Securing 5G: Trusted Execution Environments for Centrally Controlled IPsec Integrity

Grazia D'Onghia
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
grazia.donghia@polito.it

Flavio Ciravegna
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
flavio.ciravegna@polito.it

Giacomo Bruno
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
giacomo.bruno@polito.it

Mattin Antartiko Elorza Forcada
Telefónica CTIO
Telefónica Innovación Digital
Madrid, Spain
mattinantartiko.elorzaforcada@telefonica.com

Antonio Pastor
Telefónica CTIO
Telefónica Innovación Digital
Madrid, Spain
antonio.pastorperales@telefonica.com

Antonio Lioy
Politecnico di Torino
Dip. Automatica e Informatica
Torino, Italy
antonio.lioy@polito.it

***Abstract*—This demo introduces a novel method to enhance IoT communication security using a Trusted Execution Environment (TEE). Secure IPsec channels between two devices with x86 and RISC-V platforms are established by employing a platform equipped with a dedicated hardware Root of Trust. Enarx on x86 (equipped with Intel SGX) and Keystone on RISC-V machines serve as TEEs, ensuring integrity and confidentiality. This demo exhibits a workflow where the IPsec configuration is received from a centralized controller and is securely stored and managed within the TEE. By providing a comprehensive solution for securing IoT communications, the demonstration highlights the importance of TEEs in ensuring the integrity and confidentiality of interconnected devices in modern network infrastructures.**

I. Introduction

The SPIRS project [3] aims to create a secure platform with dedicated hardware Root of Trust and a versatile processor core for security services. This platform enables trusted communication channels required by 5G infrastructures.

The SPIRS platform integrates with networks via the Trusted Network Environment for Devices (TNED) [4]. The TNED hosts Network Security Functions (NSFs), offering interchangeable network and security features. In the SPIRS context, an NSF comprises:

1. *Trusted Application (TA)*, which runs within a TEE and manages critical data like keys and certificates;
2. *Rich Application (RA)*, which acts as a non-critical and untrusted process proxy between the TA and the external world.

This work was supported by 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. This work has also received funding from the SERICS (PE00000014) project under the NRRP MUR program funded by the European Union - NextGenerationEU.

The initial stage of this work was implemented by Hugo Ramón Pascual (ORCID: 0000-0002-3034-8150), while working for Telefónica CTIO.

ISBN 978-3-903176-63-8 ©2024 IFIP

One of the TNED NSFs is the Centrally Controlled IPsec (CCIPS), which is the focus of the demo.

A. Trusted Execution Environments (TEEs)

A TEE is an isolated execution area not accessible to the operating system, which is assumed to be untrusted. This prevents attackers from intercepting or tampering with the data during transmission. The isolation between trusted and untrusted areas provided by TEEs ensures that sensitive operations and data, like those involved in IPsec communication (cryptographic keys and secure key management), are protected from malware and other security threats in the non-secure world.

The SPIRS project uses Enarx [2], which leverages the Intel Software Guard Extension (SGX), and the Keystone Framework [5], which operates on RISC-V architectures. In this way, interoperability between TEEs on different architectures is demonstrated.

B. Threat Model

This context requires several assets to be protected: cryptographic keys, Security Association Database (SAD) and Security Policy Database (SPD) entries, the data transmitted over IPsec channels, and the Device Identity. The potential attacks can be network-based (such as Man-in-the-Middle), kernel attacks (such as privilege escalation), or physical attacks (such as hardware tampering).

C. Centrally Controlled IPsec

Beyond the traditional point-to-point IPsec configuration, CCIPS offers a centralized architectural approach for controlling IPsec endpoints. This system, based on the IPsec engine in IKE-less mode (without requiring IKE protocol [10]), consists of a centralized E2E manager (Controller) and two or more agents.

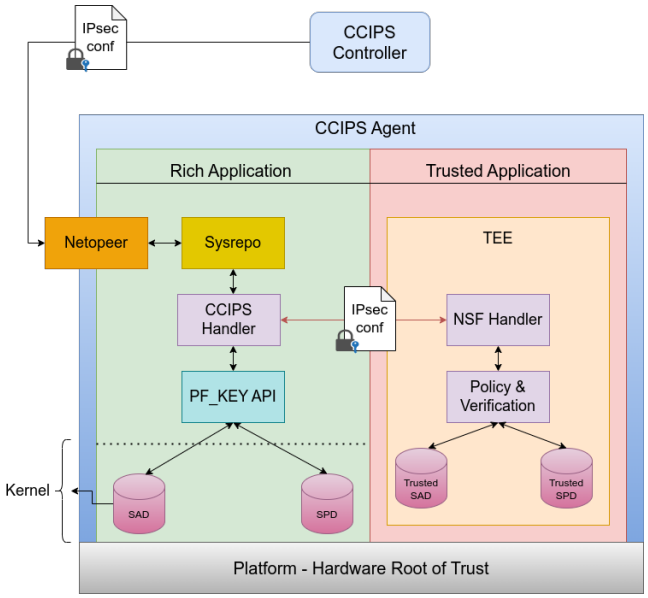

Fig. 1. Architecture of the CCIPS Agent

II. Design

The IPsec channel deployment is managed by two actors: the CCIPS Controller and the CCIPS Agent.

The CCIPS Controller is the core element of the system, and oversees requests from the Operation Administration and Management (OAM) component. It is responsible for the security settings of the IPsec tunnel within the network device, exchanged using the IKE-less data model. These settings include Agent identification, encryption, integrity, and initialization vectors associated with the IPsec protocol.

By adopting a NETCONF [7] interface, the CCIPS Controller implements the defined I2NSF Controller functionality and data model [1]. In this way, it configures and maintains the various IPsec parameters in each Agent and gathers pertinent data regarding the connectivity status.

Meanwhile, the CCIPS Agent is a network device that acts as one endpoint of the IPsec tunnel, which is based on the IPsec configuration received from the Controller. Moreover, it implements the NSF and is divided in two sections: the RA, which oversees all non-essential procedures deemed unsupported by the TEE, and the TA, which runs in the TEE and performs integrity verification.

The IPsec tunnel is implemented into the kernel through the PF_KEY Management API, which handles entries in the Security Association Database (SAD) and Security Policy Database (SPD). As shown in Figure 1, a trusted version of the SAD and SPD is stored in the TA before the installation into the kernel. Therefore, the integrity of the IPsec tunnel can be protected through a periodic verification of the entries by the TA. When requested by the Controller, the RA retrieves the entries from the kernel and forwards them to the TA for integrity verification. If the received entries do not match the ones installed in the TA, it means that the integrity of the IPsec tunnel has been compromised. Therefore, the Agent generates a critical event and forwards it to the Controller. However, the management of a failed verification process is out of the scope of this demo.

1) Enarx TA: Enarx enables running applications securely inside a TEE, without rebuilding the program or relying on many dependencies. This is achieved by offering a WebAssembly process-based runtime executed on Intel SGX. As a result, the need for trust relationships when programs are executed is reduced and problems like cross-compilation and disparate attestation methods between hardware suppliers are abstracted away.

2) Keystone TA: Keystone is a framework for building secure enclaves on RISC-V architectures. It offers memory isolation and cryptographic operations, enabling the development of applications with strong security guarantees.

The `SPIRS TEE SDK` is a template for developing applications using the Keystone framework. It wraps the Keystone SDK providing the Global Platform APIs [6] to the application developer.

The RA runs in QEMU (which implements a RISC-V toolchain) and when it launches the TA (enclave) the execution stops and the control is sent to the enclave.

III. Demo Implementation

The demo demonstrates the establishment of a secure IPsec tunnel between two CCIPS Agents, showing even the installation of a Security Association Database (SAD) entry and the re-keying process.

The architecture of the demo consists of three nodes: the CCIPS Controller, the CCIPS Agent with Enarx TA running inside a Docker container, and the CCIPS Agent with Keystone TA running inside QEMU. The SAD installation, depicted in Figure 2, is divided into two main phases: Key Generation (steps 1-6 in Figure 2) and IPsec Tunnel Configuration (steps 7-29).

A. Key generation

After being launched, the TAs generate the keypairs (steps 1-2), used by the Controller to encrypt sensitive IPsec material such as the encryption and integrity keys of the Security Association. Afterwards, upon the Controller's request (steps 3-5), the public keys and their associated certificate (for non-repudiation) are sent to the Controller (steps 4-6), while the private keys are stored inside the TAs.

B. IPsec tunnel configuration

When launching the application, the CCIPS Agents establish a session with Sysrepo [8], an open-source data store for YANG configuration modules, as shown in Figure 1. Then, the Agent subscribes to the changes of the SAD/SPD-related XPATH, which are received from the Controller by Netopeer2 [9], a NETCONF server.

To configure the IPsec tunnel, the Controller has to send two SAD entries, one for each direction of the tunnel.

Firstly, the Controller generates the SAD configurations (step 7) and encrypts them with the public keys received from the two Agents during the Key Generation phase (step 8). This configuration is in XML format, and the encrypted

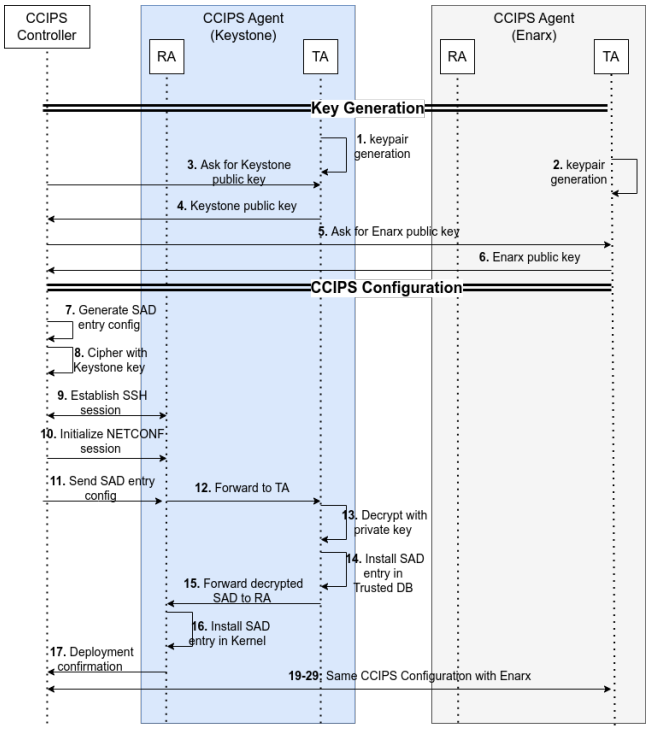

Fig. 2. Workflow for the SAD installation.

data consists of the encryption and integrity keys of the SAD entry.

Afterwards, the CCIPS Controller requests to install a SAD entry with the `<edit-config>` operation of the NETCONF protocol. This message is received by Sysrepo through the Netopeer2 server, thus notifying the CCIPS Agent with `sad_entry_change_cb` that there has been a modification in the Sysrepo datastore.

The SAD configuration is then forwarded to the TA (step 12), which decrypts with its private key the sensitive material (step 13) and stores the entry in the trusted database (step 14). Finally, the configuration is returned to the RA (step 15), which installs the SA in the kernel (step 16). If the entry is correctly installed, the confirmation is forwarded back to the Controller (step 17).

C. Re-keying Process

Once the SAD entry has been installed successfully, the re-keying process (shown in Figure 3) starts after a specific timeout. In the I2NSF IKE-less case, the nodes request the re-keying process before the entries' expiration (steps 1-2). Firstly, the Controller receives the notification from one of the Agents that a SAD entry is about to expire (step 3) and identifies which pair of Agents have the corresponding SAD entry installed in the kernel. Afterwards, a new set of SAD entries is generated (step 5) and sent to the corresponding Agents (step 6). At this point, the CCIPS Handler application proceeds as before, decrypting the cryptographic material (step 7) and installing the received entries firstly into the TEE and then into the kernel (step 8). Finally, the Controller asks to remove the old SAD from both Agents (steps 9-15).

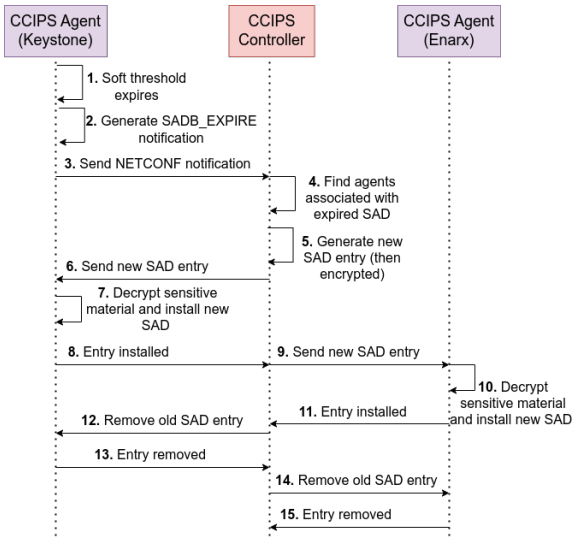

Fig. 3. Re-keying process

IV. Conclusion and Future Work

This demo is an initial implementation of the TNED developed for the SPIRS platform in a 5G network infrastructure. The increasing daily demand for IoT devices, coupled with the intensified need for ubiquitous interconnectivity, introduces additional complexities to system security. This experiment leverages integrity protection among heterogeneous IoT devices through TEEs, thus preventing attacks against the kernel and operating system. Future works concern improvements of this application from different points of view: firstly, it should be completely integrated in the SPIRS system and further evaluation about the isolation boundaries of the TAs should be performed. Finally, an additional investigation about the scalability of this solution is crucial for a comprehensive contribution to the industrial world.

References

[1] R. Marin-Lopez, G. Lopez-Millan, and F. Pereniguez-Garcia, "A YANG Data Model for IPsec Flow Protection Based on Software-Defined Networking (SDN)," RFC 9061, Jul. 2021.
[2] "Enarx," https://enarx.dev/.
[3] "Secure Platform for ICT Systems Rooted at the Silicon Manufacturing Process (SPIRS)," https://www.spirs-project.eu/.
[4] A. Pastor, H. Ramon, D. R. López, A. Cabrera, D. Arroyo, S. Galan, M. Liebsch, G. Yilma, S. Briongos, D. Margaria, A. Vesco, R. Guzzoni, A.Lioy, S.Sisinni, "TNED proof-of-concept," 2023.
[5] Dayeol Lee and David Kohlbrenner and Shweta Shinde and Krste Asanovic and Dawn Song, "Keystone: An Open Framework for Architecting Trusted Execution Environments," *15th European Conference on Computer Systems*, 2020.
[6] Global Platform, "TEE Client API Specification v1.0," https://globalplatform.org/wp-content/uploads/2010/07/TEE_Client_API_Specification-V1.0.pdf
[7] R. Enns, M. Björklund, A. Bierman and J. Schönwälder, "Network Configuration Protocol (NETCONF)," RFC 6241, Jun. 2011.
[8] "Sysrepo: YANG-based configuration and operational state data store for Unix/Linux applications," https://github.com/sysrepo.
[9] "Netopeer2," https://github.com/CESNET/netopeer2.
[10] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)," RFC 7296, Oct. 2014.