," in *Proc. 30th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2025, pp. 1282–1301, doi: 10.1145/3676641.3716277.

[37] J. Hao, Y. Li, H. Bai, X. Dong, H. Wang, and Y. Xiao, "DPU-enhanced network security situation awareness model for new power systems," in *Proc. 3rd Int. Conf. Algorithms, Data Mining, Inf. Technol.*, Sep. 2024, pp. 294–299, doi: 10.1145/3701100.3701161.

[38] A. Panda, S. Han, K. Jang, M. Walls, S. Ratnasamy, and S. Shenker, "NetBricks: Taking the V out of NFV," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, Savannah, GA, USA, Nov. 2016, pp. 203–216.

[39] R. Poddar, C. Lan, R. A. Popa, and S. Ratnasamy, "SafeBricks: Shielding network functions in the cloud," in *Proc. 15th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Renton, WA, USA, Apr. 2018, pp. 201–216.

[40] J. Thalheim, H. Unnibhavi, C. Priebe, P. Bhatotia, and P. Pietzuch, "Rkt-Io: A direct I/O stack for shielded execution," in *Proc. 16th Eur. Conf. Comput. Syst.*, Apr. 2021, pp. 490–506, doi: 10.1145/3447786.3456255.

[41] M. Li, S. Srivastava, and M. Yan, "Bridge the future: High-performance networks in confidential VMs without trusted I/O devices," 2024, *arXiv:2403.03360v1*.

[42] M. Simon, S. Warter, S. Gallenmüller, and G. Carle, "TEE time at P4-performance analysis of trusted execution environments for packet processing," in *Proc. IEEE 11th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2025, pp. 348–352, doi: 10.1109/NETSOFT64993.2025.11080555.

[43] W. Li, F. Lin, and G. Sun, "SDIG: Toward software-defined IPsec gateway," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Singapore, Nov. 2016, pp. 1–8, doi: 10.1109/ICNP.2016.7785316.

[44] M. Vajaranta, J. Kannisto, and J. Harju, "IPsec and IKE as functions in SDN controlled network," in *Proc. Int. Conf. Netw. Syst. Secur.*, Aug. 2017, pp. 521–530.

[45] G. Lopez-Millan, R. Marin-Lopez, and F. Pereniguez-Garcia, "Towards a standard SDN-based IPsec management framework," *Comput. Standards Interface*, vol. 66, pp. 1–90, Oct. 2019, doi: 10.1016/j.csi.2019.103357.

[46] J. A. Parra-Espín, R. Marín-López, G. López-Millán, F. Pereñíguez-García, and O. Canovas, "SDN-based automated rekey of IPsec security associations: Design and practical validations," *Comput. Netw.*, vol. 233, pp. 1–15, Sep. 2023, doi: 10.1016/j.comnet.2023.109905.

[47] J. A. Parra-Espín, R. Marín-López, and G. López-Millán, "Establishment of IPsec security associations with Diffie–Hellman following a SDN-based framework: Analysis and practical validation," *Comput. Netw.*, vol. 253, pp. 1–13, Nov. 2024, doi: 10.1016/j.comnet.2024.110720.

[48] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[49] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, Sep. 2016, doi: 10.1109/COMST.2015.2477041.

[50] E. Bravi, D. Gratiela Berbecaru, and A. Lioy, "A flexible trust manager for remote attestation in heterogeneous critical infrastructures," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2023, pp. 91–98, doi: 10.1109/CLOUDCOM59040.2023.00027.

[51] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, "Benchmarking open source NFV MANO systems: OSM and ONAP," *Comput. Commun.*, vol. 161, pp. 86–98, Sep. 2020, doi: 10.1016/j.comcom.2020.07.013.

[52] European Technical Standard Institute (ETSI). *Open Source MANO*. Accessed: May 25, 2026. [Online]. Available: https://osm.etsi.org/

[53] S. Hares, D. Lopez, M. Zarny, C. Jacquenet, R. Kumar, and J. P. Jeong, *Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases*, document RFC-8192, Jul. 2017, doi: 10.17487/RFC8192.

[54] R. Marín-López, G. López-Millán, and F. Pereñíguez-García, *A YANG Data Model for IPsec Flow Protection Based on Software-Defined Networking (SDN)*, document RFC-9061, Jul. 2021, doi: 10.17487/RFC9061.

[55] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, *Network Configuration Protocol (NETCONF)*, document RFC-6241, Jun. 2011, doi: 10.17487/RFC6241.

[56] M. Björklund, *YANG—A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, document RFC-6020, Oct. 2010, doi: 10.17487/RFC6020.
[57] Trusted Computing Group. *Device Identifier Composition Engine (DICE) Architecture*. Accessed: May 25, 2026. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/DICE-Layering-Architecture-r19pub.pdf
[58] E. Bravi, S. Sisinni, L. Ferro, V. Donnini, and A. Lioy, "Implementation of the TCG DICE specification into the keystone TEE framework," *IEEE Access*, vol. 13, pp. 142284–142303, 2025, doi: 10.1109/ACCESS.2025.3596829.
[59] C. Metz, D. L. McDonald, and B. Phan, *PFKEY Key Management API, Version 2*, document RFC-2367, Jul. 1998, doi: 10.17487/RFC2367.
[60] CESNET. *Netopeer2 Github Repository (v2.1.49)*. Accessed: May 25, 2026. [Online]. Available: https://github.com/CESNET/netopeer2
[61] F. Ciravegna, E. Bravi, S. Sisinni, and A. Lioy, "Run-time enclave measurement in the keystone framework," *IoT*, vol. 7, pp. 1–25, Jun. 2026, doi: 10.3390/iot7020048.
[62] J. Catalano, E. Bravi, S. Sisinni, and A. Lioy, "Enabling integrity measurement for secure applications in the enarx framework," *J. Netw. Syst. Manag.*, vol. 34, pp. 1–33, Oct. 2025, doi: 10.1007/s10922-025-09983-4.
[63] The strongSwan Team. *strongSwan*. Accessed: May 25, 2026. [Online]. Available: https://strongswan.org/

FLAVIO CIRAVEGNA received the M.Sc. degree in computer engineering (specializing in cybersecurity) from the Politecnico di Torino, where he is currently pursuing the Ph.D. degree in computer engineering. He is a member of the TORSEC Cybersecurity Research Group. His current research interests include trusted and confidential computing, remote attestation, and networks security.

GIACOMO BRUNO received the M.Sc. degree in computer engineering from the Politecnico di Torino. His current research interests include trusted computing, trusted execution environments, remote attestation, and trusted channels.

MATTIN ANTARTIKO ELORZA FORCADA received the M.Sc. degree in telecommunications engineering from the Universidad Politécnica de Madrid, Spain. He joined Telefónica Innovación Digital, in 2023, where he is currently a Research and Development Engineer with the CDO Unit. For the past three years, he has been involved in European and national research projects addressing secure, programmable, and next-generation networks infrastructures. His research interests include post-quantum cryptography, software-defined networking, networks security, and trustworthiness in communication networks.

ANTONIO PASTOR is currently an Industrial Engineer from the Carlos III University of Madrid, in 1999. He joined Telefónica I+D Research Group, in 1999, working on the design and deployment of different networks worldwide. Since 2006, he has been working as an Expert in networks security at Telefónica Innovación Digital. He is currently an Expert responsible for security research in the networks transport area with the CDO Unit. He has participated in many security research programs, holding several security certifications and patents.

ANTONIO LIOY received the M.Sc. degree (summa cum laude) in electronic engineering and the Ph.D. degree in computer engineering. He is currently a Full Professor with the Politecnico di Torino, where he leads the TORSEC Cybersecurity Research Group. His current research interests include networks security, policy-based systems protection, trusted computing, and electronic identity.