IKE-less IPsec for Centralized Management of Network Security

Flavio Ciravegna[1,*,†], Giacomo Bruno[1,†] and Antonio Lioy[1,†]

[1]Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy, Dipartimento di Automatica e Informatica

Abstract
In the realm of network security, the implementation of robust security measures is crucial to safeguard sensitive data and ensure the integrity of communication channels. To this end, the IPsec protocol enables the secure communication at network level. Initially reliant on manual configuration, IPsec evolved with the introduction of the Internet Key Exchange (IKE) protocol, which streamlines the establishment of Security Associations between network endpoints. However, the overhead associated with IKE can be impractical for resource-constrained IoT devices. Consequently, the IETF introduced the concept of IKE-less IPsec. This strategy aims to move the IKE logic from the network nodes to a centralized control point. Therefore, the network device is only required to support IPsec. This paper delves into the potential of the IKE-less approach to enhance security within Software Defined Networks, particularly in IoT scenarios. We analyse in detail the features of IKE-less IPsec and compare it with the traditional IKE approach. Then, we discuss our designed solution to protect the control infrastructure. The proposal leverages established solutions, such as Trusted Execution Environment and Hardware Security Modules, to protect this kind of setup.

Keywords
IKE, IKE-less, IPsec, Trusted Execution Environment, SDN Security

1. Introduction

In the ever-evolving landscape of network security, the identification of the ideal network level to implement security is not a trivial task. Nowadays, organizations often adopt a defence-in-depth strategy, implementing security measures at multiple layers of the OSI model to provide comprehensive protection against a wide range of security threats. As we ascend the stack, security functions become more specific, allowing for a granular control over users and data while remaining independent of the underlying network. On the other hand, this approach introduces additional overhead, leaving more room for Denial of Service (DoS) attacks. Moreover, at higher levels, the security often relies on the developers. Given the common attitude of overlooking security risks [1], a human error can compromise the security of a whole network.

As a result, the Internet Engineering Task Force (IETF) defined the IP security (IPsec) architecture, recognising that security is not just a concern confined to individual software components,

ITASEC 2024: The Italian Conference on CyberSecurity, April 8-12, 2024, Salerno, Italy
*Corresponding author.
†These authors contributed equally.
flavio.ciravegna@polito.it (F. Ciravegna); giacomo.bruno@polito.it (G. Bruno); antonio.lioy@polito.it (A. Lioy)
0009-0006-6668-8022 (F. Ciravegna); 0009-0005-6975-4235 (G. Bruno); 0000-0002-5669-9338 (A. Lioy)

but an aspect that should consider the entire network infrastructure. Thanks to this solution, the applications – operating at higher level – can avoid the implementation of the security services directly into their code.

Initially, deploying IPsec required the manual configuration of Security Associations (SAs), that define the parameters for secure communication between network entities. While effective, this manual configuration process proved to be prone to errors and time-consuming, particularly in large-scale network environments.

To address these challenges, the IETF introduced the Internet Key Exchange (IKE) protocol, with IKEv2 currently considered as the standard version [2, 3]. IKE serves as a fundamental component of IPsec deployments, automating the negotiation and establishment of SAs between network endpoints.

Modern networks encompass a wide range of device categories, with different computing capabilities. Given the exponential growth of the Internet of Things (IoT), ensuring robust security mechanisms becomes imperative for safeguarding sensitive data and maintaining the integrity of communication channels. Nevertheless, their resource limitations pose challenges in the implementation of conventional security protocols. While highly effective, the overhead associated with the IKE protocol can be impractical for resource-constrained IoT devices.

In response, a novel approach known as IKE-less IPsec is emerging as a promising solution to strike the delicate balance between security and efficiency. G. López-Millán et al. [4, 5] laid the groundwork for an IKE-less IPsec design, which was subsequently standardized by the IETF. This framework enables the configuration of IPsec SAs in Software Defined Networks (SDNs), aligning with the guidelines outlined in the RFC *Interface to Network Security Functions (I2NSF)* [6]. The term Network Security Function (NSF) refers to specific functions within a network architecture that are dedicated to providing security-related services. These functions can include firewalls, Intrusion Detection and Prevention Systems, VPNs, etc. In this context, the Controller is a centralized entity responsible for managing and controlling the network infrastructure (i.e., the NSFs).

Even though the IKE-less solution seems promising, it remains imperative to protect the components involved in the IPsec setup. The usage of a Trusted Execution Environment (TEE) permits to isolate and protect the execution of critical tasks, such as SA and policy generation. Once the IPsec configuration is created, the Controller is able to transfer it by means of proper secure channels (i.e., TLS or SSH). At this point, the NSF can install the configuration in the kernel, without having to perform additional IKE negotiations with other NSFs.

This paper is structured as follows. Initially, Section 2 reports the background information required for a proper comprehension of the work. Section 3 and Section 4 present a comparison between the IKE and IKE-less approaches, along with the experimental results reported in literature. Section 5 describes our proposed design. Finally, Section 6 outlines the final considerations and planned future works.

2. Background

2.1. IPsec

IPsec is an architecture that defines a collection of security services for traffic operating at the IP layer, encompassing both the IPv4 and IPv6 environments [7]. IPsec establishes a division between trusted and untrusted networks, allowing to create Virtual Private Networks (VPNs) over untrusted networks (*tunnel mode*), or end-to-end secure packet flows (*transport mode*). Data passing through the boundary is governed by access policies, defined by the manager of the IPsec configuration. These policies determine whether packets can freely traverse the boundary, undergo further processing, or are rejected. Specifically, two specific protocols are defined:

- *Authentication Header (AH)*: it offers integrity, data authentication, and optionally anti-replay attack capabilities;
- *Encapsulating Security Payload (ESP)*: offers the same properties of AH, while providing confidentiality.

A fundamental concept in IPsec is the *Security Association (SA)*. It can be defined as a unidirectional logic connection between two IPsec hosts, where each SA can provide different security features. The term *unidirectional* is significant in this context, since establishing a secure communication channel requires the creation of two SAs at each host: one dedicated to inbound traffic and the other to outbound traffic.

The SAs are stored in a specific database, the *Security Association Database (SAD)*, where the parameters of each SA are specified. Each SAD entry includes the cryptographic algorithms used for encryption and authentication, along with the respective keys, the SA's lifetime, the endpoints' source and destination addresses (as well as port numbers), and other information. Similarly, the *Security Policy Database (SPD)* contains the security policies (DISCARD, BYPASS IPsec, PROTECT using IPsec) that must be applied to a specific IP flow. In order to provide an association between SPD and SAD entries, a third database is defined: the *Peer Authorization Database (PAD)*. This database is consulted before the SA creation, to authenticate the IKE peers. In particular, each PAD entry provides information about the authentication protocol and the authentication data.

IPsec offers versatility across various scenarios. For instance, it empowers remote employees to securely access corporate resources on the move, by establishing VPN tunnels. Additionally, IPsec enables site-to-site communication, relevant for multinational corporations with a distributed infrastructure. It can ensure the confidentiality and integrity of data exchanged among IoT device networks and within cloud-based environments, among other applications.

3. The IKE and IKE-less Cases

This section describes the two approaches to configure the IPsec databases, depending on whether the Controller is involved in the SA creation or not.

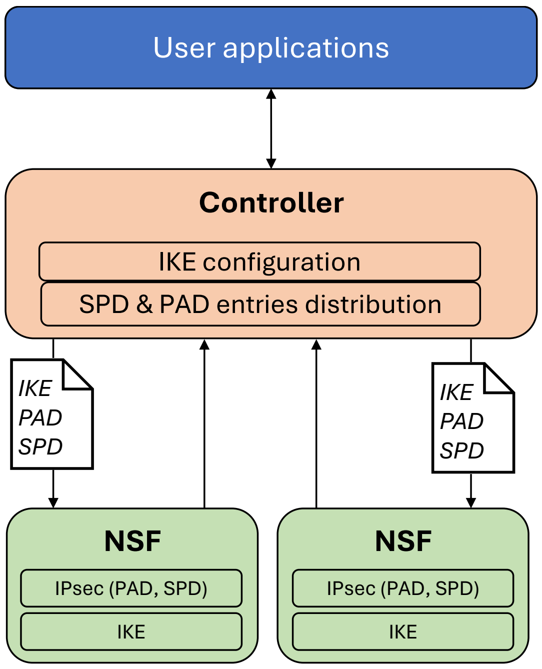

Figure 1: The I2NSF IKE architecture [5].

3.1. The IKE Case

The implementation of IKEv2 and the management of IPsec databases (SAD, SPD, and PAD) are carried out by the NSF. The Controller oversees the provision of IPsec connection information, including the IKEv2 credentials (certificates, pre-shared keys, etc.), as well as the configuration of the IKEv2 protocol itself [5]. Essentially, in this scenario (figure 1):

1. the I2NSF user provides the information about the endpoints and defines the IPsec requirements;
2. the Controller creates the IKEv2 configuration, as well as the SPD and PAD entries required by each NSF;
3. the NSF runs IKEv2 to create the IPsec SA and populates the SAD.

3.2. The IKE-less Case

In this case, the Controller is in charge of providing the required parameters to configure the SPD and the SAD. These databases are stored in the NSF, which does not implement the IKE protocol (figure 2). As a result, the logic for the management of the keys is moved to the Controller, whereas the NSF only offers support for the IPsec protocol.

Since the NSF does not support IKEv2, it is not required to manage the PAD. Additionally, the Controller is required to provide other security functionalities, such as:

- generation of the Initialization Vector (IV)
- generation of the SAs

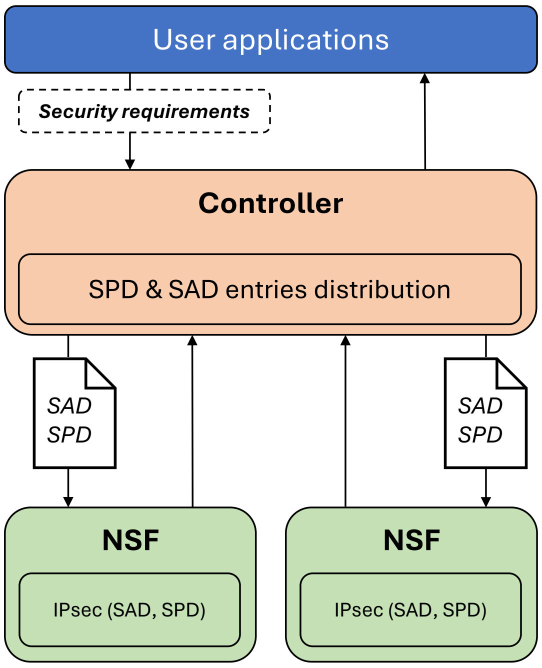

Figure 2: The I2NSF IKE-less architecture [5]

- rekey of the SA when notified by the NSF
- Security Parameter Index (SPI) generation
- algorithm selection
- generation of (pseudorandom) keys for the SAs
- etc.

In the IKE-less approach, the Controller can configure the SAD and the SPD in two different ways. With the *proactive* mode, the Controller provides all the SPD and SAD entries before any data packet arrives. Instead, with the *reactive* mode, the Controller configures only the SPD of the NSF. The needed SAs are provided only upon request by the NSF, through an appropriate notification.

4. Discussion of the IKE-less Case

4.1. Comparison with the IKE Approach

In the context of general purpose devices, the IKE case is typically easier to deploy. Generally, host and gateway machines (i.e., the NSFs) have access to a IKE implementation, thus it is just needed to exchange configuration parameters for the IKE protocol. The network node will be the entity in charge of managing the local configuration of IPsec, removing some processing complexity from the Controller. However, even if this solution can be considered straightforward to configure, a drawback is that the NSF requires additional resources compared to the IKE-less scenario. It may be necessary to provide additional storage for the IKE implementation, algorithms and parameters, and a discrete computational power. In particular, the latter can be considered as a critical point. As an example, each IPsec SA rekey operation (i.e., the procedure that involves the renewal of a SA after its lifetime has expired) may potentially involve a Diffie-Hellman exchange [5], hence a low-performance hardware may arise problems in the creation of the secure communication channel.

The IKE-less solution, on the other hand, emerges as a viable option for resource-constrained NSFs. By transferring the logic of IKE implementation to the Controller, the network devices are no more subject to heavy processing overhead. This approach facilitates the centralized management of security policies (i.e., the SPD) for each NSF. While offering different advantages, this solution introduces complexity into the Controller. The rekey process, for instance, poses a notable challenge: if the SA is not renewed before its hard lifetime expires, it will be removed and the communication will be interrupted. In this case, the Controller has to manage this process for all the supervisioned NSFs. This procedure becomes even more challenging if network latencies are taken into account. However, solutions have been recently proposed to address this issue [8].

In general, scalability issues may arise when the number of NSFs increases, This is a common problem in the context of *Software Defined Networking (SDN)* [9], a typical framework where IoT devices are involved. The scalability of SDN architectures has been a subject of various studies, which have investigated its dependency on factors such as the architecture types [10, 11], the quantity of deployed Controllers [12], or the placement of multiple Controllers [13]. As foreseen, the IKE-less solution relocates some security functionalities from the network nodes to the Controller. With the latter being responsible for generating the session keys instead of the NSFs, it becomes an even more critical entity. Hence, particular effort is required to safeguard it. The subsequent sections delve deeper into this aspect, offering insights into potential countermeasures to mitigate these vulnerabilities effectively.

4.1.1. Experimental Results in Literature

Different experiments have been conducted to evaluate the time needed by the Controller to configure the IPsec SAs. In [14], two tasks representative of SDN scenarios have been defined: a *mesh* and a *star* topology of network nodes. The mesh topology was chosen since it is common in datacenters, while the star topology is a typical scenario in SD-WAN networks. These tasks aimed at evaluating the IKE and IKE-less cases, under varying workloads and conditions. For the IKE-less case, both the proactive and reactive mode have been taken into account.

Mesh topology experiment. In the first task, it has been measured the configuration time within a mesh topology of n nodes. The experiment consisted of 15 executions for each value of n, ranging from $n = 2$ to $n = 30$. In this case, the Controller has to distribute $n \cdot (n - 1)$ unidirectional SAs. Consequently, in the worst case scenario the complexity is of order $O(n^2)$, indicating a quadratic tendency.

The data collected for both the IKE-less proactive and reactive modes revealed similar results. On average, the proactive mode required 17.5 seconds to configure a mesh network of 30 nodes, while the reactive mode took approximately 16 seconds. While it appears that the reactive mode requires less time for the configuration, this is true only under certain conditions. The previous comparison only encompasses the initial phase of the reactive mode. Specifically, in this phase, the Controller sends the security policies to the nodes. The SAs are installed only when requested by the specific node. If the configuration of the SAD is taken into account, the total time needed by the reactive mode significantly increases. Its performance decreases as the number of nodes requiring the SAD configuration simultaneously rises. Therefore, for those scenarios in which the SAD must be configured immediately for many nodes, the proactive mode is preferable. Conversely, if the SAD configuration can be spread over time, the reactive mode is better suited. This reduces the Controller's workload and prevents unnecessary population of NSFs' kernels with potentially unused IPsec SAs.

Considering the IKE case, the Controller has to provide the configuration for SPD, PAD, and IKE. It was observed that the Controller can furnish the configuration details for a mesh network comprising 30 nodes in roughly 13 seconds, slightly less than with the IKE-less approaches. Nonetheless, the configuration time still exhibits a quadratic trend.

As a result, the IKE case shows better configuration times than the other approaches. However, those values refer only to the IKE configuration time. The IKE negotiation time is not included, since it is performed after the initial configuration. Therefore, even though the performance of the IKE-less proactive approach may not match the performance of the IKE mode, the Controller provides the nodes with *all* the necessary information to establish the IPsec SAs. Hence, this eliminates the need for subsequent negotiation among the nodes, avoiding the introduction of additional traffic into the network.

Star topology experiment. The second task was centred on configuring IPsec within a star topology, wherein nodes form individual connections with a central node. Similarly to the first task, 15 executions for n nodes have been conducted, where n is in the range of 2 to 30. In this case, $n - 1$ nodes needs to be connected with the central node. Consequently, $2 \cdot (n - 1)$ unidirectional SAs must be configured for the whole network. This demonstrates a complexity of order $O(n)$. Thus, the configuration takes linear time with respect to the number of nodes.

All approaches completed the configuration of 30 nodes in approximately 7 seconds, indicating comparable performance across the board. However, the workload on the Controller in the IKE-less reactive mode was notably lighter compared to the mesh topology scenario. Consequently, the star topology appears more suitable for implementing the IKE-less reactive mode. This advantage is particularly evident in the worst-case scenario, where simultaneous configuration of the SAD is required for all nodes.

4.2. Centralized Architecture in SDN

The design of the IKE-less IPsec solution considers its potential integration within an SDN architecture. Indeed, the concept of SDN suggests the separation of network control and forwarding functions, while adopting a centralized software-driven management model, centred around a main Controller [15]. Within this framework, the main components are the *Control Plane*, the *Data Plane* and the *Application Plane*.

Specifically, the Control Plane assumes the responsibility of decision-making and the enforcement of network policies, and is composed by one (or more) Controllers. By decoupling the Control Plane from the Data Plane (i.e., the NSFs), the SDN architecture presents several benefits, including centralized management and programmability [16]. In particular, the Control Plane is able to perform flow control and forwarding, network orchestration and policy enforcement. This allows the network administrators to dynamically adjust the network behaviour depending on the requirements.

The Controllers communicate with the Data Plane through protocols like *OpenFlow* [17] or *NETCONF* [18]. Regarding the implementation of IKE-less IPsec, it has been chosen to adopt NETCONF. The latter, which stands for *Network Configuration Protocol*, is a standardized network management protocol used for managing and monitoring the configuration of network devices. It operates over a secure transport layer and uses XML to encode the configuration data exchanged between a network device and a central management system. In order to minimize network faults that can arise from manual configuration mistakes, NETCONF operates on devices according to the *YANG* model [19]. YANG is a language for data modelling, capable of providing an exhaustive description of all the data transferred between a NETCONF client and server. It models the data in a hierarchical architecture, organized as a tree, in which each node is characterised by a name. Eventually, the hierarchy can be extended, allowing for the addition of new nodes based on specific criteria. Each node is associated to a value or a set of child nodes.

4.3. Controller Configuration: Consistent and Shared State

Despite the adoption of an IKE or IKE-less model for the Controller, this component introduces a Single Point of Failure in the system that attackers can use to disrupt the proper and secure configuration of the network. Controllers can be integrated into the system according to either a centralized or distributed architecture. In the first case, a single component manages the entire network, which leads to an easier implementation since all the knowledge about the nodes and their configuration is kept in one place. However, this solution has the disadvantage of being a bottleneck for the received requests, enabling the adversaries to mount DoS attacks. Instead, the second case is more difficult to implement, due to the implementation and management of distributed nodes and information. Despite its challenges, it improves fundamental properties like resilience, throughput, latency, and scalability thanks to fault-tolerant mechanisms. Consequently, it mitigates the impact of DoS attacks on network operations [15].

As introduced before, the usage of a Controller reduces the risks brought by the manual configuration, which is prone to errors and requires a lot of time and effort for the administrator. However, these are not the only benefits that this architecture provides. The Controller receives the policies to be applied for all the nodes in the network, obtaining in this way a global view of the configuration of the connections.

4.4. Security in NETCONF Communication

In order to establish a secure IPsec channel, it is necessary to ensure that the data exchanged between Data and Control Plane are kept secure. Configuration information is inherently sensitive, encompassing passwords, usernames, service descriptions, and topological information. Transmitting this information without encryption and integrity checking exposes devices to different threats. Therefore, it is imperative to implement the communication protocol with particular attention to security. The protocol in this case is NETCONF, which is not bound to any particular transport protocol. However, it must be able to offer a well-defined set of functionalities. For instance, it is required to provide authentication, confidentiality, data integrity, and protection against replay attacks [18]. In line with the recommendations from the IETF, it is suggested to rely on well-established protocols, such as *Transport Layer Security (TLS)* or *Secure Shell (SSH)*, to encrypt the communication channels (e.g. [20]). For every implementation of NETCONF it is mandatory to support SSH.

5. Proposed Design

5.1. Security Considerations for Cryptographic Material Management

The IKE-less solution reduces the amount of resources needed by the IoT device. However, the Controller is responsible for the creation of the SAs and their related cryptographic material, introducing a new risk to the security of the IPsec communication. If the generated keys are not kept confidential and removed immediately after their transmission to the NSF, an attack on the Controller may provide the keys used to protect the secure channel, compromising the confidentiality and integrity of the transmitted data. For this reason, the Controller is obliged to remove from memory all the sensitive information as soon as is received by the IPsec node. This behaviour reduces the window of time available to an adversary for mounting attacks that aim to discover the cryptographic material used in IPsec channels. Even if the IKE solution is used, the same problem arises for the credentials inserted in the PAD entries. In this case, an adversary may compromise the IPsec traffic indirectly by impersonating the NSF during the generation of the SAs through the IKE protocol. The severity of this problem is even stated in the RFC describing the YANG model for the IKE and IKE-less I2NSF architectures [5].

Among the several works conducted in this context – e.g. [21, 22] – the threat model in [22] contains several threat vectors. Some of them are related to network communications (forged or faked traffic flows, attacks on control plane communications) but they can be easily countered by implementing secure channels as explained in previous sections. Others are related to the absence of trust in the system, including a lack of mechanisms to ensure trust between the Controller and management applications and a lack of trusted resources for forensics and remediation. Addressing these challenges requires robust certification and authentication procedures, secure and reliable logs, and traces of executed operations. Instead, the most critical threats concern the attacks and vulnerabilities in the system components, in particular the switches, the administrative stations, and the Controller. Compromising the latter could enable attackers to disrupt the entire network. For instance, an adversary may leak the cryptographic material used by the NSFs, especially in the IKE-less architecture, thus compromising the security properties guaranteed by the IPsec channels. Despite the several solutions that can be used to mitigate these risks, technologies such as Hardware Secure Modules (HSMs) [23] and Trusted Execution Environments (TEEs) [24] can serve as effective security controls.

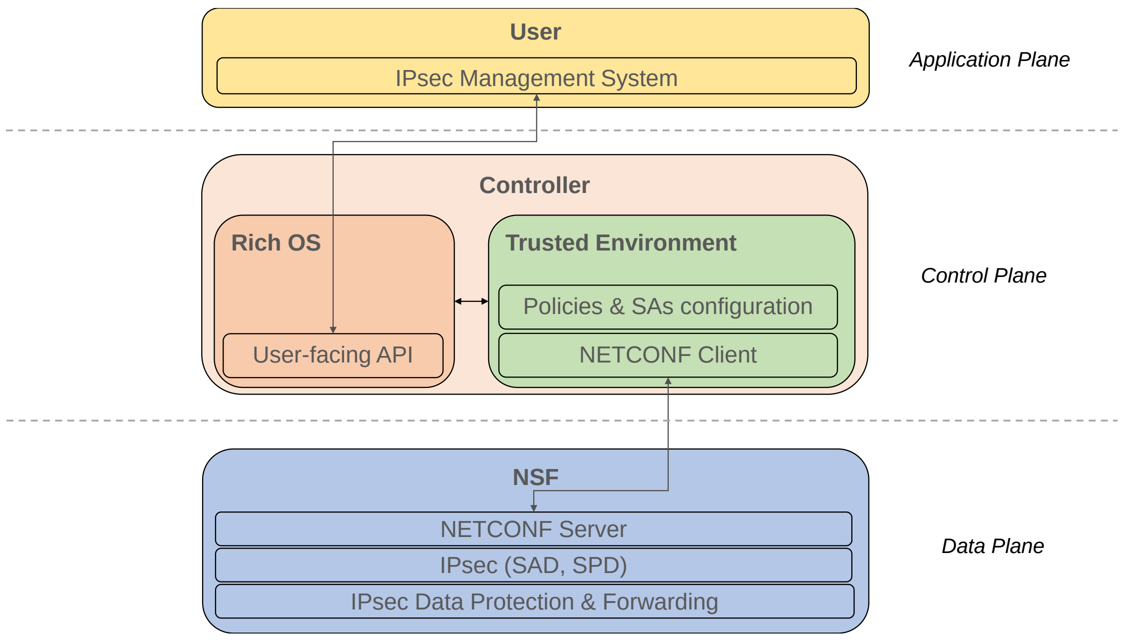

Figure 3: TEE integration in the Controller

5.2. Architecture Overview

This design leverages the concept of TEE [24], which can provide a secure environment that isolates the execution of specific applications from the untrusted system. The protection assured in this way reduces the attack vectors available to adversaries. Specifically, one of the TEE building blocks is secure storage. This component protects sensitive data stored in memory with operations such as sealing and unsealing, which consists of encrypting/decrypting the memory with keys bound to the application identity. Thanks to these security controls, the accessibility of the data to the attacker is significantly reduced. If the SAs and security policies are generated inside the trusted environment, the entries are protected even before the transmission to the NSF. As a result, this solution increases the assurance that the keys for the IPsec channel and other sensitive information are not compromised. Figure 3 shows the proposed architecture.

The system administrator, who is considered the user in this model, contacts the Controller according to a user-facing API through the IPsec management system. The interface between these two components is also known as *northbound interface (NBI)* and allows the specification of the policies in a high-level description language. Due to the absence of a standard NBI, we don't restrict the architecture to a specific interface, letting it be adaptable by the developers. The requests are served in the Controller by the code running in the *Rich Operating System* (Rich OS, i.e. untrusted execution environment). Then, the generation of the SAs and policies is executed in the TEE, together with the NETCONF Client to avoid the exposure of the entries to the untrusted environment. Finally, SAs and security policies are transmitted using NETCONF, protected by the establishment of a TLS or SSH channel. If asymmetric keys are required for the authentication methods, they can be securely stored and managed within the TEE. This further reduces the attack surface for an adversary, and consequently the possibility of discovering the cryptographic material used by the system. As the last step, the NETCONF server running on the node receives the SAs and policies which are installed in the SAD and SPD respectively. From now on, the NSF can protect specific traffic using the IPsec channels, being sure that nobody has obtained sensitive information from the Controller.

As an alternative to the secure storage provided by the TEE, an HSM can be used to generate and manage the keys used to protect the communication between the Controller and the NSF. When it is necessary to set up a TLS connection with the NSF, the HSM can generate the private key used for the handshake process. The keys stored in the HSM are typically protected by a strong authentication mechanism. Thus, these modules are protected from physical and logical attacks, such as tampering or extraction of the keys. The adoption of the HSM may also improve the performance of cryptographic operations, thanks to cryptographic processors and hardware accelerators.

6. Future Works and Conclusions

This paper proposed an analysis and a design of a management architecture for IPsec communication in IoT devices. Initially, the pros and cons of the IKE-less model are discussed, comparing them to the IKE one. Then, we have reported a possible threat model for the SDN architecture where the Controller is included. These considerations applied to a constrained IoT environment highlighted the necessity to improve the protection level of sensitive cryptographic material of the NSFs in the Controller. Despite the absence of the persistence of SAs in this node, protecting that data during their generation and before transmission is essential. For this reason, we proposed implementing the Controller into a TEE, which can protect critical functions in a secure environment. An alternative to that technology can be the HSM, which manages only the cryptographic material but improves the performance thanks to the hardware implementation of the cryptographic functions.

An implementation of the proposed solution is still a work in progress and a valid future work is to test its security and performance, comparing them with the original architecture. Moreover, the same protection mechanism adopted for the Controller can be extended to the NSF, ensuring the SAs and policies are not compromised at the NSF side.

Acknowledgments

This work was partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the European Union - NextGenerationEU. This work has also received funding from the SPIRS (Secure Platform for ICT Systems Rooted at the Silicon Manufacturing Process) project with Grant Agreement No. 952622 under the European Union's Horizon 2020 research and innovation programme.

References

[1] F. Corno, L. De Russis, L. Mannella, Helping novice developers harness security issues in cloud-IoT systems, Reliable Intelligent Environments 8 (2022) 261–283. doi: 10.1007/s40860-022-00175-4.

[2] C. Kaufman, Internet Key Exchange (IKEv2) Protocol, RFC-4306, 2005. doi: 10.17487/RFC4306.

[3] P. Eronen, Y. Nir, P. E. Hoffman, C. Kaufman, Internet Key Exchange Protocol Version 2 (IKEv2), RFC-5996, 2010. doi: 10.17487/RFC5996.

[4] G. López-Millán, R. Marín-López, F. Pereñíguez-García, Towards a standard SDN-based IPsec management framework, Computer Standards & Interfaces 66 (2019) 1–90. doi: 10.1016/j.csi.2019.103357.

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

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

[7] K. Seo, S. Kent, Security Architecture for the Internet Protocol, RFC-4301, 2005. doi: 10.17487/RFC4301.

[8] J. A. Parra Espín, R. Marín-López, G. López-Millán, F. Pereñíguez-García, O. Canovas, SDN-based automated rekey of IPsec security associations: Design and practical validations, Computer Networks: The International Journal of Computer and Telecommunications Networking 233 (2023) 1–15. doi: 10.1016/j.comnet.2023.109905.

[9] Open Network Foundation, SDN Architecture, issue 1.1, 2016. https://opennetworking.org/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf.

[10] M. A. Aglan, M. A. Sobh, A. M. Bahaa-Eldin, Reliability and Scalability in SDN Networks, in: ICCES-2018: 13th International Conference on Computer Engineering and Systems, Cairo (Egypt), December 18-19, 2018, pp. 549–554. doi: 10.1109/ICCES.2018.8639201.

[11] R. Odarchenko, O. Tkalich, G. Konakhovych, A. Abakumova, Evaluation of SDN network scalability with different management level structure, in: PIC S&T-2016: 3rd International Scientific-Practical Conference Problems of Infocommunications Science and Technology, Kharkiv (Ukraine), October 04-06, 2016, pp. 128–131. doi: 10.1109/INFOCOMMST.2016.7905357.

[12] L. Mamushiane, J. Mwangama, A. A. Lysko, Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?, in: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona (Italy), November 27-29, 2018, pp. 1–6. doi: 10.1109/NFV-SDN.2018.8725710.

[13] K. Saeed, M. O. Ullah, Toward Reliable Controller Placements in Software-Defined Network Using Constrained Multi-Objective Optimization Technique, IEEE Access 10 (2022) 129865–129883. doi: 10.1109/ACCESS.2022.3228039.

[14] G. López-Millán, R. Marín-López, F. Pereñíguez-García, O. Canovas, J. A. Parra Espín, Analysis and practical validation of a standard SDN-based framework for IPsec management, Computer Standards & Interfaces 83 (2023) 1–13. doi: 10.1016/j.csi.2022.103665.

[15] M. B. Jiménez, D. Fernández, J. E. Rivadeneira, L. Bellido, A. Cárdenas, A Survey of the Main Security Issues and Solutions for the SDN Architecture, IEEE Access 9 (2021) 122016–122038. doi: 10.1109/ACCESS.2021.3109564.

[16] Z. A. Bhuiyan, S. Islam, M. M. Islam, A. B. M. A. Ullah, F. Naz, M. S. Rahman, On the (in)Security of the Control Plane of SDN Architecture: A Survey, IEEE Access 11 (2023) 91550–91582. doi: 10.1109/ACCESS.2023.3307467.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review 38 (2008) 69–74. doi: 10.1145/1355734.1355746.

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

[19] M. Björklund, YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), RFC-6020, 2010. doi: 10.17487/RFC6020.

[20] M. Badra, A. Luchuk, J. Schönwälder, Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication, RFC-7589, 2015. doi: 10.17487/RFC7589.

[21] S. Scott-Hayward, G. O'Callaghan, S. Sezer, SDN Security: A Survey, in: SDN4FNS-2013: IEEE SDN for Future Networks and Services, Trento (Italy), November 11-13, 2013, pp. 1–7. doi: 10.1109/SDN4FNS.2013.6702553.

[22] D. Kreutz, F. M. Ramos, P. Verissimo, Towards secure and dependable software-defined networks, in: HotSDN-2013: 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong (China), August 16, 2013, pp. 55–60. doi: 10.1145/2491185.2491199.

[23] S. Mavrovouniotis, M. Ganley, Hardware Security Modules, in: K. Markantonakis, K. Mayes (Eds.), Secure Smart Embedded Devices, Platforms and Applications, Springer, 2014, pp. 383–405. doi: 10.1007/978-1-4614-7915-4_17.

[24] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted Execution Environment: What It is, and What It is Not, in: 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki (Finland), August 20-22, 2015, pp. 57–64. doi: 10.1109/Trustcom.2015.357.