Paula Borreguero Vidal

MACsec and MACsec Key Agreement (MKA) implmentation

Bachelor's Thesis

Academic tutor: Antoni Martínez Ballesté
Professional tutor: Ricardo Martínez Elizalde

Bachelor's Degree in Telecommunications Systems and Services

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2024

This bachelor's thesis contains confidential information owned by Lear Corporation, with registered address at Carrer Fusters, 54, 43800 Valls (Tarragona).

The current document is the public version, that does not include any confidential information.

Abstract

This project involves implementing MACsec, a secure protocol that protects point-to-point ethernet links on a LAN, on the state-of-the-art automotive architecture. The automotive architecture is based on a zonal architecture, with different control modules distributed throughout the vehicle, all connected to the High Processing Control Module (HPCM). All these links will be MACsec protected, requiring the implementation of the MACsec Key Agreement protocol (MKA) for proper key management, essential for the success of the MACsec implementation. The project includes extensive study on MKA and MACsec, the development of test cases to ensure that MKA and MACsec perform properly, and the final implementation of the real setup for this automotive application.

Agreements

I would like to express my gratitude to several individuals. Firstly, I'd like to thank my LEAR tutor, Ricardo Martinez, for his technical support and encouragement throughout the project. I'd also like to extend my thanks to Judit Ferran for her assistance in setting up the real automotive application with the Ethernet Switch and for her support during the Trainee Program. Additionally, I value the guidance and communication from my university tutor, Antoni Martínez, during the project's development. I am also grateful to Lear Corporation for the opportunity to participate in the Trainee Program and develop this project. I have had an incredibly awesome experience working for this company since I got the opportunity to work with cutting-edge technologies and remain in a constant learning environment. Finally, I want to express my deep gratitude to my parents and my sister for their unwavering support over the past few months.

Table of Contents

List of figures

List of tables

1. Introduction

During my internship at LEAR Corporation, I worked on a project whose aim is to build awareness about MACsec and MACsec Key Agreement (MKA), a security protocol that operates at the link layer, and create test cases to demonstrate its functionality. The initial test cases confirmed the expected behavior of MACsec and MKA. Further test cases involved implementing MACsec between two Linux PCs connected through configurable Ethernet switches. This is a real-world application in the automotive industry where MACsec will be established between automotive ECUs. In more general terms, the development of this project led to an increased understanding of MACsec and MKA, and how they can be used to ensure security in automotive ECUs.

1.1 Objectives

To guide the research, it is needed to establish objectives first. The following are the objectives that we have identified:

- Analyze the MACsec Key Agreement protocol, as specified in the 802.1X standard.
- Develop and implement test cases to assess the effectiveness of secure communication between two endpoints configured with MKA.
- Analyze the results obtained from the test cases and draw meaningful conclusions.
- Implement the 802.1X/EAP protocol, which will involve generating digital certificates and setting up a certification authority.
- Implement the MKA protocol in an automotive platform.

2. Automotive Ethernet

2.1 Context and motivation

Before the 21st century, vehicles were simpler compared to the last decade. Each vehicle consisted of only a few basic components and cables, primarily used for power distribution and basic functionalities. However, as technology has advanced, cars have evolved into highly complex machines, often described as "computers with wheels."

In essence, the evolution of cars has transformed them from more mechanical devices into sophisticated technology, blending automotive engineering with cutting-edge electronics.

Modern cars have huge processing power, due to the extensive electronics integrated into each vehicle and the growing data transfer in vehicles. These vehicles follow a strategic layout known as a zonal architecture, where various specialized control modules (Electronic Control Units or ECUs) are distributed throughout the car.

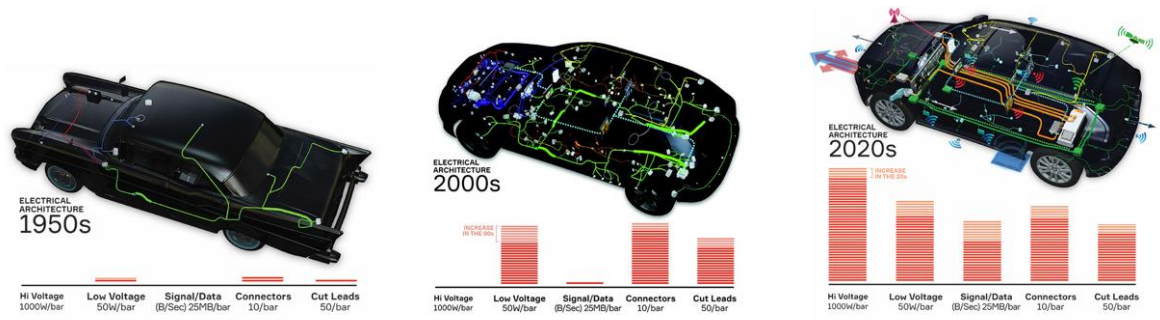


Figure 1. Automotive Ethernet context[1]

The ECUs (Electronic Control Units) are integrated computers that control various functions, such as engine management, braking systems, and airbag deployment, among others. The data transfer between these ECUs and a central processing unit is crucial for the vehicle's operation and this process involves using cables.

In the automotive context, the way different modules communicate with each other is through Control Area Network buses (CAN). CAN is a reliable and cost-effective method for transferring data between electronic modules. However, as more ECUs are added to the vehicle and due to the enormous amount of data transferred, ensuring network security to protect against cyber threats becomes a priority. This is where technologies like MACsec (Media Access Control Security) play a crucial role in safeguarding communication networks within the vehicle. Nevertheless, this transition also leads to an increase in the number of required cables, resulting in additional weight.

Cable weight significantly impacts a vehicle's fuel consumption. The three primary elements contributing to a car's weight include the engine and mechanical components, the external body, and the cables. By reducing cable weight, we can contribute to lowering car emissions.

However, the CAN bus, besides its cable weight, faces a significant speed disadvantage. Although it can handle data up to 1 MB/s, it falls short for modern car functionalities like video streaming. This is where Automotive Ethernet comes into play.

Automotive Ethernet involves replacing some of the existing CAN communication buses with Ethernet cables, equipping the vehicles with 100BASE-T1 (100Mbps) or 1000BASE-

[1] Image from https://www.aptiv.com/en/insights/article/evolution-of-vehicle-architecture

T1 (1000 Mbps). This transition enhances data transfer speed, increasing throughput and reducing cost in vehicle networks. Beyond speed improvements, it also brings additional benefits, including reduced cable weight, lower fuel consumption, and enhanced reliability. It uses one twisted pair cable for full duplex communication.

Figure 2. The future of Automotive Data Connectivity[2]

Ethernet is increasingly being implemented in vehicles due to its higher speed and bandwidth. This is essential for modern applications that require a large amount of data in vehicles. Additionally, it is important to highlight its advanced cybersecurity capabilities, such as MACsec, and its high scalability. As the number of ECUs and sensors in vehicles grows, Ethernet can seamlessly adapt to these changes.

2.2 Architecture

The zonal architecture is the latest advancement in car communication topology. The ECUs are organized in a zone architecture along with a vehicle compute module according to their internal locations in the vehicle.

In this system, the infrastructure of the vehicle features a central computer that will be directly linked to multiple zonal gateways. These gateways are responsible for controlling the electronic components of the car. This architecture offers high flexibility and make maintenance easier.

[2] Image from https://www.aptiv.com/en/solutions/connection-systems/catalog/hsca/hmtd

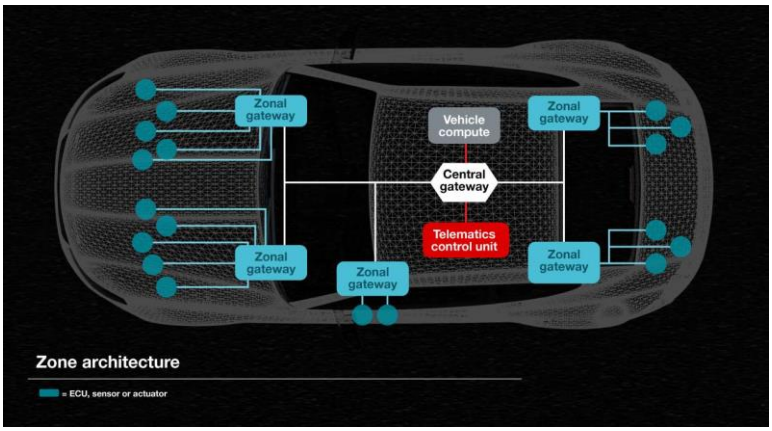


Figure 3. Zonal Control Module architecture[3]

The zone modules and the central gateway can communicate with one another using a low-bandwidth network, such as CAN. However, high-speed networks like Ethernet or PCIe are also a viable option due to their excellent reliability and smooth operation, especially in a range of vehicle temperatures.

In addition, there is a major concern with this topology improvement. The connection between the central computer and the zonal gateways becomes critical. If the connection is compromised, the entire automotive system will be at risk. Therefore, it is essential to have a network protocol that safeguards the communication between different automobile parts.

Enabling the IPsec protocol to protect vital connections was considered as a solution for the new design. However, this protocol has some limitations that prevent it from being used to its full potential, such as extending the packet header, which reduces network efficiency. It also requires dedicated CPU resources for optimal performance and provides end-to-End protection in situations where Point-to-Point protection is all that is needed. IPsec often relies on centralized ASIC technology to quickly encrypt and decrypt data. However, centralization can lead to bottlenecks when all traffic flows through a central location, potentially affecting network performance. This can lead to problems with available speed and bandwidth on high-speed networks.

Currently, Media Access Control Security (MACsec) is gaining interest in the automotive industry as the most attractive option. This protocol operates on layer 2 of the OSI model and offers data integrity, data authentication, and confidentiality. Furthermore, data integrity must be protected at the link layer, and MACsec is the only effective method that can guarantee every kind of data transmission.

There are several advantages of using MACsec over IPsec. The use of silicon in electronic components has significantly improved in recent years. The MACsec integrated PHYs have been developed with hardware offloading capabilities, enabling them to perform signature and encryption algorithms. This reduces CPU load and speeds up packet exchange compared to IPsec. Additionally, MACsec's zonal architecture allows devices to connect directly to one another without requiring packet routing. MACsec is faster, uses fewer specialized resources, and is a Point-to-Point protection protocol, unlike IPsec which is an End-to-End protection protocol. Thus, MACsec is a better choice.

On network devices, MACsec is enabled per port, thus encryption can be performed on particular ports without affecting network performance. It provides high-performance

[3] Image from SSZT211 Technical article | TI.com

encryption on fast links and is recommended for use on high-speed networks instead of IPsec. It is a simpler protocol that has minimal negative impact on performance and is very effective. In sectors such as the automotive industry, where bandwidth and speed are critical, this is essential for effective communication in networked vehicles.

The image below shows that IPsec encryption speed is slower than the link speed when the speed is greater than 40 Gb/s. In contrast, MACsec provides encryption on high-speed lines, making it the better choice for high-speed lines.

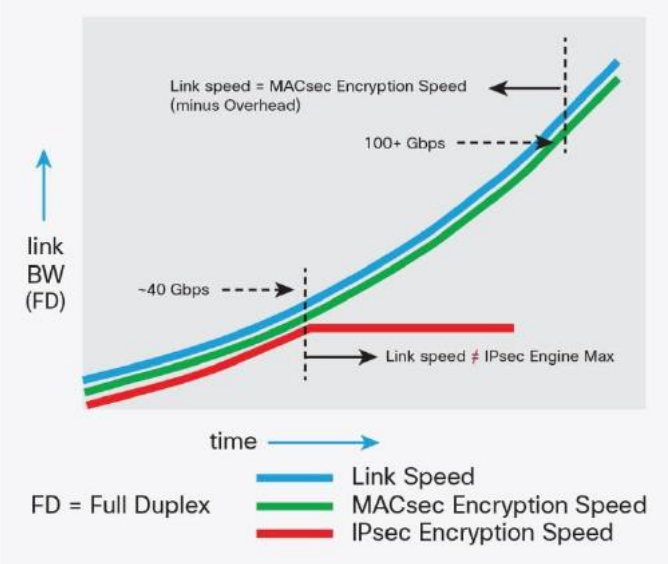


Figure 4. Link speeds[4]

However, MACsec has limited scalability and can only be used in point-to-point or point-to-multipoint network architectures. Nonetheless, it ensures point-to-point security in Ethernet links, providing confidentiality on the transmitted data in an ECU network and preventing unauthorized access that could lead to interruption or loss of data.

[4] Image from https://www.cisco.com/c/dam/en/us/td/docs/solutions/Enterprise/Security/MACsec/WP-High-Speed-WAN-Encrypt-MACsec.pdf

3. Definitions

Access priority: Priority assigned to a transmit request made at a common port by a MAC Security Entity (SecY).

Association Number (AN): A number used to identify a Secure Association (SA) that is concatenated with the Secure Channel Identifier (SCI).

Authentication exchange: Exchange of information for the purpose of authenticating an entity.

Authentication Server: An entity that offers an authenticator an authentication service. This service determines whether the Supplicant is permitted to access the service offered by the system where the Authenticator resides based on the credentials they have provided.

Authenticator: A device that helps other devices connected to the same LAN authenticate.

Cipher Suite: A collection of one or more algorithms intended to offer data integrity, authenticity, and confidentiality.

Client: The protocol entity that makes use of a service.

Cryptographic key: A parameter that controls how a cryptographic function works, such as:

a) Converting plain text to cipher text and back.

b) Coordinated production of keying materials

c) The computation or validation of digital signatures.

Data integrity: It is the quality of having not had any unauthorized changes made to it since it was created, transferred, or stored.

Extended packet number (XPN): A 64-bit packet number (PN) defined in IEEE Standard 802.1AE.

Frame: Data unit for the MAC protocol (MPDU).

Initialization vector (IV): A vector within a cryptographic technique that specifies the beginning of an encryption process.

Integrity check value (ICV): It is a value derived from the corresponding CAK. The ICV is sent with the protected data unit and is recalculated and compared by the receiver to detect data modification to ensure the integrity and authenticity of the protected MKPDUs.

Key Agreement Entity (KaY): A PAE entity in charge of MKA. It monitors a Secure Channel during its life cycle and assists in setting it up using a key agreement.

key management: It is the process of generating, storing, distributing, erasing, archiving, and application of keys in accordance with a security policy.

Key Server: It is a MACsec peer in the CA which creates and distributes Secure Association Keys (SAKs).

MACsec Protocol Data Unit (MPDU): It defines the frame layout, which is found in the MAC layer, the layer 2.

MAC Security Entity (SecY): The entity in charge of running a MAC Security protocol within a system. It is responsible of integrity protecting, validating and, if necessary, encrypt and decrypt to ensure confidentiality of user data. The SecY may be implemented either in software or hardware.

MAC Security TAG (SecTAG): A protocol header that starts with an EhterType and consists of several octets, that is prepended to the service data unit that the protocol client supplies.

MAC service data unit (MSDU): A sequence of zero or more octets that compose the data to be communicated with a single MAC Service request or indication.

Master key: A secret key that is used to derive one or more cryptographic keys that are used directly to secure data transfer.

Message authentication: The cryptographic guarantee that a message was not altered while in transit and that it came from a source with the right cryptographic credentials is provided if the message is received authenticated.

Packet Number (PN): A value that increases monotonically and is guaranteed to be unique for each MACsec frame sent using a given Secure Association Key (SAK).

Port: A MAC Service or MAC Internal Sublayer Service access point.

Port access entity (PAE): The protocol entity connected to a Port. It allows members of a Secure Connectivity Association (CA) to mutually authenticate one another and agrees the cryptographic keys and associated parameters to satisfy the requirements of the SecY.

Port Identifier: A 16-bit identifier that uniquely identifies each of a system's transmit Secure Channels (SCs) that share the same MAC address as part of their Secure Channel Identifier (SCI).

Protocol Data Unit (PDU): A data unit defined in a protocol and may include user data as well as protocol information.

Secret key: A cryptographic key used with a secret key cryptographic algorithm that is exclusively linked to one or more entities and should not be made public.

Secure Association (SA): A security relationship that ensures the security of frames transmitted between members of a Connectivity Association (CA).

Secure Association Identifier (SAI): An identifier for a Secure Association (SA). It is made up of the SCI and the two-bit Association Number (AN) that is allocated to the SA. It allows the receiving SecY to identify the SA and thus the SAK used to decrypt and authenticate the received frame.

Secure Association Key (SAK): The secret key derived from the CAK used to encrypt MSDU messages transmitted between two MACsec devices. Each SAK is associated with a SA.

Secure Channel (SC): A security relationship used to assure the security of frames being transmitted from one member of a Connectivity Association (CA) to the others.

Secure Channel Identifier (SCI): A unique identifier for a Secure Channel (SC).

Secure Connectivity Association (CA): A security relationship established between two or more MACsec devices attached to a single LAN network.

Secure Connectivity Association Key (CAK): A secret key that only members of a certain CA possess. Members of a particular CA will use it to extract the key material (ICK, KEK, and SAK keys) required for MKA and MACsec

Secure Connectivity Association Key Name (CKN): A text that identifies a CAK.

Secure Key Encrypting Key (KEK): Key derived from the CAK used for the encryption and decryption of the SAK.

Secured connectivity: Data transfer between two or more Controlled Ports that is protected by MACsec.

Short Secure Channel Identifier (SSCI): A 32-bit value that is specific to each SCI and is controlled by the key agreement protocol when using a certain Secure Association Key (SAK) in the context of all MAC Security Entities (SecYs).

Supplicant: An entity at one end of a point-to-point LAN segment that seeks to be authenticated by an Authenticator connected to the other end of that link.

User priority: The priority assigned with a transmit request that a MAC Security Entity's (SecY) Controlled Port receives.

4. Abbreviations and acronyms

AAA: Authentication, Authorization, and Accounting

AES: Advanced Encryption Standard

AN: Association Number

BPDU: Bridge Protocol Data Unit

CA: secure Connectivity Association

CAK: Secure Connectivity Association Key

CKN: Secure Connectivity Association Key Name

CRC: Cyclic Redundancy Check

DA: Destination Address

DHCP: Dynamic Host Configuration Protocol

EAP: Extensible Authentication Protocol

EAP-TLS EAP: Transport Layer Security

EAPOL: EAP over LANs

EDE: Ethernet Data Encryption device

EPON: Ethernet Passive Optical Network

ES: end station

GCM: Galois Counter Mode

ICK: ICV Key

ICV: integrity check value

IP: Internet Protocol

IV: Initialization Vector

KaY: MAC Security Key Agreement Entity

KDF: Key Derivation Function

KEK: Key Encrypting Key

KI: Key Identifier

LAN IEEE 802: Local Area Network

LMI: Layer Management Interface

LPN: Lowest acceptable PN

MAC: Media Access Control

MACsec: Media Access Control Security

MI: Member Identifier

MKA: MACsec Key Agreement protocol (IEEE Std 802.1X)

MKPDU: MACsec Key Agreement Protocol Data Unit

MN: Message Number

MPDU: MACsec Protocol Data Unit

MSDU: MAC Service Data Unit

MSK: Master Session Key

OLPN: Lowest acceptable PN for the Old Key

OLT: Optical Line Terminal

ONU: Optical Network Unit

PAC: Port Access Controller

PACP: Port Access Control Protocol

PAE: Port Access Entity

PDU: Protocol Data Unit

PN: Packet Number

PSK: Pre-Shared Key

QoS: Quality of Service

RADIUS: Remote Authentication Dial in User Service

RNG: Random number generator

SA: Secure Association or Source Address, as applicable

SAI: Secure Association Identifier

SAK: Secure Association Key

SC: Secure Channel

SCB: Single Copy Broadcast

SCI: Secure Channel Identifier

SecTAG: MAC Security TAG

SecY: MAC Security Entity

SL: Short Length

SSCI: Short Secure Channel Identifier

VLAN: Virtual LAN

XPN: Extended Packet Number

5. MACsec

MACsec, as defined in IEEE 802.1AE, is a network security standard at the link layer of the OSI model within Ethernet Local Area Networks (LANs). It guarantees connectionless data confidentiality, data integrity, and data origin authenticity for wired ethernet LANs. MACsec is typically run in the "hop-by-hop" mode. This means that Ethernet frames are protected by wire but not inside a switch, requiring hardware support.

MACsec provides security for point-to-point and point-to-multipoint on links between connected hosts at layer 2, protecting against threats like Denial of Service (DoS) attacks, intrusions, Man-in-the-Middle (MITM) attacks, masquerading, passive wiretapping, and replay attacks. All LAN traffic, including traffic from higher layer protocols and from the link layer such as DHCP and ARP, can be protected by MACsec.

Secure links are established when hosts exchange and verify security keys, which can be configured manually or generated dynamically depending on the chosen security mode within MACsec. Furthermore, the IEEE 802.1AE standard can be used alongside authentication methods in IEEE 802.1X.

It operates autonomously, unaffected by higher-level security protocols. Providing protection for all layer 2 protocols, encompassing unicast, multicast, and broadcast traffic. MACsec implementations are situated close to the hardware, ensuring high-performance encryption at port speed.

MACsec packet processing and crypto processing are usually done in hardware. Software can also be used for MACsec at low speeds, but this is not helpful for the automotive industry.

5.1 IEEE 802.1AE-2018 terminology

Secure Connectivity Association (CA): It is a security relationship created by MKA established between two or more MACsec devices attached to a single LAN network.

SC (Secure Channel): It is a security relationship for frames transmitted from a member of a CA to other members of the same CA, unidirectional point-to-pint or point-to-multipoint communication. A Secure Channel (SC) is supported by a sequence of Secure Associations (SAs), enabling periodic use of new keys without ending the relationship.

The identifier of a SC is called SCI (Secure Channel Identifier) and is composed of the MAC address (48 bits) of the device along with the port identifier (16 bits). All the SCs use the same Cipher Suite at one time.

SA (Secure Association): Each Secure Channel (SC) comprises a sequence of SAs, each one uses a different encryption secret key (SAK) or a set of them (at least two, defined in the standard) to provide the MACsec service guarantees and security services for a sequence of transmitted frames. This partnership is established and maintained through key agreement protocols (MKAs).

The identifier of an SA is called SAI (Secure Association Identifier).

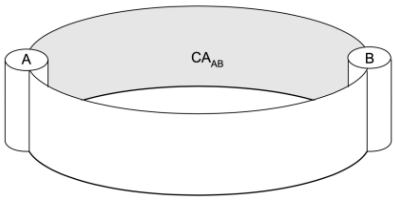


Figure 5. CA between two nodes

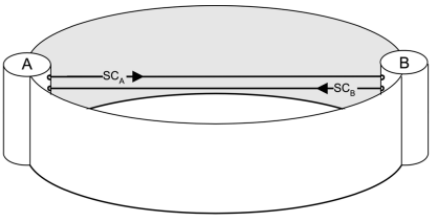


Figure 6. SC between two nodes

5.2 MACsec security services

Once the secure session is established, they establish the following security services:

- **Confidentiality:** The MSDU can either be unencrypted or encrypted to ensure data confidentiality, MACsec encryption is optional and user configurable. The AES-GCM algorithms are employed for this purpose and the data in the Ethernet frame cannot be viewed by anybody on the link.

- **Integrity:** MACsec adds 8 bytes header, and 16 bytes tail to all ethernet frames that travel throw MACsec secured link.
  The Integrity Check Value (ICV) is exclusive and unique in each MACsec packet. This value is responsible for verifying the integrity of various components, including the MAC addresses (both source and destination), the SecTAG, and the MSDU (User Data). This allows any unauthorize changes to be detected.

- **Authenticity of origin:** With the ICV is guaranteed that the MAC packet received has been send by whoever is indicated in the filed Source MAC address of the packet.

- **MACsec-only traffic filtering:** With the first two bytes of the SecTAG, which indicates the EtherType of MACsec (0x88e5), it is possible to distinguish a MACsec packet from any other unprotected packet.

- **Anti-replay protection:** Each packet comes with a unique PN assigned for each SAK. This PN serves the purpose of detecting and discarding forwarded packets, using the ReplayProtection and ReplayWindow parameters.

- **Anti-delay protection:** Using the bounded time parameter, which value is usually under 2 seconds, the anti-delay protection can be guarantee. This value establishes a time limit for the reception of the packet since the moment is sent by the source. It serves to discard all those packets that take more time to be received, what means that they could have been intercepted and modified between the source and the destination.

- **Hop-by-hop architecture:** MACsec protects communication is based on “hop-by-hop” security mode. This security mode is the key for MACsec to protect all messages on the layer 2 and to ensure link protection it is done with different keys for everyone. Packets are decrypted and subsequent encrypted at each MACsec node along their route.

- **Point-to-multipoint security:** The operation of MACsec is grounded in unidirectional point-to-multipoint Security Associations (SAs). Each MACsec

station maintains an SA for outgoing traffic and another SA for incoming traffic from each corresponding MACsec station.

5.3 MACsec protocol operation

MACsec offers secure communication between stations connected to the same LAN, securing data frame by frame through cryptographic techniques within the framework of security relationships managed by MACsec Key Agreement. MACsec is designed to function in networks composed of end stations and various interconnected elements, including point-to-point or shared media LANs and intermediate systems like MAC Bridges, VLAN-aware Bridges, and routers.

Each request has the following four parameters:

- Destination Address
- Source Address
- Priority
- MAC Service Data Unit (MSDU)

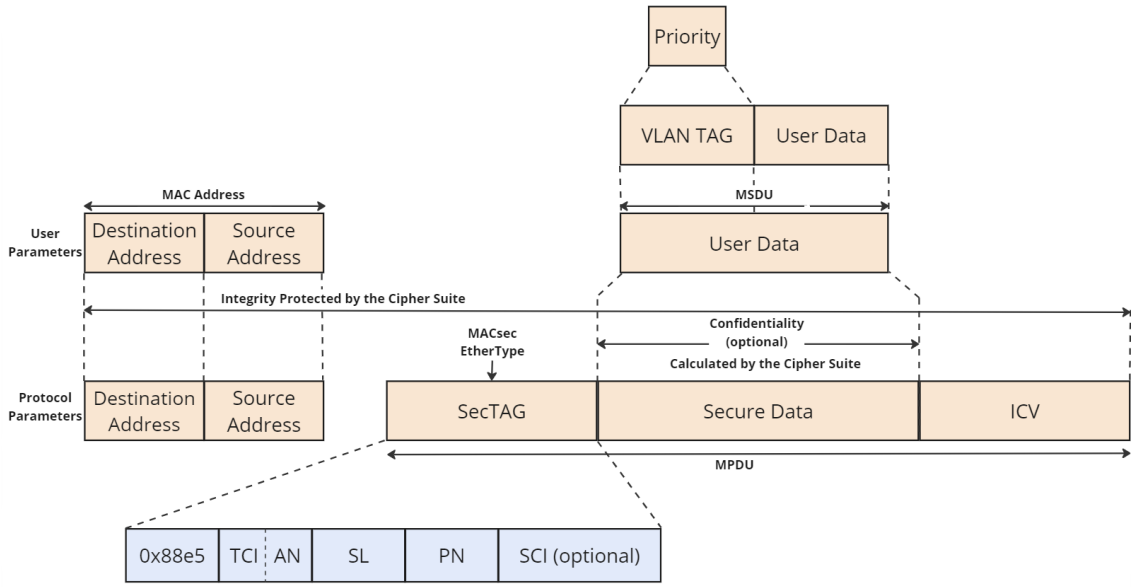


Figure 7. MACsec Frame, VLAN TAG

MACsec operates without the transmission of additional frames, ensuring that each frame is delivered unaltered to peer users.

During transmission, the frame is first assigned to a transmit Security Channel (SC) and a Security Association (SA). To the packet header a parameter is added, the SecTAG. Subsequently, the Integrity Check Value (ICV) is calculated for the entire packet, and optionally, the payload is encrypted simultaneously. Finally, the ICV is attached to the end of the packet before it is transmitted through the network.

Upon receiving the MACsec frame, the SecTAG undergoes validation by examining the packet format. Afterwards, if replay protection is enables, the Packet Number (PN) is checked against the receiver's window making sure the packet number matches what the

receiver is expecting. Finally, the cryptographic signature is verified, and the data is decrypted.

The validation process of the current Cipher Suite involves presenting the SAK, PN, SCI, destination and source addresses of the frame, the SecTAG octets, and the Secure Data and Integrity Check Value (ICV). If the received frame is correct, the replay protection checks that the received PN is not lower than the lowest acceptable PN for the SA. If the check is valid, the frame's parameters, which are identical to those transmitted, are provided to the MACsec client, and the lowest acceptable PN is updated accordingly.

5.4 MACsec Key Agreement (MKA)

MKA handles the creation and maintenance of secure keys for network device authentication and encryption. When two devices, such as switches or routers, need to communicate securely over an Ethernet link, they use MKA to negotiate and exchange keys.

MACsec Key Agreement protocol (MKA) is responsible for discovering, authenticating, and authorizing the potential participants in a CA. It ensures authentication by confirming the mutual possession of a CAK in all the peers that take part in the same CA. It is responsible for deriving other keys form the CAK, the ICK used to protect the integrity and authenticity of the MKA packets, and the KEK used to encrypt the SAK key.

The Key Server selects the Cipher Suite, used to protect the communication within a CA. It is responsible for deriving and securely distributing the Secure Association Key (SAK) used to encrypt the MSDUs. And using the Member Identifier (MI) and the Message Number (MN) parameters, is able to ensure that the frames sent are not delayed.

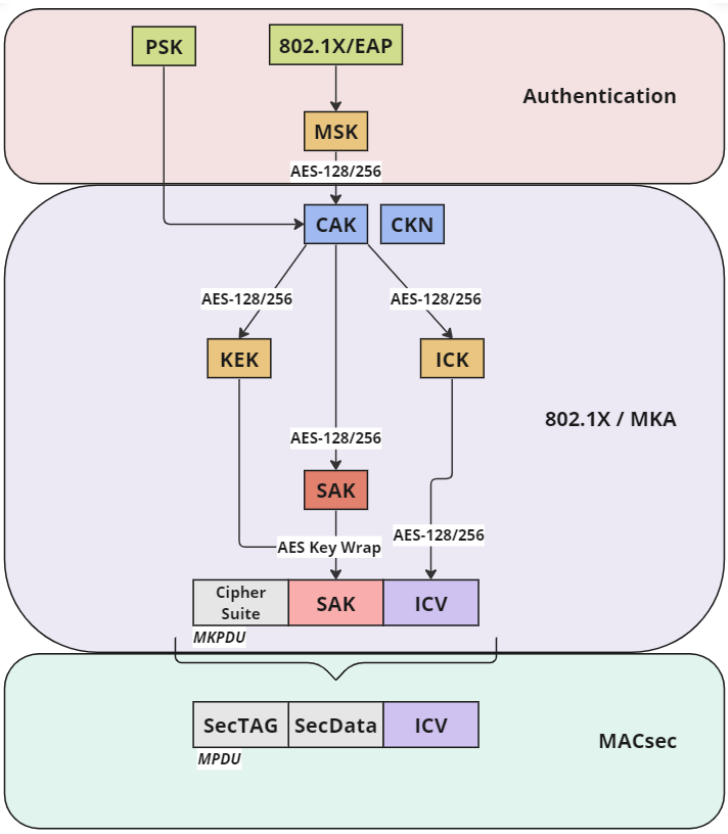


Figure 8. MACsec general phase diagram

5.4.1 Authentication of participants

The MKA Entity (KaY) identifies other participants belonging to its configured CA (Connectivity Association).

The participants of the same MKA communication will exchanged MKPDUs frames to identify and authenticate each other based on the CKN (Connectivity Association Key Name) and the ICV (Integrity Check Value) parameters inside the frame send. All participants of the same CA shared the same CAK. Once a participant identifies and authenticate another participant belonging to the same CA, the member identifier of the other participant is included in its transmitted “Potential Peer List”.

If a participant recognizes another peer and it is listed in the others “Potential Peer List” it will mark it as Live Participant. The Member Identifier (MI) of the other participant will be included in the transmitted “Live Peer List”.

There are two different methods to use, using Pre-Shared Keys or the IEEE 802.1X/EAP.

5.4.1.1 Implementing Pre-Shared Keys (PSK)

First, on a Pre-Shared Key authentication the user introduces manually in each device a pre-shared key, which in essence is the CAK (Connectivity Association Key) with its identifier, the CKN (Connectivity Association Key Name).

The MACsec Key Agreement (MKA) protocol becomes operational once the keys are verified and exchanged successfully. To activate the MKA protocol, it is essential that the pre-shared keys, both the CKN and CAK, match on both ends of the link.

Figure 9. Authentication and CAK generation using Pre-Shared Keys

5.4.1.2 Implementing IEEE 802.1X/EAP

IEEE 802.1X/EAP is used for the authentication, generation, and distribution of a master key MSK (Master Session Key). This master key derives the Connectivity Association Key (CAK). The transmitted frames between the nodes to negotiate the secure channel are encapsulated within the EAPOL protocol.

IEEE 802.1X define several terms related to authentication:

- **Supplicant:** A client device that seeks network access.
- **Authenticator:** A device, such as a switch, that grants or denies network access.
- **Authentication Server (AS):** A server responsible for determining the supplicant's authorization to access the services offered by the authenticator. Additionally, it generates and exchanges cryptographic keys with the supplicant and securely delivers them to the authenticator.

The EAPoL protocol (EAP for LAN) defines the authentication and authorization process.

The authentication and authorization process develops as follows:

1. It begins with the supplicant initiating the authentication process by sending an EAPoL Start message to the Authenticator. The authenticator responds by requesting the supplicant's identity, accomplished through an EAP Request / Identity frame. In response, the supplicant provides an EAP Response, indicating its identity, thereby confirming its authentication request.

2. Subsequently, the Authenticator assumes the role of an intermediary. It communicates with the Supplicant using EAP messages, while engaging with the Authentication Server (AS) through messages that correspond to the authentication protocol employed, such as RADIUS. The AS and the supplicant initially agree on the authentication method (EAP method). Depending on the chosen method, the supplicant provides the appropriate credentials to the AS, which then verifies them through an exchange of EAP Request and Response messages, facilitated by the Authenticator. If the entire process succeeds, the Authenticator concludes with an EAP Success message, granting the supplicant access to the network.

3. Once the supplicant has been authenticated and granted network access, the authentication server and the supplicant reach an agreement on a Master Session Key (MSK). EAP is also employed for this purpose. This cryptographic material can either be derived through mutual consent between the supplicant and the AS or generated by the AS

and distributed to the supplicant. Subsequently, the AS must securely transmit it to the authenticator over a secure channel.

Derived from the MSK, the CAK key serves as a long-lived key from which cryptographic material for MKA and MACsec is generated.

All devices that use this CAK key are considered part of the same CA.

The KDF derivation function, which is used to create the CAK, KEK, and the SAK key in the Key Server, relies on the pseudo-random function AES-CMAC-128/256.

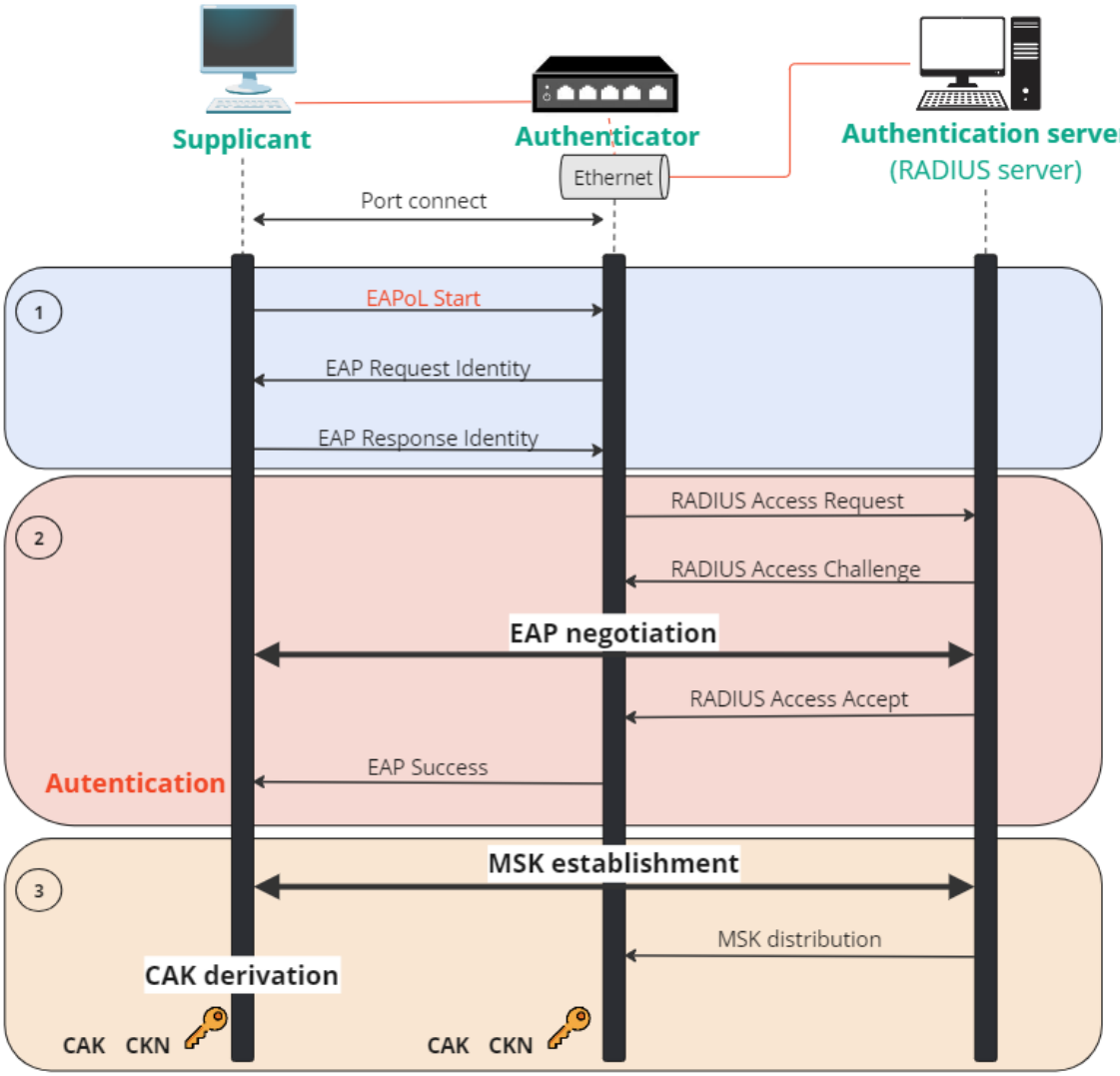


Figure 10. Authentication and CAK generation using 802.1X/EAP

5.4.2 Session negotiation

5.4.2.1 Derivation of ICK and KEK keys

MKA does not directly use the CAK key, instead it uses the CKN. Only the entities that share the same CA have the same CAK. This key is responsible for deriving two different keys, essentials to establish a secure MACsec channel:

- **ICK (Integrity Check Key):** This key is employed in calculating the ICV, which serves to confirm the integrity of MKPDUs packets and authenticate the peers. The

ability to verify authentication is possible thanks to the ICV, which acts as evidence that the node transmitting the message possesses the CAK key.

- **KEK (Key Encryption Key):** The SAK encryption key is transmitted via MKA packets (MKPDUs). To secure the transmission of the SAK key, it is wrapped in AES Key Wrap encryption using the KEK key.

The Key Derivation Function (KDF), used for the generation of the KEK key, use the pseudorandom function AES-CMAC-128 or AES-CMAC-256 depending on the expecting key length of 128 bits or 256 bits key. The KDF use the pseudorandom function AES-CMAC-128 if the CAK has 128 bits or the AES-CMAC-128 if the CAK has 256 bits.

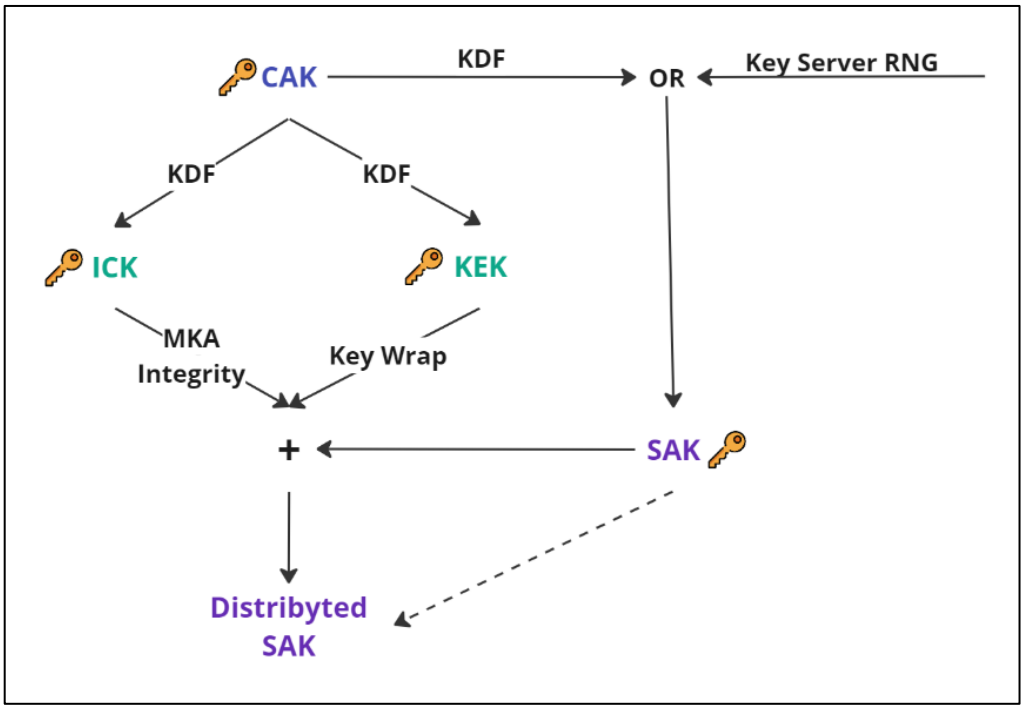


Figure 11. MKA key hierarchy

5.4.2.2 SAK key

It is the secret key derived from the CAK used by a Secure Association (SA). The SAK key, which is distributed over the MKA protocol, serves the dual purpose of authentication (through ICV calculation) and encryption of each packet. Only the entities within the Secure Association (SA) possess this SAK key, and only these entities have the capability to encrypt, decrypt and authenticate packets that are protected by this key.

The SAK is generated by one MKA participant, the Key Server, and distributed during the MKA sequence.

5.4.2.3 Key Server Election

The Key Server is elected dynamically between all the participants within the same CA. Its main duties include generating, from the CAK key, the SAKs keys associated with the SAs and securely distributing them to the relevant members.

The Key Server has several responsibilities:

- It decides when and how to use MACsec, ensuring data confidentiality and integrity.
- It chooses the appropriate chipper suites uses to secure communication.
- It is responsible for generating and distributing the SAK to all the members in the same CA.
- The Key Server assigns unique identifiers to the different communications between nodes, known as Security Associations (SA).

The choice of the key server is made as follows:

1. Each MACsec node transmits a broadcast frame, which includes among other parameters the key server priority and anti-forwarding details. The anti-forwarding details consists of a list of current live peers.
2. Once all the nodes agree on the alive nodes participating in the secure communication, the node with the highest priority is designates as the Key Server.
3. If a node leaves the live peer list, the still alive nodes should select a new Key Server.

When a Key Server cannot be selected, SAKs are not distributed.

5.4.2.4 SAK generation, distribution, and selection

As explained in the last section, the Key Server's primary responsibility lies in generating and distributing MACsec Secure Association Keys (SAKs) to other CA members using the MKA transport, achieved through AES Key Wrap encryption.

Each SAK key is unique and is allocated a 128-bit Key Identifier (KI), which comprises:

- A 96-bit Key Server identifier (Message Identifier, MI).
- A 32-bit number (KN – Key Name) assigned to the SAK key by the Key Server. This sequential assignment begins with the number 1.

These KIs serve as references for SAI assignment and are openly visible in MKPDUs.

Once a Key Server generates a SAK, it must include it in every MKPDU it transmits through its principal actor. This continues until all live peers capable of using the chosen Cipher Suite report that they have installed the SAK for reception. Alternatively, if there's a change in the CA's live membership, a new SAK is needed, then the distribution process will stop.

The security provided by each SAK rest on the security provided by the Cipher Suite, and both are used for encryption and integrity protection.

Importantly, once a Key Server has distributed a SAK, it should never include any previously generated SAKs in subsequent MKPDUs.

5.4.2.5 Cipher Suite selection

The Key Server plays a crucial role in selecting the cipher suite and announcing it alongside each SAK key and it is used to protect the frames.

There are four possible cipher suites: GCM-AES-128, GCM-AES-256, GCM-AES-XPN-128 and GCM-AES-XPN-256 and they are detailed in the Cipher Suites section.

This process involves distributing the identification code of the cipher suite (8 octets) and the "confidentiality offset," which can be set to 0, 30, or 50. These values indicate that the first 0, 30, or 50 octets of the MACsec packet (MSDU) remain unencrypted, carrying only integrity protection.

An MKA Key Server can also elect not to use MACsec to secure communication, using MKA simply to prove prior authentication.

Any participant possessing the CAK, from which the KEK is derived, has the capability to decrypt the packet. Within the MKA packet (MKPDU), the Key Server also includes the cipher suite selected for that specific SAK, along with a unique number assigned to that SAK by the Key Server (KN).

Subsequently, the Key Server will distribute the SAK key among MACsec nodes, encrypting it using AES Key Wrap and the KEK key, within the MKPDU packets sent to all nodes in the network. This distribution continues until the nodes respond to the message, confirming the successful installation of the SAK key.

SAK keys are regenerated under the following circumstances:

- Exceeding the key usage limit, which depends on the cipher suite utilized. More detailed information can be found in the section Cipher Suites.
- Addition or removal of a new node from the CA.
- Modification of the cipher suite to be employed.
- Introduction of a timeout for SAK key regeneration.

The Key Server holds the responsibility for generating a fresh SAK key and distributing it to all CA members.

5.4.3 MKA transport

As mentioned earlier, MKA provides secure transportation between participants in the same CA. Every transmitted packet (MKPDU) is authenticated using the CAK key, which is known to all CA members. Possession of the CAK certifies the transmitter as an authorized CA member.

Each transmitter member includes in the MKPDU packet:

- The CKN, the identifier of the CAK.
- Their SCI (Secure Channel Identifier), composed by the MAC direction and the port.
- Their MI (Member identifier), a random number of 96 bits that each member chooses when starts its participation in the protocol.
- Their MN (Message Number), a 32 bits sequential number, starting by 1, assigned to the MKPDU packets. It increments by one in each MKPDU transmitted.
- The ICV (Integrity Check Value) calculated from the ICK key, which is derived previously by the CAK.

Including the CKN the receptor will be able to confirm the ICV using the correct CAK. This verification shows that the sender was in possession of the CAK, allowing the MKPDU packets to be authenticated.

On the other hand, using the MI and MN values together allows transmission to be protected from delas or replay attacks.

5.4.4 Secure MACsec session

Once the nodes or MACsec stations have correctly installed the SAK key, all transmitted traffic will be encrypted using the cipher suites selected by the Key Server, with the corresponding SAK keys for each Security Association (SA). SAK keys are refreshed when the number of packets transmitted with a particular key exceeds a certain threshold.

In the secure session, MACsec-encrypted traffic is allowed to flow, except for control traffic such as EAPoL frames.

The secure session employs a unique identifier known as Secure Channel Identifier (SCI), and during the session, only packets containing the SCI identifier for that specific session are accepted.

The secure session also offers anti-replay protection using the PN (Packet Number) and two parameters:

- **Replay Protection:** It shows if the anti-replay protection is enabled or not.
- **Replay Window:** It shows the window of time within which an out of order packet may be accepted.

For example, if the replay window is 300, it indicates that if the receives packet has a PN = 5, the next packet accepted may have a PN between 6 and 306. A zero replay Window serves as anti-forwarding protection, this is the default value indicated in the IEEE 802.1AE.

6. MACsec Protocol Data Units (MPDUs) frame structure

This specifies the structure and coding of the MACsec Protocol Data Units (MPDUs) that are exchanged between MACsec Entities (SecYs).

The Secure Data is the data in the frame that is encrypted using MACsec. Each MPDU consist of an integer number of octets, starting with one and increasing in the same order in which they are inserted into the MAC Service Data Unit (MSDU).

An octet refers to a sequence of 8 bits, numbered from 1 to 8 in increasing order of bit significance.

At the link layer (Layer 2 of the OSI model), when communicating without MACsec, the LAN network encapsulates packets as follows:

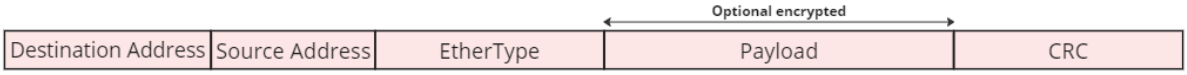


Figure 12. Ethernet packet

MACsec adds to the Ethernet frames the SecTAG and the Integrity Check Value (ICV), and encryption of the data is optional. The Secure Data is never of zero length, because MAC service requires a non-null MSDU parameter.

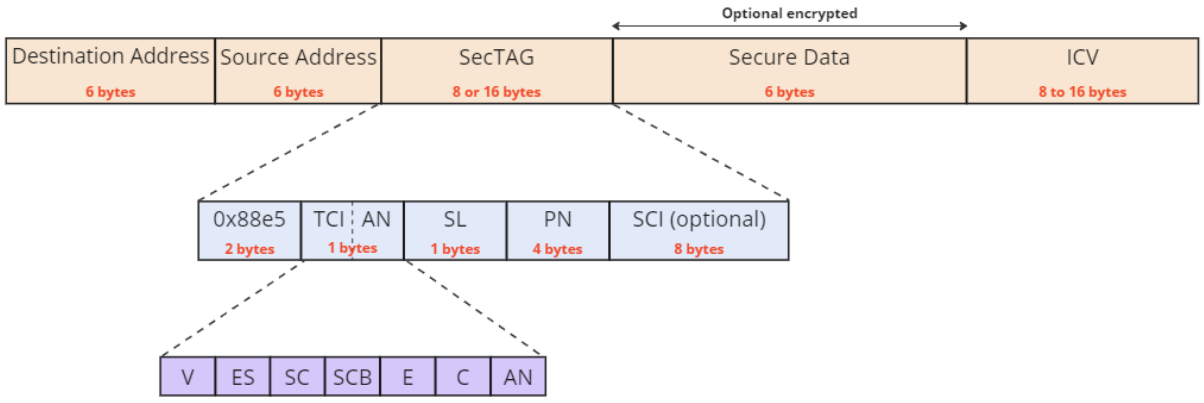


Figure 13. MACsec frame

6.1.1 SecTAG header

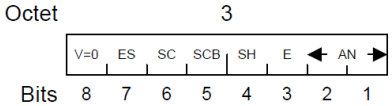
Name		Bytes	Description
Security TAG (SecTAG)	MACsec EtherType	2 octets	Ethernet Type of MACsec frames, its value is 0x88e5.
	TCI	6 bits	TAG Control Information indicates how the frame is protected. For details, see the next table.  *Figure 14. TCI*
	AN	2 bits	Association Number identifies up to four different SAs within the context of SC. It identifies the SAK and the next PN of the SA.
	SL	6 bits	Short Length value set the length of the encrypted data (MSDU) and indicates the number of octets of the User Data. If the Secure Data (user data) is less than 48 bytes, padding is added. Now the SL is the length in bytes of the Secure data, otherwise is set to 0. (Padding is before ICV) It is 1 byte.
	PN	4 octets	Packet Number serves as a unique identifier for packets exchanged between MACsec peers. The PN is a continuously increasing number assigned to each MACsec frame sent using a single SAK key. This number is always greater than zero and gets incremented each time a protect request is generated for a specific SCI. If the cipher suite supports XPN, then only the lower 32bit are encoded in the PN.
	SCI	8 octets	Secure Channel Identifier is optional and only present when the SC bit is set. It is designated in each port to make the communication safer. Comprises the Sender's MAC address (48 bits) and the Port Identifier (2 bytes, as an integer). IMPORTANT! → FF-FF-FF-FF-FF-FF-FF-FF is never used as an SCI.
Integrity Check Value (ICV)		16 octets	This field is responsible for verifying the integrity of the transmitted data, including the MAC addresses (both the source and destination), and ensuring that they have not been altered or corrupted during transfer. Its length depends on the cipher suite used, it will never be less than 8 octets, nor more than 16 octets.

Table 1. SecTAG header

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>V</td><td>Version number. This value it is set to 0 by the standard.</td></tr>
<tr><td>ES</td><td>End Station identifier. It indicates if the sender is an end station. This happen when the System Identifier bytes of the SCI match the MAC source address. If the source MAC address is not used to determine the SCI, the ES bit is cleared. If the ES bit is set, the SC bit should not be set.</td></tr>
<tr><td>SC</td><td>Secure Channel. It is set to one if the SCI is explicitly encoded.</td></tr>
<tr><td>SCB</td><td>Single Copy Broadcast.
The Ethernet Passive Optical Network (EPON) broadcast flag is activated only when the MPDU is associated with a Service Channel (SC) that supports the EPON Single Copy Broadcast (SCB) capability. If the SCB bit is set, the SC bit should remain unset.</td></tr>
<tr><td>E</td><td>Encryption flag. It is set if data is encrypted, confidentiality is being provided.</td></tr>
<tr><td>C</td><td>Flag for changed text. If the C bit is set, the User Data is modified. As such, this bit indicates whether the User Data is the same as the Secure Data.
It works in conjunction with E (encryption):

<table>
<tr><th>E</th><th>C</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>Integrity-only is provided, the User Data is not encrypted (the Secure Data bytes match the User Data bytes), and the ICV is 16 bytes.</td></tr>
<tr><td>1</td><td>0</td><td>The frame is not processed by the Secure Entity (SecY) but is reserved for use by the KaY. It is not protected with MKA. Not protected -> MKA</td></tr>
<tr><td>0</td><td>1</td><td>Indicates that integrity-only is provided. The integrity check algorithm modifies the User Data.</td></tr>
<tr><td>1</td><td>1</td><td>Indicates that the packet data is encrypted. As the C bit is set, the User data is modified.</td></tr>
</table>
Table 2. Encryption flag & Changed text flag</td></tr>
</table>

Table 3. TAG Control Information (TCI) header

7. Cipher Suites

MACsec protocol uses an algorithm for packet encryption and integrity protection through what is called the cryptographic suite or cipher suite.

A cipher suite encrypts traffic on a MACsec-secured link, employing cryptographic algorithms and functions with parameters to ensure packet encryption and integrity protection within the MACsec protocol. The parameters used are the following ones:

- The encryption algorithm: Cipher
- The key length: Key Length
- The operation mode: Mode

The cipher suite, generally, is represented as follows: Mode-Cipher-Key Length.

In block cipher operations, the use of an Initialization Vector (IV) is required, and it must be unique for each packet encrypted with the SAK key. To compose the IV, the PN parameter is used of 32 bits. This means that they will be 2^{32} uses allowed for a single SAK key, which might cause problems on high-speed networks (40 Gb/s) because the PN expires quickly and new SAK keys must be generated.

Every time Ethernet is getting faster and therefore more frequent key regeneration is required. In the automotive industry, this is an inconvenience that needs to be avoided. XPN (Extended PN) was introduced to support higher speed links and is highly important for automotive use cases. Thus, to reduce the frequency of SAK refresh, you can select cipher suites like GCM-AES-XPN-128 and GCM-AES-XPN-256, which expand the packet number to 64 bits. This new PN length would take several years for the PN to be exhausted, ensuring the SAK key regeneration does not occur as frequently on high-speed links.

It is crucial to note that XPN can only be used with devices that have a throughput of 40 Gb/s or more.

There are four cipher suites accepted in the current version of the MACsec standard: GCM-AES-128, GCM-AES-256, GCM-AES-XPN-128, and GCM-AES-XPN-256. These cipher suites all make use of a 32-bit Packet Number (PN). When all the available packet numbers are exhausted, it's necessary to refresh the SAK.

It's crucial for MACsec peers to employ the same cipher suite to ensure compatibility. In cases where a specific MACsec cipher suite isn't configured, the default cipher suite used is GCM-AES-128.

Cipher Suite ID	Cipher Suite Name
00-80-C2-00-01-00-00-01	GCM–AES–128
00-80-C2-00-01-00-00-02	GCM-AES-256
00-80-C2-00-01-00-00-03	GCM–AES–XPN-128
00-80-C2-00-01-00-00-04	GCM-AES-XPN-256

Table 4. Cipher Suites

Integrity protection and encryption are provided by the SAKkey an the Cipher Suite.

All these Cipher Suites support “Integrity with Confidentiality” mode and “Integrity without Confidentiality”, but the XPN Cipher Suites do not support confidentiality offsets.

MACsec uses two ciphers:

- Secure forwarding via SAK key encryption. Using the MKA protocol and the AES Key Wrap function, which is performed by the key server.
- Using the SAK key for data encryption (MSDU). One of the indicated Cipher Suite is used for this encryption.

As specified in the IEEE 802.1AE-2018 standard, in MACsec the cipher is optional.

The ICV (Integrity Check Value) is calculated and added to the MACsec packet in order to ensure its integrity. The ICV calculation function is mandatory. The data (encrypted or not), the SecTAG and the source and destination MAC addresses are all protected by the ICV.

8. Crypto overview

Cryptography is the study and implementation of secure communication methods in the presence of adversaries, who are malicious actors seeking to intercept or modify information. Its primary goal is to develop and evaluate policies that safeguard information security principles by preventing unauthorized access to shared data. Through cryptography, communication can only be read and processed by the intended recipient. This practice has been in use for many years and remains essential in today's technology, including e-commerce, bank cards, and computer passwords. Modern cryptographic approaches use 128-bit and 256-bit encryption keys, along with sophisticated algorithms and ciphers, such as the nearly impenetrable Advanced Encryption Standard (AES). Cryptography combines computer science, engineering, and mathematics to create complex codes that ensure the secrecy and integrity of data.

The core principles of modern cryptography are data confidentiality, data integrity, authentication, and non-repudiation. Cryptography is crucial for protecting data and users, ensuring confidentiality, and preventing cyber criminals from intercepting sensitive information.

8.1 Symmetric encryption

Symmetric encryption is a method that uses the same key for both encryption and decryption processes. This approach is particularly efficient for encrypting large volumes of data.

The Advanced Encryption Standard (AES) is an example of symmetric encryption, offering support for key sizes of 128, 192, and 256 bits while operating on 128-bit data blocks. Its widespread adoption is attributed to its robust security features and versatility.

There are different modes of operation for symmetric encryption, including:

- **Electronic Codebook (ECB)** mode independently encrypts each data block, but its susceptibility to pattern-based attacks necessitates careful consideration.
- **Cipher Block Chaining (CBC)** mode enhances security by incorporating an Initialization Vector (IV) and interlinking data blocks, mitigating vulnerabilities associated with ECB mode.
- **Cipher Feedback (CFB)** and **Output Feedback (OFB)** modes are used to convert block ciphers into stream ciphers, providing a flexible approach to data encryption.
- **Counter (CTR)** mode transforms block ciphers into stream ciphers by encrypting counter values, offering a high degree of parallelism and efficiency in encryption operations.

8.2 AES-GCM (Galois/Counter Mode)

AES-GCM (Galois/Counter Mode) is an advanced and efficient mode of operation for the AES encryption algorithm that offers both confidentiality and authentication with a high level of security.

Confidentiality is achieved by encrypting the plaintext using a counter mode, similar to AES-CTR, which ensures that the original data remains private and inaccessible to unauthorized parties.

Authentication plays a crucial role in ensuring the integrity and authenticity of the data being transmitted. It accomplishes this by using a cryptographic hash function, and GCM

further enhances this process by combining encryption with Galois field multiplication for robust authentication.

The inputs and outputs for AES-GCM encompass various components:

- **Secret Key:** This key is fundamental for both encryption and decryption processes, safeguarding the confidentiality of the data.
- **Initialization Vector (IV):** A 12-byte value that ensures the uniqueness of each encryption operation, even when using the same secret key.
- **Plaintext:** This refers to the original data that needs to be encrypted while maintaining its confidentiality.
- **Additional Authentication Data (AAD):** This optional data is authenticated but not encrypted, providing an additional layer of security.
- **Output:** The output consists of ciphertext and an authentication tag (ICV), both of which are essential for ensuring the integrity and authenticity of the data.

The encryption process involves encrypting the plaintext with AES in counter mode to ensure confidentiality. Simultaneously, the ciphertext and AAD are authenticated to produce an authentication tag (ICV), which serves as a crucial indicator of the data's integrity and authenticity.

Upon receiving the encrypted data, the receiver can decrypt the ciphertext using the secret key and then verify the ICV to ensure that the data has not been tampered with and remains authentic. This meticulous process ensures that the data remains secure and unaltered throughout its transmission.

8.3 Additional Cryptographic Concepts

8.3.1 Public Key Cryptography (Asymmetric Encryption)

Public Key Cryptography, also known as Asymmetric Encryption, is a cryptographic system that uses a pair of keys - a public key and a private key. The public key is used to encrypt data, while the private key is used to decrypt it. This technology is widely used for secure key exchange, digital signatures, and various other cryptographic applications.

8.3.2 Hash Functions

Hash functions are algorithms that convert input data into a fixed-size hash value, often represented as a string of numbers and letters. Common algorithms for hash functions include SHA-256 and MD5, although MD5 is now considered insecure due to vulnerabilities. Hash functions are used for data integrity checks and digital signatures in cryptographic systems.

8.3.3 Digital Signatures

Digital signatures provide authentication and non-repudiation for digital messages or documents. They are created by encrypting a hash of the data using a private key, and the recipient can verify the digital signature using the corresponding public key. This ensures the integrity and authenticity of the signed data.

8.3.4 Key Management

Key management is an essential aspect of maintaining the security of cryptographic systems. It involves the entire lifecycle of cryptographic keys, including key generation, secure distribution, key storage, and key revocation. Effective key management practices are crucial for ensuring the confidentiality and integrity of sensitive data.

9. Testing environment setup

This section contains the practical part done in the project. The first chapter describes the setup used to develop the two testcases done. In the second chapter, the first MKA implementation test case with Pre-Shared Keys (PSK) is explained, along with its configuration and the analysis of the transmitted frames. Moving on to the third chapter, the MKA implementation with the Extensible Authentication Protocol (EAP) is presented, including its requirements for implementation and the results obtained. Finally, the conclusions are shown in the final chapter of this section.

Firstly, analyzing the behavior of MACsec and MKA is essential to ensure that they work as expected. This point-to-point communication will be used in a real test case; therefore, it is important to test it thoroughly and be confident about its correct implementation and the results.

9.1 Setup development

To ensure the correct implementation of MKA (MACsec Key Agreement) and obtain the expected results, a point-to-point communication between two virtual machines has been set up. This test bench is only for testing purposes.

The test bench set up involves using a laptop with Oracle VM Virtual Machine installed. Two Ubuntu 22.04 machines have been created, with Wireshark installed on each, and configured them with MKA.

- **Wireshark** is an application used for analyzing each TCP/IP segment that is transmitted over a network. This application provides an important advantage for troubleshooting segments since it can modify each layer of the TCP/IP architecture independently.
- **Oracle VM Virtual Machine** is a windows application that can run on virtual machines.

To distinguish between the two Linux Virtual Machines, a unique identifier has been assigned to each. One of them is identified as Alice, and the other is identified as Bob. From this point forward, they will be referred to as Alice and Bob, respectively.

9.2 MACsec configuration with PSK

For this initial setup, MKA has been configured in both VMs using PSK.

9.2.1 Linux MKA implementation

To configure these Linux virtual machines, it is used the wpa_supplicant.conf file. This configuration file is used to define network settings.

This tool is versatile and primarily used for managing wireless connections and ensuring their security. However, wpa_supplicant can also handle wired connections, which is particularly useful for networks using 802.1X.

When implementing MACsec, security is enhanced by using the MACsec Key Agreement protocol (MKA), which is an extension defined in IEEE 802.1X-2010. MKA sets up Secure Channels, Associations, and key exchanges between network nodes. Once this setup is done, MACsec protects the data traffic at the link layer.

Wpa_supplicant runs in the background, ensuring that network connections stay secure, even as encryption keys are updated regularly. It manages 802.1X port-based authentication, where the switch port remains inactive until the supplicant is authenticated.

To successfully configure MKA, let's proceed with explaining the different steps to follow:

- **Step 1:**

The initial step involves generating two important components: the CAK key, which consists of 16 hexadecimal bytes, and the CKN parameter, which consists of 32 hexadecimal bytes. This can be accomplished by using a random key generator. The required command for this operation is as follows:

```
16 bytes: dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2
"%02x"'

32 bytes: dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2
"%02x"'
```

- **Step 2:**

After defining the CAK and CKN values, the next step is to create a configuration file that will be used by the tool, specifying the required settings. The user has the flexibility to choose the filename, and for this example, it is used "wpa_supplicant.conf."

The file structure is as follows:

Wpa_supplicant.conf

```
ctrl_interface=/var/run/wpa_supplicant
eapol_version=3
ap_scan=0
fast_reauth=1

network={
key_mgmt=NONE
eapol_flags=0
macsec_policy=1
mka_cak=cfbdb470315147b385194516490bf490
mka_ckn=1b6bec17cd5024a6ed65675ce61965c0b8f474233ed04e5dfedb964bae1e80b
c
}
```

The parameters listed are as follows:

- **ctrl_interface:** This specifies the path to the control interface location, with the default value being /var/run/wpa_supplicant.
- **eapol_version:** This parameter denotes the version of the IEEE802.1X standard to be used. Since MKA was introduced in the latest version, it is referenced with the value 3.
- **ap_scan:** For wired Ethernet connections, this should be set to 0.
- **fast_reauth:** This variable is used to disable fast EAP reauthentication.

- **key_mgmt:** It indicates which protocol is employed for key establishment and management. In this case, the parameter is set to NONE, as pre-shared keys will be utilized. If IEEE 802.1X is to be used, specify the IEEE8021X parameter.
- **eapol_flags:** When conducting wired authentication, this should be configured as 0 for proper authentication.
- **macsec_policy:** Set this to 1 to enable the MACsec protocol. By default, it is deactivated.
- **mka_cak:** This refers to the pre-shared CAK key created earlier.
- **mka_ckn:** This corresponds to the pre-shared CKN value created previously.

- **Step 3:**

After creating the configuration file, the final step is to activate the connection. This is done with the following command:

```
sudo wpa_supplicant -i enp0s3 -Dmacsec_linux -c wpa_supplicant.conf
```

The command parameters are as follows:

- **i:** Specifies the network interface for creating the MACsec device.
- **-Dmacsec_linux:** Indicates the usage of the MACsec driver for Linux.
- **-c:** Specifies the configuration file to be utilized.
- **-B:** This parameter can be added to run the protocol as a background daemon process.

- **Step 4:**

After executing the command mentioned above, the connection will be initiated, and a new MACsec interface named "macsec0" will have been successfully created.

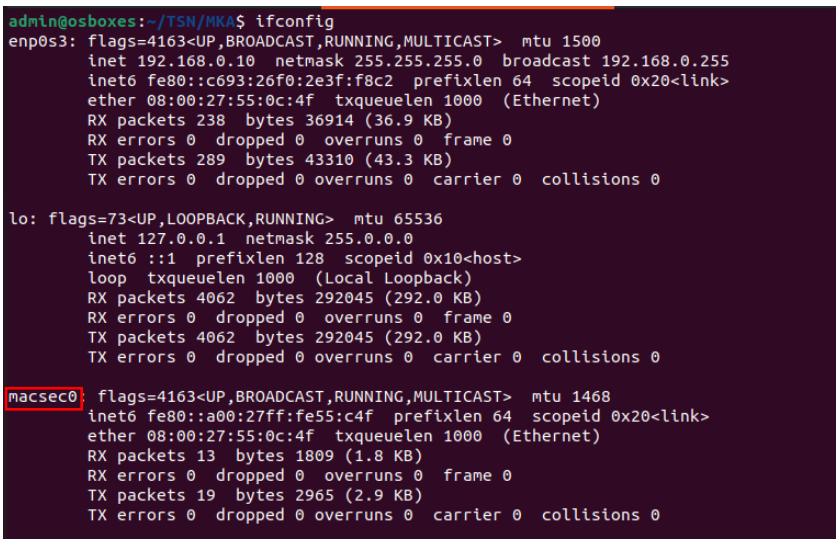


Figure 15. MACsec interface created

- **Step 5:**

In addition, the MACsec protocol is enabled with the command:

```
ip macsec show
```

```
admin@osboxes:~/TSN/MKA$ ip macsec show
4: macsec0: protect on validate strict sc off sa off encrypt on send_sci on end_station off scb off replay off
    cipher suite: GCM-AES-128, using ICV length 16
    TXSC: 080027550c4f0001 on SA 0
        0: PN 21, state on, key 0a40e43fc06352172550da3f01000000
    RXSC: 080027caa02d0001, state on
        0: PN 26, state on, key 0a40e43fc06352172550da3f01000000
    offload: off
```

Figure 16. MACsec protocol enabled

This command provides essential information, including the active cipher suite, encryption status, PN values for both transmission and reception channels, and the activation of anti-replay protection. Furthermore, it shows the presence of two distinct secure channels, each equipped with its SAK key and PN value: the TXSC, responsible for data transmission, and the RXSC, dedicated to data reception.

Additionally, it allows the determination of the packet direction on each secure channel. For the transmit channel, the address corresponds to the MAC address of Alice, whereas on the receive channel, it is associated with the MAC address of Bob.

- **Step 6:**

To ensure correct functionality of the MACsec channels, Bob's machine must be configured with the same keys and configurations.

- **Step 7:**

Next, a simple test will be performed to verify the operation of the protocol. To achieve this, IP addresses will be assigned to each "macsec0" interface on the devices, initiate a ping, and capture the resulting data using a protocol analyzer, Wireshark.

```
sudo ifconfig macsec0 10.0.0.10   #Alice’s machine

sudo ifconfig macsec0 10.0.0.20   #Bob’s machine

ping 10.0.0.20   #Ping performed form Alice’s machine to Bob’s machine
```

a) From the analyzer, the MACsec packets are captured:

In comparison to a normal LAN, the Ethernet frame format in a MACsec-protected network contains MACsec EtherType 0x88e5 which means traffic is protected. It can be observed that the MACsec-protected frame size is 130 bytes. This is longer than a normal network frame since it includes an additional 32 bytes of SecTag and ICV.

No.	Time	Source	Destination	Protocol	Length	Info
121	43.580801504	PcsCompu_ca:a0:2d	PcsCompu_55:0c:4f	MACSEC	130	MACsec frame
122	44.032471607	PcsCompu_55:0c:4f	Nearest-non-TPMR-br…	EAPOL-MKA	162	Key Server, Live Peer List, MACsec SAK Use
123	44.124843050	PcsCompu_ca:a0:2d	IPv4mcast_fb	MACSEC	119	MACsec frame
124	44.582840182	PcsCompu_55:0c:4f	PcsCompu_ca:a0:2d	MACSEC	130	MACsec frame
125	44.582995769	PcsCompu_ca:a0:2d	PcsCompu_55:0c:4f	MACSEC	130	MACsec frame
126	44.723128016	PcsCompu_ca:a0:2d	Nearest-non-TPMR-br…	EAPOL-MKA	162	Live Peer List, MACsec SAK Use
127	45.584319036	PcsCompu_55:0c:4f	PcsCompu_ca:a0:2d	MACSEC	130	MACsec frame
128	45.584375153	PcsCompu_ca:a0:2d	PcsCompu_55:0c:4f	MACSEC	130	MACsec frame
129	46.034489711	PcsCompu_55:0c:4f	Nearest-non-TPMR-br…	EAPOL-MKA	162	Key Server, Live Peer List, MACsec SAK Use
130	46.588480039	PcsCompu_55:0c:4f	PcsCompu_ca:a0:2d	MACSEC	130	MACsec frame
131	46.588527827	PcsCompu_ca:a0:2d	PcsCompu_55:0c:4f	MACSEC	130	MACsec frame
132	46.723721945	PcsCompu_ca:a0:2d	Nearest-non-TPMR-br…	EAPOL-MKA	162	Live Peer List, MACsec SAK Use
133	47.612167885	PcsCompu_55:0c:4f	PcsCompu_ca:a0:2d	MACSEC	130	MACsec frame
134	47.612211934	PcsCompu_ca:a0:2d	PcsCompu_55:0c:4f	MACSEC	130	MACsec frame

Figure 17. MACsec frame length

In the figure below, the packet is of type MACsec (802.1AE). Besides, inside the Security TAG header of the packet, the different parameters of the protocol are visible. Among these parameters, the E parameter is set, indicating that encryption is enabled, as configured before in the terminal. The C parameter is also set. When these two parameters are set means that confidentiality is provided, and the User Data is encrypted.

```
▸ Frame 51: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
▾ Ethernet II, Src: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d), Dst: PcsCompu_55:0c:4f (08:00:27:55:0c:4f)
  ▸ Destination: PcsCompu_55:0c:4f (08:00:27:55:0c:4f)
  ▸ Source: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d)
    Type: 802.1AE (MACsec) (0x88e5)
▾ 802.1AE Security tag
  ▾ 0010 11.. = TCI: 0x0b, VER: 0x0, SC, E, C
      0... .... = VER: 0x0
      .0.. .... = ES: Not set
      ..1. .... = SC: Set
      ...0 .... = SCB: Not set
      .... 1... = E: Set
      .... .1.. = C: Set
    .... ..00 = AN: 0x0
    Short length: 30
    Packet number: 46
    System Identifier: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d)
    Port Identifier: 1
    ICV: f2fc13a8fbf77058781acd1ea1067851
▸ Data (30 bytes)
```

Figure 18. MACsec frame parameters

9.2.2 MKA protocol frames analysis

This frequency diagram summarizes all the frames exchanged between Alice and Bob, two virtual machines. In the upcoming sections, the communication between both nodes will be explained in detail until the MKA protocol successfully enabled MACsec between them.

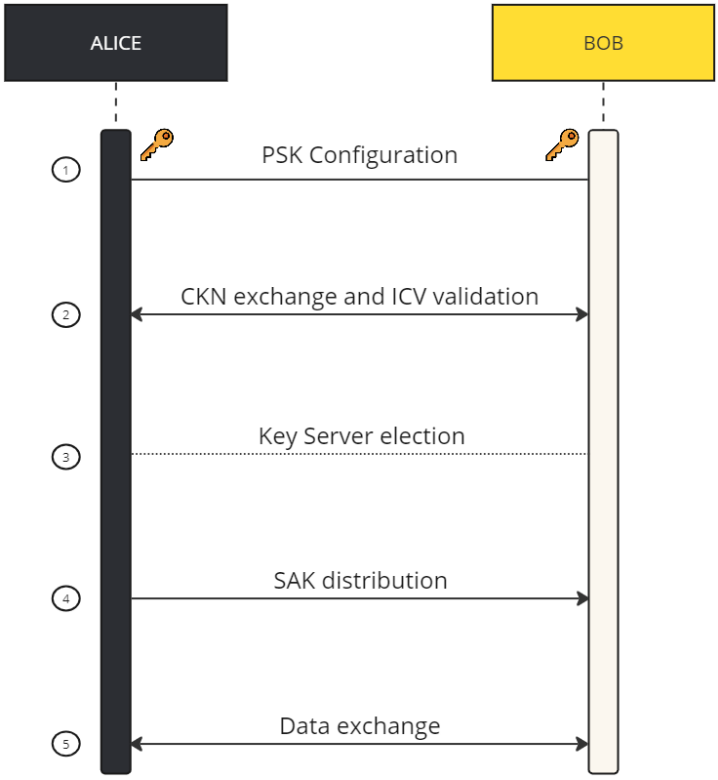


Figure 19. MKA Encryption Process

Here's an overview of the frames present in the exchange. Frames 1 through 7 are all part of the MACsec Key Agreement protocol (MKA). Frame 13 and some others contain the same information and serve as MACsec keepalives.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_55:0c:4f	Nearest-non-TPMR-bridge	EAPOL-…	98	Key Server
2	2.000611390	PcsCompu_55:0c:4f	Nearest-non-TPMR-bridge	EAPOL-…	98	Key Server
3	2.701614215	PcsCompu_ca:a0:2d	Nearest-non-TPMR-bridge	EAPOL-…	118	Key Server, Potential Peer List
4	4.002139633	PcsCompu_55:0c:4f	Nearest-non-TPMR-bridge	EAPOL-…	150	Key Server, Live Peer List, Distributed SAK
5	4.010608944	PcsCompu_ca:a0:2d	Nearest-non-TPMR-bridge	EAPOL-…	162	Live Peer List, MACsec SAK Use
6	4.021988928	PcsCompu_55:0c:4f	Nearest-non-TPMR-bridge	EAPOL-…	194	Key Server, Live Peer List, MACsec SAK Use, Distributed SAK
7	4.025628633	PcsCompu_ca:a0:2d	Nearest-non-TPMR-bridge	EAPOL-…	162	Live Peer List, MACsec SAK Use
8	4.031826908	PcsCompu_55:0c:4f	IPv6mcast_16	MACSEC	122	MACsec frame
9	4.036648718	PcsCompu_ca:a0:2d	IPv6mcast_16	MACSEC	122	MACsec frame
10	4.444419729	PcsCompu_55:0c:4f	IPv6mcast_16	MACSEC	122	MACsec frame
11	4.492713632	PcsCompu_ca:a0:2d	IPv6mcast_16	MACSEC	122	MACsec frame
12	4.524692199	PcsCompu_ca:a0:2d	IPv6mcast_ff:ca:a0:2d	MACSEC	118	MACsec frame
13	4.702019201	PcsCompu_ca:a0:2d	Nearest-non-TPMR-bridge	EAPOL-…	162	Live Peer List, MACsec SAK Use
14	4.828903828	PcsCompu_55:0c:4f	IPv6mcast_ff:55:0c:4f	MACSEC	118	MACsec frame

Figure 20. MKA frames overview

9.2.3 MKPDU analysis

This analysis is based on the 802.1X-2020 standard, and Wireshark was used to analyze the frames.

9.2.3.1 Frame 1 (Alice – Bob)

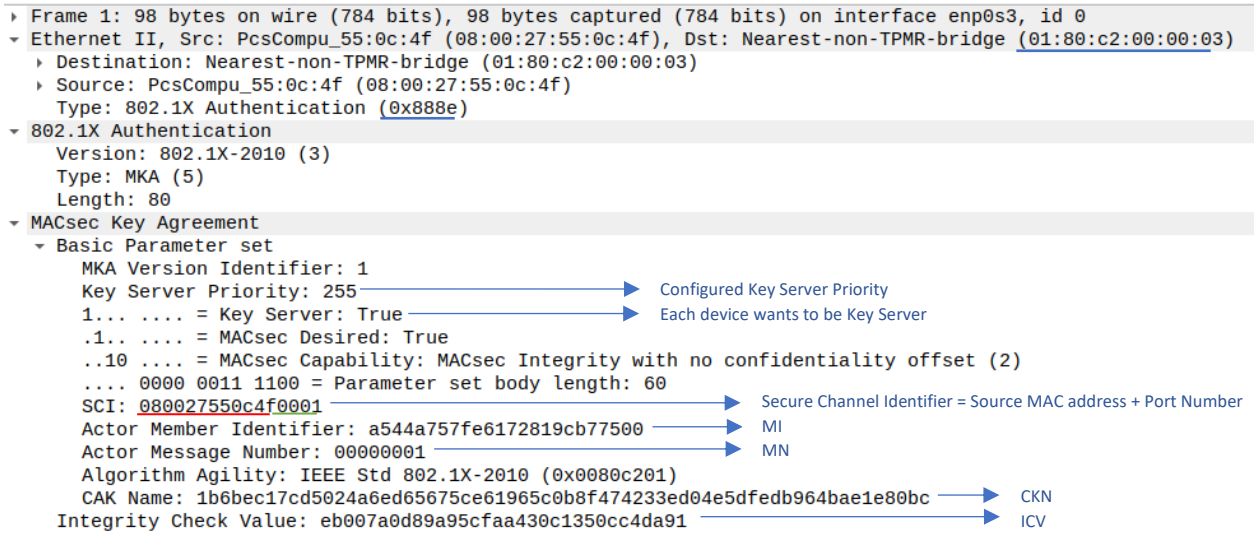


Figure 21. Frame 1

The MACsec protocol begins with Alice (one of the nodes) sending an MKPDU frame to Bob (the other node) to establish a link between them. Alice has been configured for MACsec, which means that the first frame transmitted after the link is established is the EAPoL frame. Once the link is established, Alice and Bob become peers.

In the MKPDU frame, Alice's MAC address is set as the source, and the destination MAC address is the well-known MKA multicast address, represented as 01:80:c2:00:00:03. The frame type is identified as an MKA frame, indicated by the hexadecimal value 0x888e.

Within the **Basic Parameter set** several parameters are included:

- **MKA Version Identifier** → It indicates the version we are currently working on, with three distinct versions available.
- **Key Server Priority** → The server priority, in this frame, is fixed at 0xFF, equivalent to 255.
- **Key Server** → The Key Server flag is set to TRUE, which serves as the default value before electing a key server.
- **MACsec Desired** → It is set to TRUE, expressing the participant's intention to employ MACsec for protecting Ethernet frames.
- **MACsec Capability** → The MACsec capability parameter holds a value of 2, indicating that MACsec encryption will be enabled, with no confidentiality offset specified to indicate where encryption begins.
  This are the different values it could take:
  - 0 if MACsec is not implemented,
  - 1 if 'Integrity without confidentiality',
  - 2 if 'Integrity without confidentiality' and 'Integrity and confidentiality' with a confidentiality offset of 0,
  - 3 if 'Integrity without confidentiality' and 'Integrity and confidentiality' with a confidentiality offset of 0, 30, or 50.

- **Parameter set body length** → This two-octet field encodes an unsigned binary number, specifying the length in octets of the Packet Body field; a value of 0 indicates the absence of a Packet Body. For instance, an MKPDU with a body length below 40 octets would be discarded, as seen in the case of a Basic Parameter Set with a body length of 17.
- **SCI** → An SCI (Secure Channel Identifier) is encoded in a fixed length field of eight octets.
- **Actor MI** → MKA uses a random 96-bit member identifier (MI) to identify each node in the CA domain.
- **Actor MN** → The Actor Message Number (32-bit) increases sequentially in subsequent MKPDUs, starting from 1, to protect against delayed transmission or replay.
- **Algorithm Agility** → The Algorithm Agility Value, 0x0080c201, determines the algorithm to use for the derivation of ICK form CAK, in this case should be calculated following the IEEE 802.1X-2010 standard.
  CKN → Every MKPDU includes a Secure Connectivity Association Key Name (CKN), enabling the intended recipients to recognize the CAK and, consequently, the ICK for the verification of the ICV.
  In our implementation, the CAK is pre-shared and, therefore, hard-coded.
- **ICV** → Within the MKPDU, the presence of ICV is optional; nevertheless, it becomes mandatory when traffic is encrypted. This value authentication is confirmed using the CAK and it is generated by the AES-CMAC using the ICK.
  As the ICK is not directly distributed by any protocol, only derived from the CAK, verification of the ICV serves the dual purpose of ensuring the integrity of the MKPDU contents and verifying that it was constructed by a system possessing the CAK.

Upon receiving the frame, Bob first checks the CKN. In this case, the CKN is a match. Next, he verifies the Integrity Check Value (ICV) using the ICK. This verification ensures that the sender also has the same CAK, thus authenticating network peers. The ICV also ensures the integrity of each MKPDU. If the verification is successful, Bob proceeds with the protocol. If not, the frame is discarded. In this instance, the ICV is identical in both Alice's and Bob's frames, confirming the successful verification of the ICV. This indicates that the frame remained unaltered during communication, ensuring its integrity.

Since the source address is included in the sent frame, it confirms the authenticity of the data source. Moreover, it demonstrates that both virtual machines (VMs) share the same CAK, indicating their membership in the same Connectivity Association (CA). The VM then uses the corresponding CAK to generate the ICK and KEK, which cannot be seen in the EAPOL frame.

There are a couple of important points to keep in mind regarding EAPoL frames. Firstly, each new frame increases the Member Number by one. Secondly, the SCI is made up of the source MAC address and the Port Number, which helps maintain security requirements such as integrity and authenticity using the ICV. Confidentiality is provided through the encryption of the SAK using KEK or by encrypting the data using SAK.

Additionally, replay protection is ensured through the use of the Message Number (MN) in MKA and the Packet Number (PN) in MACsec. These measures help prevent any unauthorized access to the data being transmitted.

9.2.3.2 Frame 2 (Alice – Bob)

This second frame is identical to the first frame, except for the ICV, Actor MI, and Actor MN values. The Actor MN value increments in each frame and is set to 2 in this frame. The participants are waiting for Bob to establish the connection, and once Bob is connected, they won't send this frame anymore.

When a receiver authenticates an MKPDU with a specific MI and MN, any subsequent MKPDUs with the same MI and an equal or lower MN value are ignored. When the MN value reaches its maximum, the participant can choose a new MI and start the MN again, resetting it to 1.

```
▸ Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_55:0c:4f (08:00:27:55:0c:4f), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)
▾ 802.1X Authentication
    Version: 802.1X-2010 (3)
    Type: MKA (5)
    Length: 80
▾ MACsec Key Agreement
  ▾ Basic Parameter set
      MKA Version Identifier: 1
      Key Server Priority: 255
      1... .... = Key Server: True
      .1.. .... = MACsec Desired: True
      ..10 .... = MACsec Capability: MACsec Integrity with no confidentiality offset (2)
      .... 0000 0011 1100 = Parameter set body length: 60
      SCI: 080027550c4f0001
      Actor Member Identifier: a544a757fe6172819cb77500
      Actor Message Number: 00000002
      Algorithm Agility: IEEE Std 802.1X-2010 (0x0080c201)
      CAK Name: 1b6bec17cd5024a6ed65675ce61965c0b8f474233ed04e5dfedb964bae1e80bc
    Integrity Check Value: 732c2631976ffe452f8b773f4066ec50
```

Figure 22. Frame 2

9.2.3.3 Frame 3 (Bob – Alice)

The following definitions will help to understand the terms used in the context of Live Peer List, and Potential Peer List:

The **Live Peer List** includes all the participants who have included the MI and a recent MN in an MKPDU, indicating which participants are currently in possession of the CAK.

The **Potential Peer List** includes all other peers who have transmitted an MKPDU that has been directly received by the participant or that were included in the Live Peer List of an MKPDU transmitted by a peer that has proved liveness.

In the presence of one or more Potential Peers, the Member Identifier and Message Number pairs of these Potential Peers are organized within a Potential Peer List.

Participants are removed from each list if an interval between the MKA Life Time and MKA Life Time plus MKA Hello Time has passed since the transmission of the participant’s most recent Message Number (MN). This duration ensures that enough time has passed for the potential loss or delay of two or more MKPDUs before mistakenly removing a live peer.

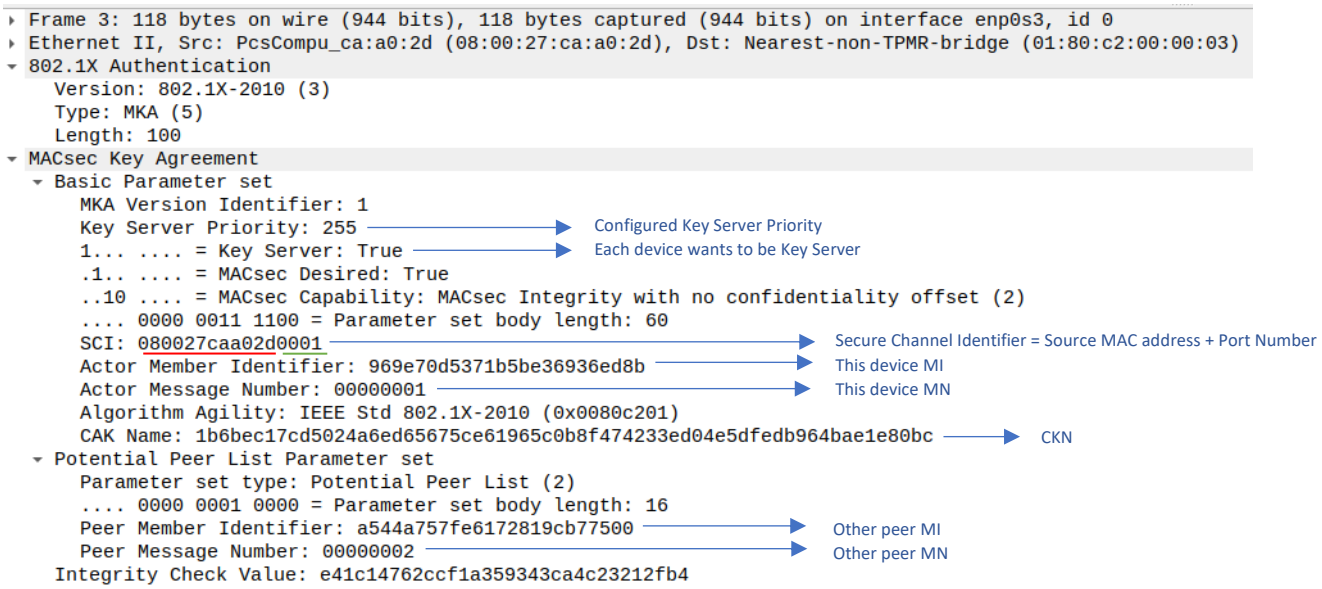


Figure 23. Frame 3

At this point, both switches are aware of each other's Key Server priority.

Bob's **Basic Parameter set** includes several parameters such as Key Server Priority, which is fixed at 0xFF, equivalent to 255. Then includes the Key Server flag and the MACsec desired flag, both set to TRUE, as well as the MACsec capability parameter set to 2. It also includes the SCI and Actor Member ID values. The Algorithm Agility parameter represents the IEEE 802.1X-2010 standard, and the CAK name remains the same as the one transmitted by Alice. Each participant proves liveness, the current possession of the CAK, and their active participation in the protocol by including their Member Identifier (MI) along with a suitably recent Message Number (MN) within an MKPDU.

Within the **Potential Peer List Parameter set** has a body length is 16 bytes. Alice's MI is the Peer Member Identifier, which is the same one sent in the last frames, and the Peer Message Number value is the number of frames sent by Alice. As Alice has sent two frames, the MN values are set to two.

The parameter set type is 1 for the live peer list and 2 for the potential peer list.

Upon receiving the frame, Alice identifies Bob as an active peer and initiates the process of selecting the key server. In this context, both Alice and Bob have a key server priority of 0xFF. This value is typically used to indicate that a device does not want to act as the key server, but its behavior may vary based on the implementation. According to the MKA standard, the lower numerical value indicates higher priority. However, when a tie occurs, the SCIs of their transmit channels are compared to elect the key server, and the live peer with the lowest Secure Channel Identifier (SCI) becomes the elected key server. The Key Server is responsible for distributing the Secure Association Keys (SAKs) required for MACsec.

As a result, Alice is elected as the key server. Therefore, when Bob sends the MKPDU with the basic parameter set, the key server flag is set to FALSE. However, in subsequent frames sent from Alice to Bob, the key server flag will be set to TRUE. After this frame, the two devices can transmit data across a secure link.

9.2.3.4 Frame 4 (Alice – Bob)

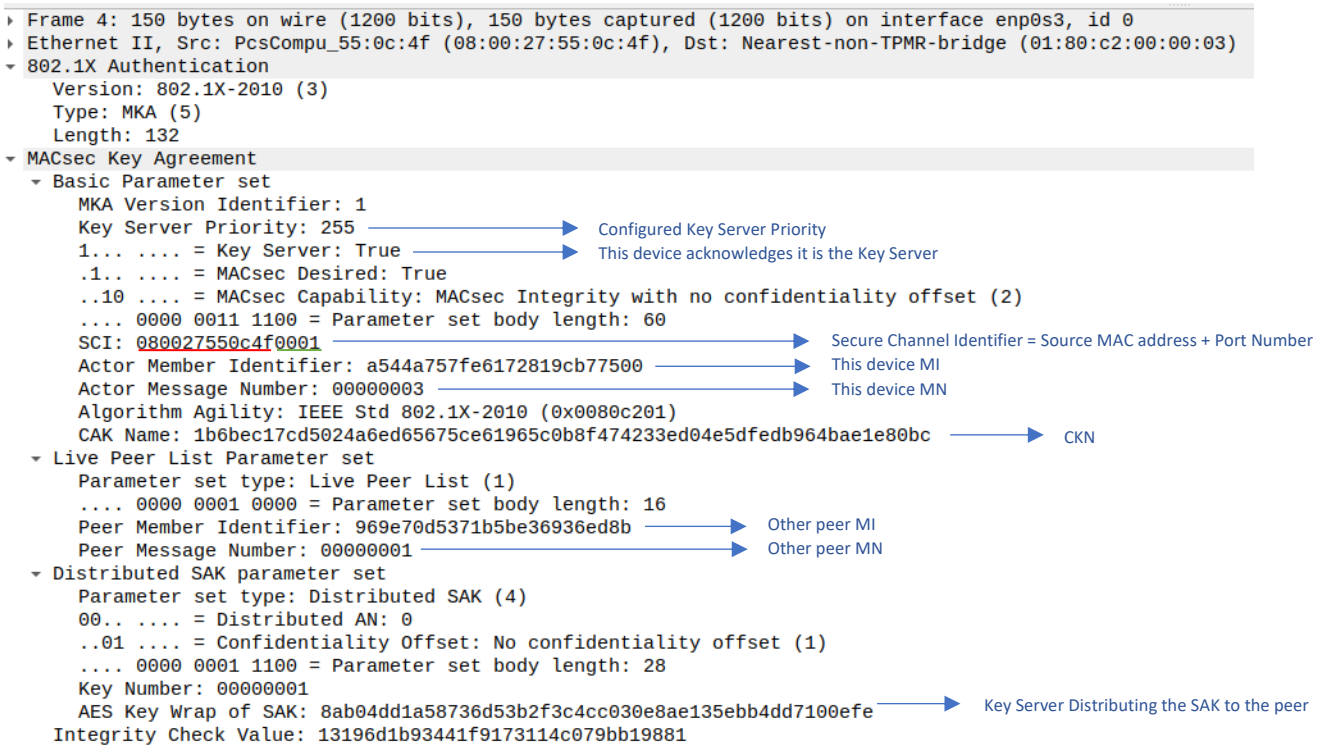


Figure 24. Frame 4

Alice has been chosen as the key server, which means she is responsible of generating and distributing the SAK. Before distributing it, Alice encrypts the SAK using an AES Key Wrap function so that the encryption key is not sent in clear text. The KEK generated at Alice's node is used to encrypt the SAK. However, if a Key Server cannot be selected, SAKs are not distributed.

In the **Basic Parameter set**, when Alice becomes become the Key Server, the Key Server is set to TRUE, and the Actor Message Number increments its value to 3.

The **Live Peer List Parameter set** includes both Bob's member identifier (MI) and the most recent message number (MN).

Alice sends the encrypted Secure Association Key (SAK) to Bob as part of the **Distributed SAK Parameter set** in an MKPDU. Below are the parameters that can be observed in this set:

- **Parameter set type** → This value is always 4 for the Distributed Parameter set. However, it may change depending on the Parameter set used in a MKPDU.
- **Distributed AN** → The AN (Association Number) identifies the SAK used in the MKA instance, and the SA used. This AN is assigned sequentially by the Key Server for every distributed key, it begins form zero. MKA uses the same AN for al Security Associations (SAs) that are related to a particular SAK. If the Key Server decides not to use MACsec, the value of this parameter is set to zero.
- **Confidentiality Offset** → It determines where the encryption starts in a frame. In this case, there is no confidentiality offset.
- **Parameter set body length** → It specifies the number of bytes in the parameter set, except for the header. If plain text transmission is used instead of MACsec, the length

is set to 0. If GCM-AES-128 is used, the length is set to 28. If the Cipher Suite reference number is explicitly included, the length is set to 36 or greater.

- **Key Number** → The key number is assigned sequentially, starting from 1 and identifies the SAK.
- **AES Key Wrap of SAK** → In the context of AES key wrapping, the 32 bytes represent the encrypted SAK. Our implementation uses a 256-bit SAK and KEK. The AES key wrap process involves encrypting the SAK using both the KEK (Key Encryption Key) and the GCM-AES-128 Algorithm.

At the end of the frame, an Integrity Check Value (ICV) is included.

Upon receiving this MKPDU, Bob decodes the frame to obtain the encrypted SAK. He first verifies the Integrity Check Value (ICV). After a successful verification, he proceeds to decrypt the SAK using the Key Encryption Key (KEK).

9.2.3.5 Frame 5 (Bob – Alice)

After receiving the Secure Association Key (SAK), Bob sets the MACsec protected reception channel to TRUE, which allows the decryption of encrypted messages using the SAK. It's important to note that the SAK can only encrypt and decrypt data packets when both the SAK receiving and SAK transmitting are enabled on both ends. However, Bob needs to wait for the key server to enable its transmission channel before he sets his transmission channel to TRUE, in accordance with the IEEE standard 802.1X-2020.

The peers who have received the SAK will find a MACsec SAK Use parameter set in their MKPDUs, and its parameter set type is 3.

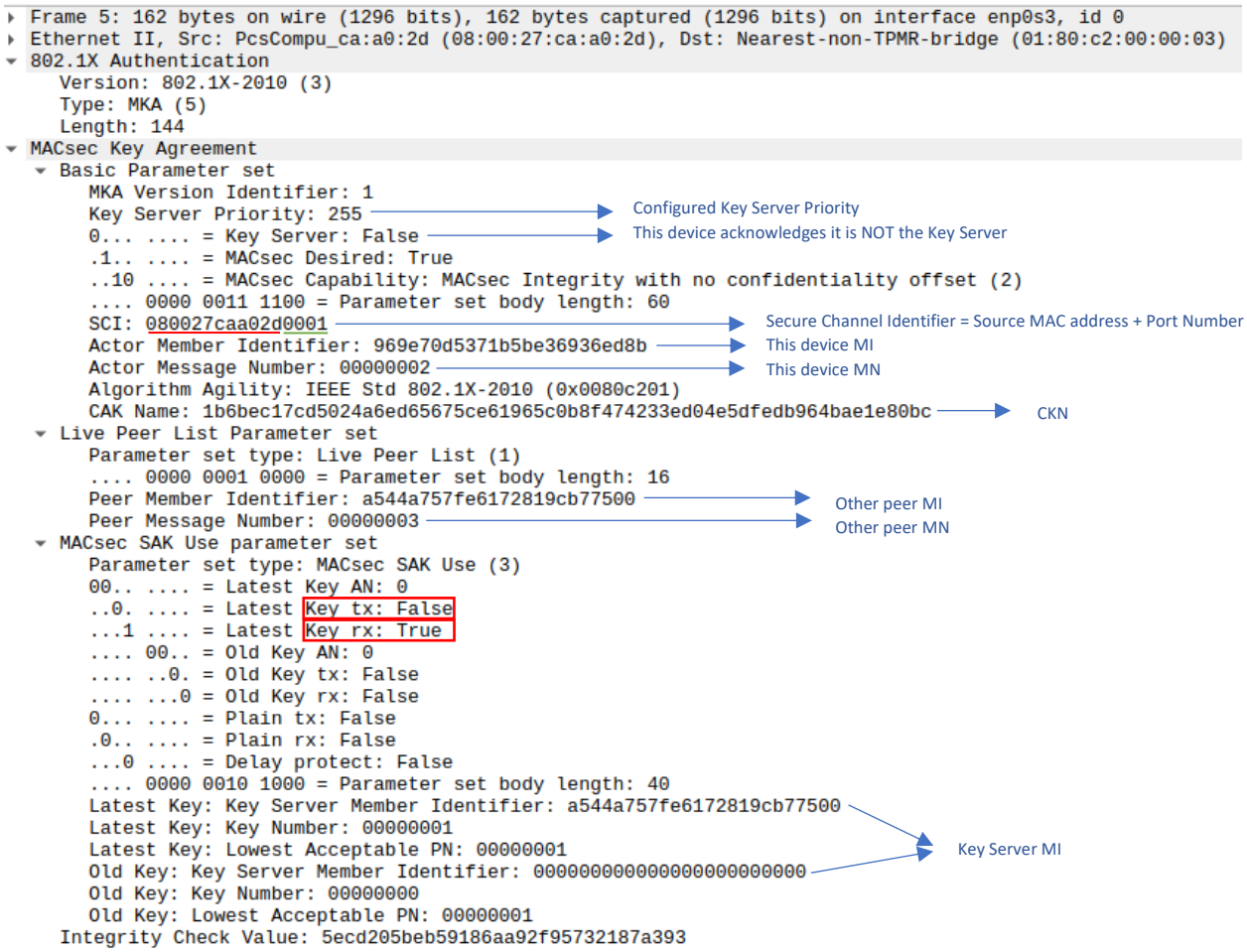


Figure 25. Frame 5

In line with the figure, when Bob transmits the MKPDU to Alice, the **MACsec SAK use parameter set** indicates that the latest transmitted key (Tx) is FALSE, while the latest received key (Rx) is TRUE.

All parameters for the old key are set to FALSE, since this is the first SAK generated in this MKA instance. The AN (Association Number) of the latest and old SAK identifies the SA. The plain transmission (Tx) and reception (Rx) options are also set to FALSE, since the transmission and reception channels require MACsec protection to ensure data is not transmitted in plain form. In this scenario, delay protection is set to FALSE, since our proof of concept does not provide this functionality. If it is set to TRUE, the network peer rejects the next packet received after a specified time.

The parameter set body length is set to 0 if MACsec is not supported, otherwise it is set to 40. The values of the Key Server's Member Identifier and Key Number are derived from the data present in the last frame sent by the Key Server, Alice node. The Lowest Acceptable Packet Number (PN) specifies the PN that is acceptable for that SAK within a specified period.

9.2.3.6 Frame 6 (Alice – Bob)

```
▸ Frame 6: 194 bytes on wire (1552 bits), 194 bytes captured (1552 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_55:0c:4f (08:00:27:55:0c:4f), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)
▾ 802.1X Authentication
    Version: 802.1X-2010 (3)
    Type: MKA (5)
    Length: 176
▾ MACsec Key Agreement
  ▾ Basic Parameter set
      MKA Version Identifier: 1
      Key Server Priority: 255
      1... .... = Key Server: True
      .1.. .... = MACsec Desired: True
      ..10 .... = MACsec Capability: MACsec Integrity with no confidentiality offset (2)
      .... 0000 0011 1100 = Parameter set body length: 60
      SCI: 080027550c4f0001
      Actor Member Identifier: a544a757fe6172819cb77500
      Actor Message Number: 00000004
      Algorithm Agility: IEEE Std 802.1X-2010 (0x0080c201)
      CAK Name: 1b6bec17cd5024a6ed65675ce61965c0b8f474233ed04e5dfedb964bae1e80bc
  ▾ Live Peer List Parameter set
      Parameter set type: Live Peer List (1)
      .... 0000 0001 0000 = Parameter set body length: 16
      Peer Member Identifier: 969e70d5371b5be36936ed8b
      Peer Message Number: 00000001
  ▾ MACsec SAK Use parameter set
      Parameter set type: MACsec SAK Use (3)
      00.. .... = Latest Key AN: 0
      ..1. .... = Latest Key tx: True
      ...1 .... = Latest Key rx: True
      .... 00.. = Old Key AN: 0
      .... ..0. = Old Key tx: False
      .... ...0 = Old Key rx: False
      0... .... = Plain tx: False
      .0.. .... = Plain rx: False
      ...0 .... = Delay protect: False
      .... 0000 0010 1000 = Parameter set body length: 40
      Latest Key: Key Server Member Identifier: a544a757fe6172819cb77500
      Latest Key: Key Number: 00000001
      Latest Key: Lowest Acceptable PN: 00000001
      Old Key: Key Server Member Identifier: 00000000000000000000000000
      Old Key: Key Number: 00000000
      Old Key: Lowest Acceptable PN: 00000001
  ▾ Distributed SAK parameter set
      Parameter set type: Distributed SAK (4)
      00.. .... = Distributed AN: 0
      ..01 .... = Confidentiality Offset: No confidentiality offset (1)
      .... 0000 0001 1100 = Parameter set body length: 28
      Key Number: 00000001
      AES Key Wrap of SAK: 8ab04dd1a58736d53b2f3c4cc030e8ae135ebb4dd7100efe
    Integrity Check Value: 6754e6f5362e967e5d41d9efee296e6b
```

Figure 26. Frame 6

Upon receiving frame 5, Alice sets its transmission channel to TRUE. This is in accordance with the IEEE standard, which states that the key server should wait for all live peers to activate their reception channels before enabling its transmission channel. In this case, Bob is the only live peer for Alice. Therefore, upon receiving the MKPDU with Rx as TRUE in the SAK use parameter set, Alice activates its own Tx channel to TRUE.

9.2.3.7 Frame 7 (Bob – Alice)

After receiving frame 6, Bob activates its most recent key Tx channel to TRUE since the key server has set its Tx channel to TRUE as well. As shown in the figure illustrating the MKPDU, when both Alice and Bob have their latest key Tx and Rx channels set to TRUE, it allows MACsec protected communication for this specific MKA instance.

```
▸ Frame 7: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)
▾ 802.1X Authentication
    Version: 802.1X-2010 (3)
    Type: MKA (5)
    Length: 144
▾ MACsec Key Agreement
  ▾ Basic Parameter set
      MKA Version Identifier: 1
      Key Server Priority: 255
      0... .... = Key Server: False
      .1.. .... = MACsec Desired: True
      ..10 .... = MACsec Capability: MACsec Integrity with no confidentiality offset (2)
      .... 0000 0011 1100 = Parameter set body length: 60
      SCI: 080027caa02d0001
      Actor Member Identifier: 969e70d5371b5be36936ed8b
      Actor Message Number: 00000003
      Algorithm Agility: IEEE Std 802.1X-2010 (0x0080c201)
      CAK Name: 1b6bec17cd5024a6ed65675ce61965c0b8f474233ed04e5dfedb964bae1e80bc
  ▾ Live Peer List Parameter set
      Parameter set type: Live Peer List (1)
      .... 0000 0001 0000 = Parameter set body length: 16
      Peer Member Identifier: a544a757fe6172819cb77500
      Peer Message Number: 00000003
  ▾ MACsec SAK Use parameter set
      Parameter set type: MACsec SAK Use (3)
      00.. .... = Latest Key AN: 0
      ..1. .... = Latest Key tx: True
      ...1 .... = Latest Key rx: True
      .... 00.. = Old Key AN: 0
      .... ..0. = Old Key tx: False
      .... ...0 = Old Key rx: False
      0... .... = Plain tx: False
      .0.. .... = Plain rx: False
      ...0 .... = Delay protect: False
      .... 0000 0010 1000 = Parameter set body length: 40
      Latest Key: Key Server Member Identifier: a544a757fe6172819cb77500
      Latest Key: Key Number: 00000001
      Latest Key: Lowest Acceptable PN: 00000001
      Old Key: Key Server Member Identifier: 00000000000000000000000000
      Old Key: Key Number: 00000000
      Old Key: Lowest Acceptable PN: 00000001
    Integrity Check Value: bd11e222165c7ae8d502c2122ee88cab
```

Figure 27. Frame 7

9.2.3.8 Frame 8 (Bob – Alice)

After completing the MKA protocol, both Alice and Bob can send and receive MACsec protected frames. The frame described is the first one transmitted form Bob to Alice.

```
▸ Frame 119: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
▾ Ethernet II, Src: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d), Dst: PcsCompu_55:0c:4f (08:00:27:55:0c:4f)
  ▸ Destination: PcsCompu_55:0c:4f (08:00:27:55:0c:4f)
  ▸ Source: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d)
    Type: 802.1AE (MACsec) (0x88e5)
▾ 802.1AE Security tag
  ▾ 0010 11.. = TCI: 0x0b, VER: 0x0, SC, E, C
      0... .... = VER: 0x0
      .0.. .... = ES: Not set
      ..1. .... = SC: Set
      ...0 .... = SCB: Not set
      .... 1... = E: Set
      .... .1.. = C: Set
    .... ..00 = AN: 0x0
    Short length: 30
    Packet number: 35
    System Identifier: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d)
    Port Identifier: 1
    ICV: f4ca03e4764b7d00d7bebb62d1b9bc8b
▾ Data (30 bytes)
    Data: f557a017a1ca3503c92baabf43ccc03340a373fd78f96b6b60bfc89e7b3d
    [Length: 30]
```

Figure 28. Frame 8

Bob initiates the process by encrypting the plain data and calculating the ICV. This forms the MPDU, which is then sent, as illustrated in the figure.

Bob's address is used as the source, while Alice's address serves as the destination. The MACsec EtherType, which indicates secure traffic, is identified as 0x88e5, the standard MACsec frame type.

Additionally, the frame includes a Security tag that holds information about the security features of the frame for the recipient. The first parameter within the Security tag is the TAG Control Information, which provides information about the MACsec protocol version and whether 'integrity only' or 'encryption only' modes are being used, among other things. In this case, the encryption is enabled for confidentiality. The short length denotes the length of the secure data field, which is 30 bytes in this example. The system identifier and the port are used to identify the MKA instance, while the AN (Association Number) is used to identify the SAK for decryption when Alice receives the MPDU. Finally, the encrypted data is followed by the ICV. Upon reception, Alice first verifies the ICV, and if the verification is successful, decryption takes place.

9.2.3.9 Frame 9 (Alice – Bob)

When Alice transmits a frame to Bob, Bob encrypts the data and sends it as an MPDU (MACsec Protocol Data Unit). Once Bob receives the MPDU, he verifies the Integrity Check Value (ICV) and then decrypts the data. The diagram illustrates this MPDU process.

```
▸ Frame 120: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface enp0s3, id 0
▾ Ethernet II, Src: PcsCompu_55:0c:4f (08:00:27:55:0c:4f), Dst: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d)
  ▸ Destination: PcsCompu_ca:a0:2d (08:00:27:ca:a0:2d)
  ▸ Source: PcsCompu_55:0c:4f (08:00:27:55:0c:4f)
    Type: 802.1AE (MACsec) (0x88e5)
▾ 802.1AE Security tag
  ▾ 0010 11.. = TCI: 0x0b, VER: 0x0, SC, E, C
      0... .... = VER: 0x0
      .0.. .... = ES: Not set
      ..1. .... = SC: Set
      ...0 .... = SCB: Not set
      .... 1... = E: Set
      .... .1.. = C: Set
    .... ..00 = AN: 0x0
    Short length: 0
    Packet number: 38
    System Identifier: PcsCompu_55:0c:4f (08:00:27:55:0c:4f)
    Port Identifier: 1
    ICV: 67fae5a63f540c0f51335bdecc413fee
▾ Data (86 bytes)
    Data: ee0893f15bed641a6d84db6bed311f2190b3c96661c78e4f1b8db5f64db7f0e51398c21e…
    [Length: 86]
```

Figure 29. Frame 9

9.2.3.10 Keepalive frames

When MACsec protected communication is turned on, participants such as Alice and Bob send a specific frame to let each other know that they are still active. This is called a keepalive frame. It contains three parameter sets: the Basic Parameter set, the Live Peer List Parameter set, and the MACsec SAK Use Parameter set.

If Alice is the Key Server peer, her Basic Parameter set will have the Key Server flag set as TRUE. However, in Bob's Basic Parameter set, the Key Server flag will be set as FALSE. You can see these parameter configurations in the associated image.

```
▸ Frame 111: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_55:0c:4f (08:00:27:55:0c:4f), Dst: Nearest-non-TPMR-bridge (01:80:c2:00:00:03)
▸ 802.1X Authentication
▾ MACsec Key Agreement
  ▾ Basic Parameter set
      MKA Version Identifier: 1
      Key Server Priority: 255
      1... .... = Key Server: True
      .1.. .... = MACsec Desired: True
      ..10 .... = MACsec Capability: MACsec Integrity with no confidentiality offset (2)
      .... 0000 0011 1100 = Parameter set body length: 60
      SCI: 080027550c4f0001
      Actor Member Identifier: a544a757fe6172819cb77500
      Actor Message Number: 00000016
      Algorithm Agility: IEEE Std 802.1X-2010 (0x0080c201)
      CAK Name: 1b6bec17cd5024a6ed65675ce61965c0b8f474233ed04e5dfedb964bae1e80bc
  ▸ Live Peer List Parameter set
  ▸ MACsec SAK Use parameter set
    Integrity Check Value: 33be6377fa745aa99bb442e9d8891bfa
```

Figure 30. Keep alive frame

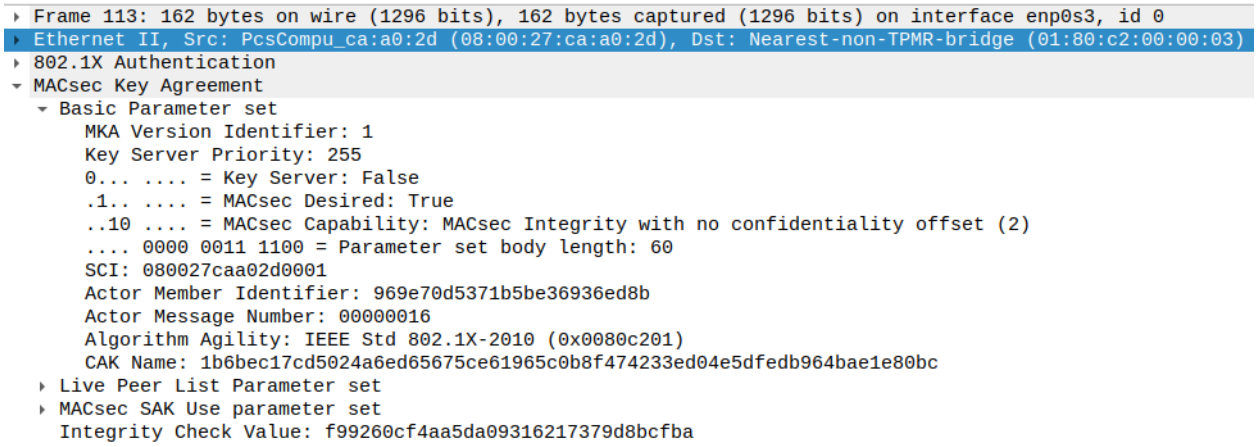


Figure 31. Keep alive frame

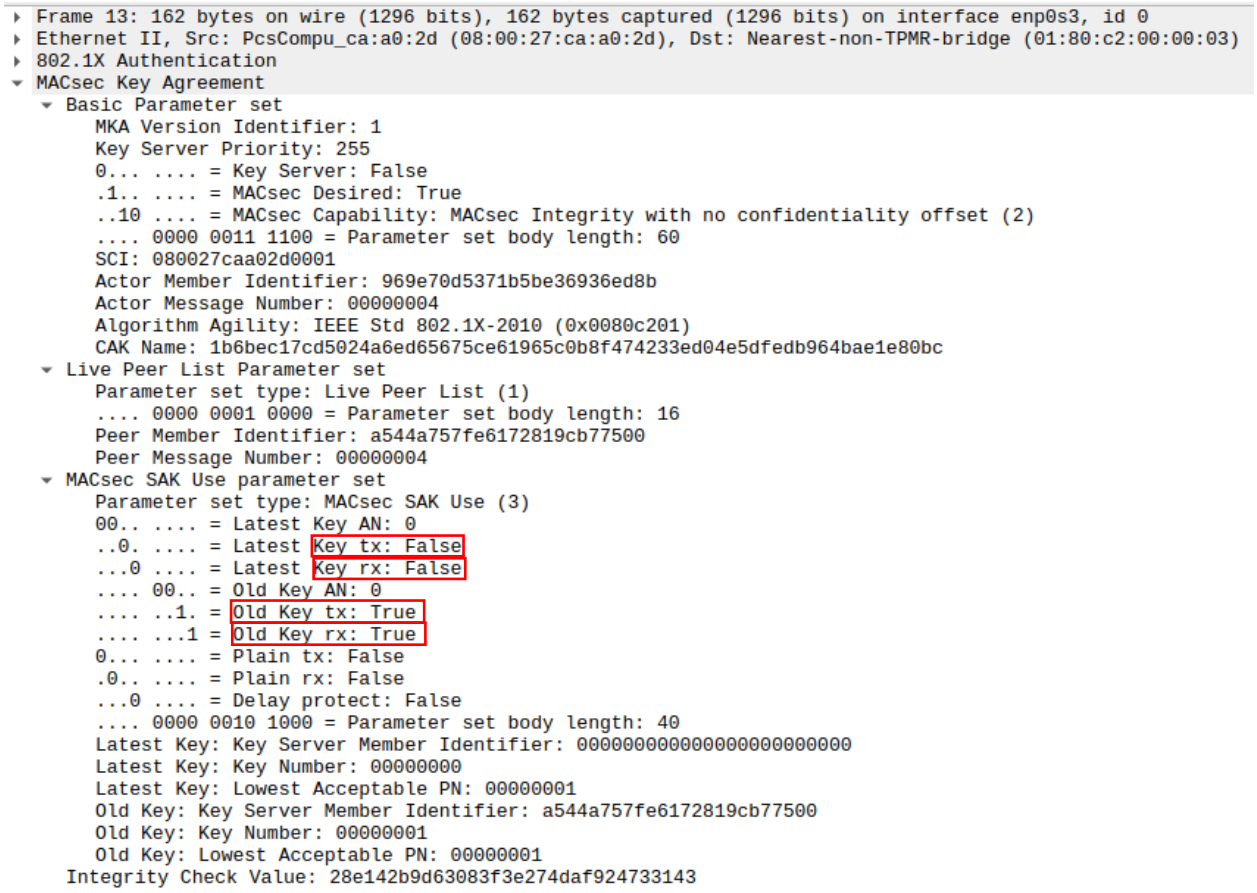


Figure 32. Keep alive frame

It can be observed that the Old Key now has the information that was previously held by the Latest Key. Additionally, the Latest Key has been set as False or 0.

Nevertheless, the SAK remains unchanged. If the SAK is modified, the configuration frame containing the Distributed SAK parameter set will be resent to all nodes that are part of the configuration, along with a new SAK.

9.3 MACsec configuration with 802.1X/EAP

When configuring MACsec between two Virtual Machines using PSK, the next logical step is to implement MACsec using another method called 802.1X/EAP. However, this method uses digital certificates to provide enhanced security, as each node has its unique digital certificate. This ensures a stronger authentication process and better protection against advanced threats. Certificates enable mutual authentication, meaning that both the sending and receiving node can verify each other's authenticity. This adds an extra layer of security to the system.

In the automotive context, when the 802.1X/EAP protocol is implemented, the supplicant would act as an Electronic Control Unit (ECU), and a powerful switch would be used to serve as both the server and authenticator. Each ECU would possess a unique digital certificate, ensuring a robust authentication process and better protection against advanced threats. These certificates enable mutual authentication, allowing both the sending and receiving ECUs to verify each other’s authenticity. Consequently, this approach adds an extra layer of security to the system.

9.3.1 Certificates generation

A digital certificate employs a key pair that includes both a public key and a private key to encrypt and decrypt information. It is trustworthy because it can only be signed by a public certificate authority that has undergone rigorous vetting. The certificate authority issues the certificate with an expiration date, after which the certificate is no longer guaranteed by the certificate authority. To obtain a digital certificate, you must send a request to a Certificate Authority (CA) of your choice, such as Verisign or RSA. The request includes your distinguished name, public key, and signature. A distinguished name is a unique identifier for each user or host for which you are applying for a certificate.

The CA verifies your signature using your public key and conducts some level of identity verification (which differs from one CA to another). After verification, the CA sends you a signed digital certificate that contains your distinguished name, public key, the distinguished name of the CA, and the signature of the CA. You must store this signed certificate in your key database. This allows you to obtain a trustworthy digital certificate that adds an extra layer of security to your system.

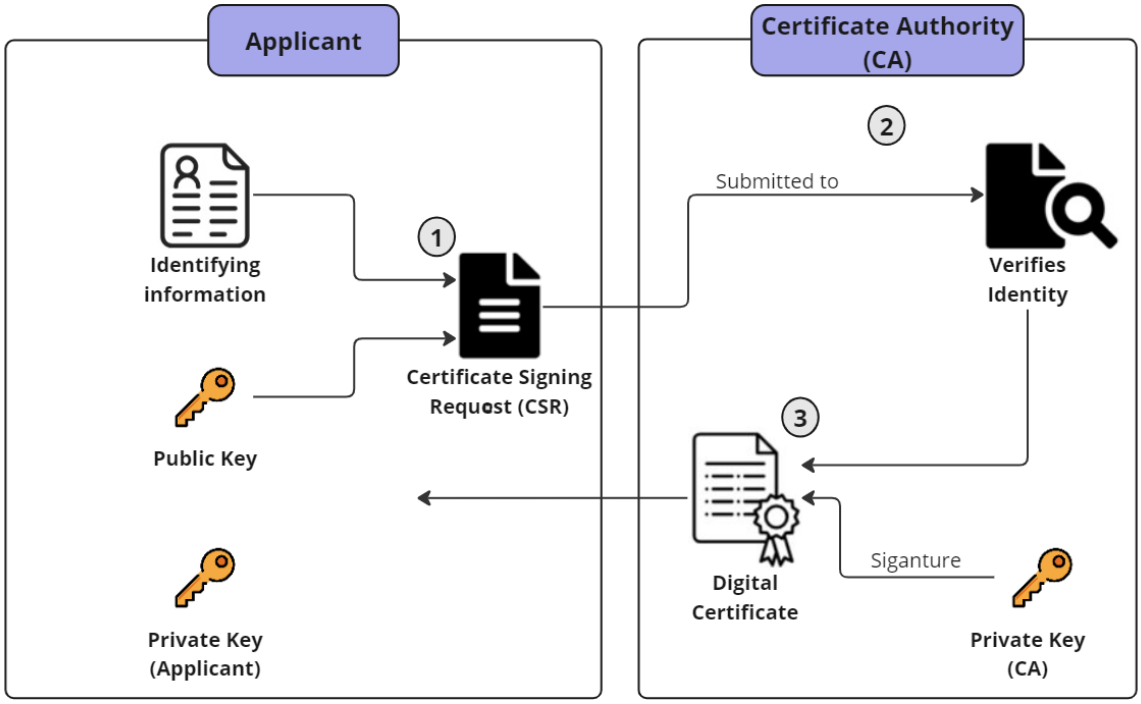


Figure 33. Digital Certificate process

As explained before, it is necessary to install a Certificate Authority in our Ubuntu system that can verify the applicant's identity and generate a trusted digital certificate. In the following chapters, I explain all the necessary steps required to set up and configure a Certificate Authority on Ubuntu 22.04.

Installing easy-rsa:
```
sudo apt install easy-rsa
```

Preparing a Public Key Infrastructure Directory:
```
mkdir ~/easy-rsa
ln -s /usr/share/easy-rsa/* ~/easy-rsa/   #symlinks creation
cd ~/easy-rsa
/easyrsa init-pki   #initialize the PKI inside easy-RSA directory
```

Creating a Certificate Authority:
```
nano vars
./easyrsa build-ca   #to build a CA
```

- **Easy-RSA:** It is a tool used to manage Certificate Authorities. To generate a private key and a public root certificate, the user will use this tool. The certificate will be used to sign requests from clients and servers that will rely on the CA.
  Upon installation of the package, the user will have everything they need to use Easy-RSA. Once done, the user can create a Public Key Infrastructure and start building the Certificate Authority.
- **Public Key Infrastructure Directory:** To create a skeleton Public Key Infrastructure (PKI) on the CA Server, the system will create a new directory called easy-rsa in the home folder.

This directory will be used to create symbolic links pointing to the easy-rsa package files that were installed in the previous step. These files are located in the /usr/share/easy-rsa folder on the CA Server.

- **Certificate Authority:** To create your Certificate Authority's private key and certificate, you will first need to create a file called "vars" with some default values. Once you've finished editing, save, and close the file. If you're using nano, you can do this by pressing CTRL+X, then Y, and ENTER to confirm.
  Now you're ready to build your CA. To create the root public and private key pair.

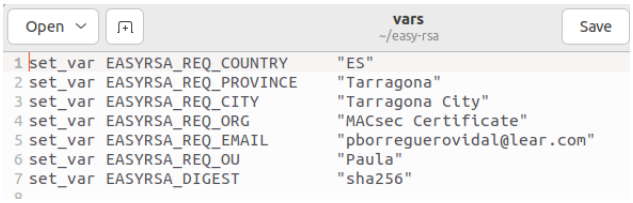


Figure 34. Certificate Authority

The output will display some lines about the OpenSSL version and prompt you to enter a passphrase for your key pair. You will need to enter this passphrase every time you interact with your CA, such as when signing or revoking a certificate.
You will also be asked to confirm the Common Name (CN) for your CA. The CN is the name used to refer to this machine within the context of the Certificate Authority. Although you can enter any string of characters, for simplicity's sake, you can press ENTER to accept the default name.

Now two important files are created: *~/easy-rsa/pki/ca.crt* and *~/easy-rsa/pki/private/ca.key*. These files are the public and private components of a Certificate Authority.

- The **ca.crt** file is the is the public certificate file of the CA. Users, servers, and clients use this certificate to verify that they belong to the same web of trust. Each user and server that uses your CA must have a copy of this file. All parties rely on the public certificate to ensure that someone is not impersonating a system and performing a Man-in-the-middle attack.
- The **ca.key** file is the private key that the CA uses to sign certificates for servers and clients. If an attacker gains access to your CA and, in turn, your ca.key file, you will need to destroy your CA. Therefore, you should keep your ca.key file only on your CA machine. Ideally, your CA machine should also be kept offline when not signing certificate requests to provide an extra layer of security.

Now that your CA is ready, it can be used to sign certificate requests and revoke certificates.

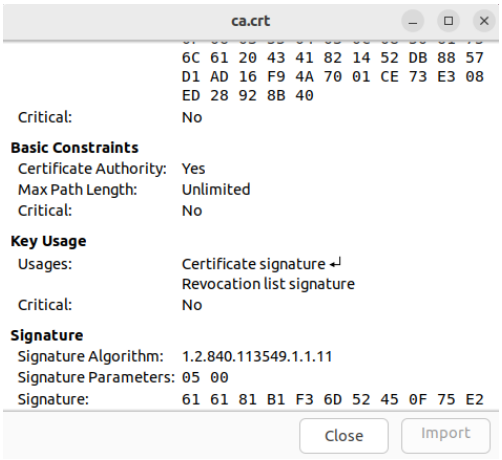


Figure 35. Certificate Authority certificate

The Key Usage extension is used when the subject public key is used to verify a signature on certificates. This extension can only be used in CA certificates.

9.3.2 Sign a Certificate with Root CA

In the following chapters, I explain how to sign a certificate using a Certification Authority.

Generate a Server key and Request For Signing (CSR):

```
openssl genrsa -aes256 -out server.key 4096
```

Create a Certificate Authority Certificate:

```
openssl req -new -key server.key -out server.csr

openssl req -noout -text -in server.key #OpenSSL verify the Server Key content
openssl req -noout -text -in server.csr   #OpenSSL verify the CSR
```

Sign a certificate with CA:

```
openssl x509 -req -days 365 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt
#issue the certificate server.crt, signed by the CA root certificate ca.crt and CA key ca.key which was created previously
```

- **Generate a Server Key and CSR:** This step involves creating a server key and generating a request (the .csr file) to have it signed by a Certificate Authority.
- **Create a CA certificate:** To create a certificate, a Certificate Signing Request (CSR) needs to be generated, containing the information to be included in the certificate. The CSR is signed with the subject's private key server.key to prove ownership of the private key. When generating the .csr file, it is important to carefully input the Common Name (CN), matching the DNS name or IP address specified in your Apache configuration.
- **Sign a certificate with CA:** Openssl takes the signing request (csr) and generates a one-year valid signed server certificate (crt) from it. The Certificate Authority (CA) to use, the CA key, and the Server key to sign must be specified. The serial number is set using CAcreateserial, and the signed key is output in a file named sever.crt.

9.3.3 Linux 802.1X/EAP implementation

After generating and signing all the digital certificates by the Certification Authority, 802.1X/EAP can be implemented. The structure of the configuration file used in the implementation is the same as the one used in the PSK implementation, but the content is different.

The wpa_supplicant configuration file should be as follows:

Wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant
eapol_version=2
ap_scan=1
fast_reauth=1

network={
      key_mgmt=IEEE8021X
      eap=TLS
      identity="08:00:27:e0:36:ee"
      ca_cert="ca.cer"
      client_cert="server.cer"
      private_key="server.p12"
      eapol_flags=0
      macsec_policy=1
}
```

The parameters added in this file are the following ones:

- **key_mgmt:** When using the IEEE 802.1X standard, the value that must be set is IEEE8021X.
- **eap:** In case of using IEEE 802.1X, you must indicate the EAP method to be used. The available options are MD5, MSCHAPV2, OTP, GTC, TLS, PEAP, or TTLS. EAP-TLS is the recommended option.
- **identity:** Value used as your EAP identity or username.
- **ca_cert:** Path to the Certification Authority (CA) certificate.
- **client_cert:** Path to the client certificate.
- **private_key:** Path to the client's private key.
- **private_key_passwd:** the password for the client's private key.

The configuration file stopped working properly after several attempts to run it on both virtual machines (VMs). Additional research on the implementation of 802.1X/EAP is recommended.

9.4 Results

EAP methods must be capable of generating and distributing cryptographic key material, this process is complex and particularly critical from a security perspective. Therefore, mature technologies should be employed. However, this limitation significantly narrows the choices among EAP methods. Currently, only methods based on TLS (Transport Layer Security) are recommended.

In automotive scenarios, pre-shared keys (PSKs) are commonly used for authentication. PSKs provide a simple way to verify the identity of Electronic Control Units (ECUs).

Authenticity is more important than confidentiality in the automotive context. It's essential to ensure that the data received is from a trusted source within the vehicle's network. If an unauthorized or compromised node injects false data, it could lead to dangerous situations. Trusting the source of data is crucial for safety-critical functions such as braking, steering, and engine control.

Even though confidentiality is less critical for non-private data such as sensor data and control signals, it's still essential to maintain a balance because certain information like location data or vehicle status may still require protection. Prioritizing authenticity while recognizing the context of non-private data ensures a reliable and secure automotive communication system.

10. MKA implementation on Ethernet Switch

10.1 Switch election

This chapter will explain all the essential features and requirements for Automotive Ethernet.

Automotive Ethernet is a specialized form of Ethernet network designed for vehicles. It has many key capabilities, including **1000BASE-T1**, which enables high-speed data transfer between different car components at speeds up to 1Gbps, using a single twisted pair of cables for full-duplex communication.

Integrated PHYs are also important in Automotive Ethernet, as they allow communication with other devices that have external ports in the switch, enabling fast and reliable communication within modern vehicles.

The Reduced Gigabit Media Independent Interface (RGMII) is a crucial port for Automotive Ethernet, as it serves as a critical link between the Ethernet MAC (Media Access Control) and the PHY (Physical Layer Transceiver) within vehicles. It uses 4 cables to transfer data in parallel, with each transmit (Tx) and receive (Rx) lane operating at 125 Mbps, achieving an effective throughput of 1 Gbps. Sampling occurs at a Double Data Rate (DDR), making it an efficient way of transferring data.

The **Serial Giga-bit Media Independent Interface (SGMII)** is a type of serial port used for communication in Automotive Ethernet. It allows transmit data up to 1000 Mbps, but it is more expensive and complex than other communication methods. SGMII uses two wires and operates at high speeds, making communication efficient and reliable.

Peripheral Component Interconnect Express (PCIe) is a type of serial bus that provides high data throughput, which is great for expanding various external devices.

MACsec, on the other hand, is a security technology that works at the layer 2 level. It secures data transmissions over Ethernet networks and ports. Since switches operate at Layer 2 of the OSI model, they do not support IPsec. As a result, the security standard used is MACsec. Thus, it is crucial that the switch has MACsec capability on the ports.

The switch used also must have AVB and mirroring capabilities. Furthermore, our device requires a 100BASE-TX port and a 10BASE-T1S port.

10.1.1 Ethernet Switch features

The switch used, meets all the requirements explained in the last chapter. This section provides more specific information about the switch, which has eleven different ports.

Each port supports different features:

Port	Switch port type	MACsec
1 and 2	100BASE-T1 // 10BASE-T1S MACPHY-SPI	Yes
3 and 4	100BASE-T1 // 10BASE-T1S	Yes
5 and 6	100/1000BASE-T1	Yes
7	100BASE-TX RGMII/RMII SGMII	No
8	RGMII/RMII	No
9	SGMII // HSGMII/USXGMII/KR	Yes

10	RGMII/RMII SGMII // HSGMII/USXGMII/KR PCIE3.0	No
11	SGMII // HSGMII/USXGMII/KR	No

Table 5. Ethernet Switch features

The Ethernet Switch supports IEEE 802.1AE MACsec and individual per port TX and RX MACsec module in port 1, 2, 3, 4, 5, 6 and 9.

The switch is equipped with MKA implementation, allowing it to independently establish a Secure Channel. The key features of the MKA implementation are as follows:

- It uses the IEEE 802.1X-2020 MKPDU, version 3, standard for secure communication.
- It supports pre-shared Controlled Access Key (CAK) for authentication.
- It facilitates Dynamic Key Server election for efficient key management.
- It enables the use of the XPN parameter set for enhanced security.
- It supports the GCM-AES cipher suite for encryption and authentication.
- It allows up to 4 peers to operate within the same Controlled Access (CA) domain.
- It is compatible with external MACsec PHY for seamless integration.
- It incorporates True Random Number Generator (TRNG) for generating the Secure Association Key (SAK).
- It supports Hardware Security Module (HSM), ensuring that the SAK remains within the HSM and is inaccessible by the CPU during its lifetime.

However, it's important to note that the MKA implementation has certain limitations:

- It does not support EAP (Extensible Authentication Protocol), limiting the authentication methods available.
- Only one CA is supported per port, which may restrict certain deployment scenarios.
- Group CAK is not supported, potentially impacting key distribution in certain network setups.
- KMD (Key Management Distribution) parameter set is not supported, which may limit certain key distribution configurations.
- Temporary suspension of MKA is not supported, which may impact certain maintenance or troubleshooting procedures.

10.2 Ethernet Switch Automotive Switch Tool used

This section has been deleted due to confidentiality issues.

10.3 Linux MKA configuration

In the test cases, Linux PCs are used as an end point in its implementation with a switch. I’ve implemented two different MKA configurations on them. The first one is implementing wpa_supplicant (already explained in MACsec configuration with PSK section) and the other implementation is using Technica MKA Daemon.

10.3.1 Technica MKA Daemon

The Technica MKA Daemon is a software project that is open-source and has been developed by Technica Engineering. This daemon currently supports Linux operating systems and implements MKA according to IEEE802.1X-2020 specifications.

The Linux implementation of the MKA Daemon has been tested on Ubuntu 20.04 and 22.04. It complies with IEEE 802.1X-2020, the MKA standard. It does not yet support Extended Authentication Protocol (EAP), but it does support Pre-Shared Key authentication. The GCM-AES-128, GCM-AES-256, GCM-AES-XPN-128, and GCM-AES-XPN-256 MACsec algorithms are all supported.

There are two different ways to license the software: regular commercial licensing or open source (licensed under GPLv2). The open-source license is applied in the test scenarios.

A daemon is a background application that doesn't require user input to operate. In this context, the MKA Daemon continuously manages MACsec key agreements, ensuring secure communication between network devices.

10.4 Setup development

- 2x Linux laptops
- 2x Technica 100/1000BASE-T1 Media-Converter MATEnet
- 2x Ethernet switch

10.5 Test cases

10.5.1 Test 0: Linux to Linux MKA (without switch)

In this test case, the configuration is similar to the first test done using a laptop with two Ubuntu 22.04 machines. The only difference is in the setup, which involves two Linux laptops connected to each other using an ethernet cable.

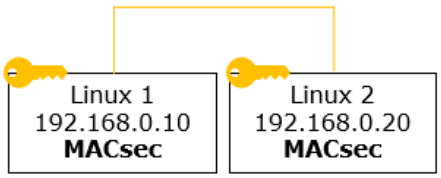


Figure 36. Setup: Linux - Linux

The MKA configuration was done using two methods: first using the configuration file of wpa_supplicant, and then the Technica MKA Daemon configuration. Both configurations worked as expected, and the results obtained were the same in both tests.

The wpa_supplicant configuration file is explained in the chapter MACsec configuration with PSK. Here's the Technica MKA Daemon configuration file that was used:

```
# Copyright 2021 Technica Engineering
#
# This file fully configures the MKA daemon.
#

# Global settings ------------------------------------------------
log_level: debug
verbosity: 1
hello_time: 2000
bounded_hello_time: 500
life_time: 6000
sak_retire_time: 3000
```

```
hello_time_rampup: 100 200 400 800 800
transmit_empty_dist_sak: on
transmit_empty_sak_use: on
transmit_null_xpn: on
secy_polling_ms: 500

# Interface settings ------------------------------------------------
interfaces:
  - device: "enp0s25"
    protected_device: "enp0s25.macsec"
    intf_mode: STATIC
    macsec: INTEGRITY
    kay: on
    priority: 11
    role: AUTO
    replay_protect: 0
    delay_protect: on
    unauth_allowed: never
    unsecure_allowed: never
    ciphers: GCM_AES_128
    cak: 07 4C 7D C3 11 CC 18 8F AB A4 8C 85 18 A5 21 82
    ckn: 21 4A 65 20 70 68 BE A3 4B 6C 0C 3D 1E 21 47 C8 7C 83 45 1B 84
B9 0B FA E1 5D C1 9B
    drv_macsec_mode: software

# To prevent the switch from discarding ping messages, the following has
been added
    phy_transmit_sci: true
```

As a result of the MKA protocol, the link between both PCs is now MACsec protected. To confirm the success of the protocol, a ping command is done to the other Linux machine, thus confirming that the ARP messages and the PING messages are sent with the MACsec tag.

In Linux PCs, the Techinca MKA Daemon configuration file is used to configure MACsec in all the following setups due to versions issues.

Test results

As shown in the image, different MKA frames are exchanged between both nodes. Once the protocol is established, I sent a ping message to the other Linux PC to make sure that the messages sent are MACsec protected. The image shows an ARP message, which is MACsec tagged, which means that the message is MACsec protected. The protocol succeeded.

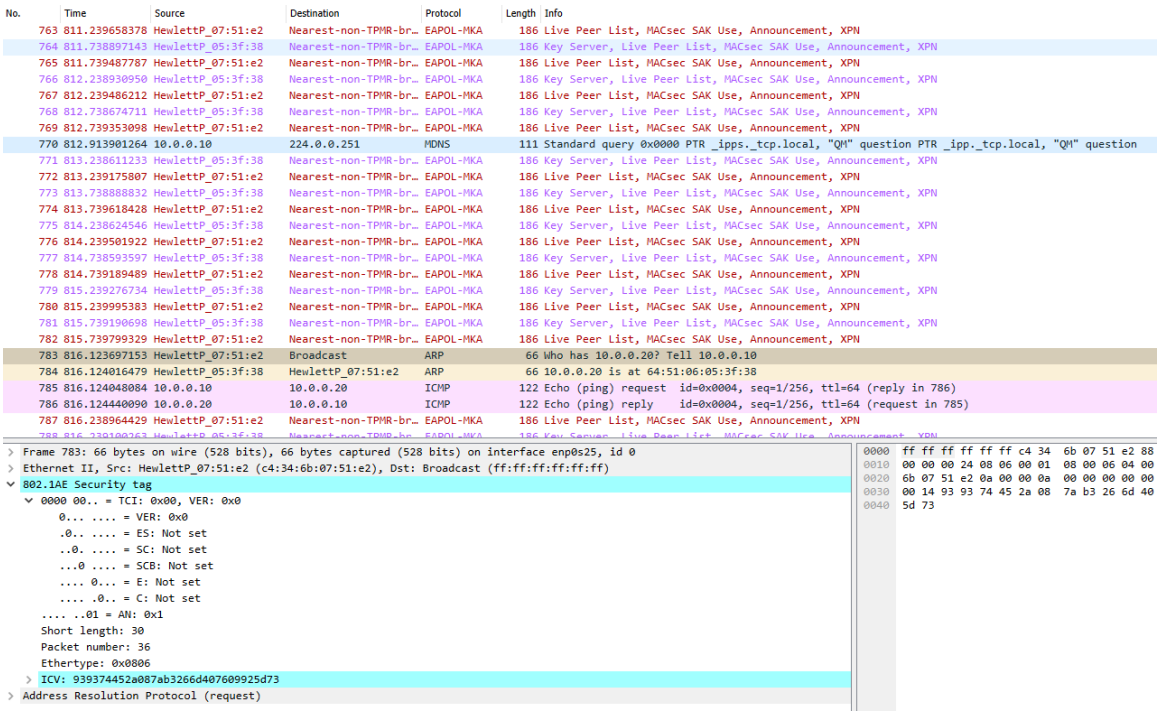


Figure 37. MKA exchanged frames

10.5.2 Test 1: Hop-By-Hop MACsec

For this test, two Linux laptops are utilized, connected through a Realtek RLT9071CP switch. To ensure secure point-to-point communication, MACsec must be configured on each endpoint of the communication. There are two point-to-point links in this setup: the first between Linux 1 and port 5, and the second between Linux 2 and port 6. Each link would use a different CAK and CKN.

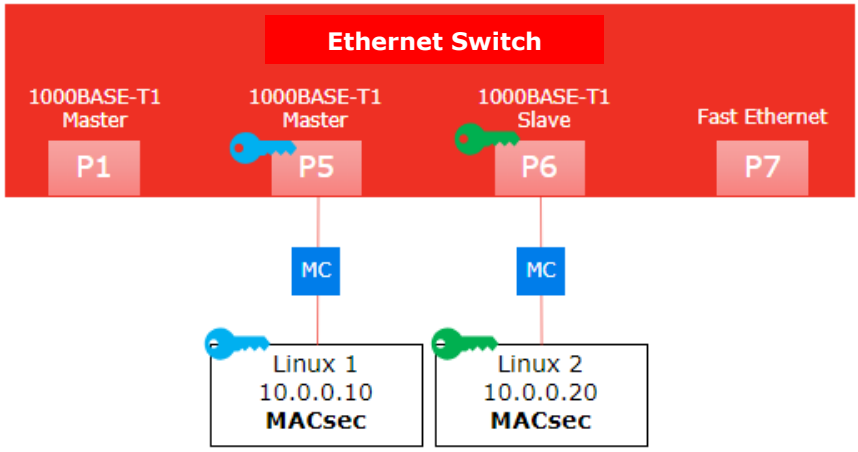


Figure 38. Setup: Linux -Switch - Linux

To configure MACsec in Linux, the Technica MKA Daemon is used instead of wpa_supplicant due to incompatibility issues caused by different MKA versions. Wpa_supplicant used MKA version 1 while the switch used MKA version 3. However, Technica MKA Daemon allows us to configure MKA version 3 in the laptops.

To configure MACsec in the switch, you should follow the steps explained in the previous chapter.

It's worth mentioning that the configuration implemented has the following important details:

- When running the Technica MKA Daemon configuration file, IP addresses are created on the MACsec interface, which are then used to implement the testing process. This is an important detail; this step will be done in all the next setups done.
- The configured CAK in each endpoint has 16 bytes since the GCM-128 cipher suite is used. In case a 32 bytes CAK is used, the ICV becomes 0 due to an error. To avoid it, it's necessary to choose GCM-256 cipher suite.
- To prevent the switch from discarding ping messages, the following command has been added. This is a limitation of the Ethernet Switch used. In the configuration of switch ports, the parameter "Force SCI" must be enabled as well.

Added parameter in Technica MKA Daemon configuration file:

```
phy_transmit_sci: true
```

Test results

In the given image, it can be observed that Linux with IP address 10.0.0.10 is attempting to send a ping message to another laptop with IP address 10.0.0.20. However, due to the limitations of the switch, the messages are being discarded, which is preventing Linux 1 and Linux 2 from communicating with each other.

No.	Time	Source	Destination	Protocol	Length	Info
313	118.620725	HewlettP_07:51:e2	Broadcast	ARP	66	Who has 10.0.0.20? Tell 10.0.0.10
316	119.644594	HewlettP_07:51:e2	Broadcast	ARP	66	Who has 10.0.0.20? Tell 10.0.0.10
320	120.668564	HewlettP_07:51:e2	Broadcast	ARP	66	Who has 10.0.0.20? Tell 10.0.0.10
323	121.692708	HewlettP_07:51:e2	Broadcast	ARP	66	Who has 10.0.0.20? Tell 10.0.0.10
327	122.716578	HewlettP_07:51:e2	Broadcast	ARP	66	Who has 10.0.0.20? Tell 10.0.0.10

Figure 39. ARP messages

As previously discussed, the solution to this problem is to add the command to the Technica MKA Daemon configuration file, as defined earlier. Once this command is added, the switch will no longer discard the messages, enabling both PCs to establish communication.

In the next image, it can be seen that the Linux PC with IP address 10.0.0.10 successfully sends a ping request message to 10.0.0.20 and receives a ping reply message in return. These messages are MACsec tagged, and with the addition of the SCI command, this parameter can be observed in the frame.

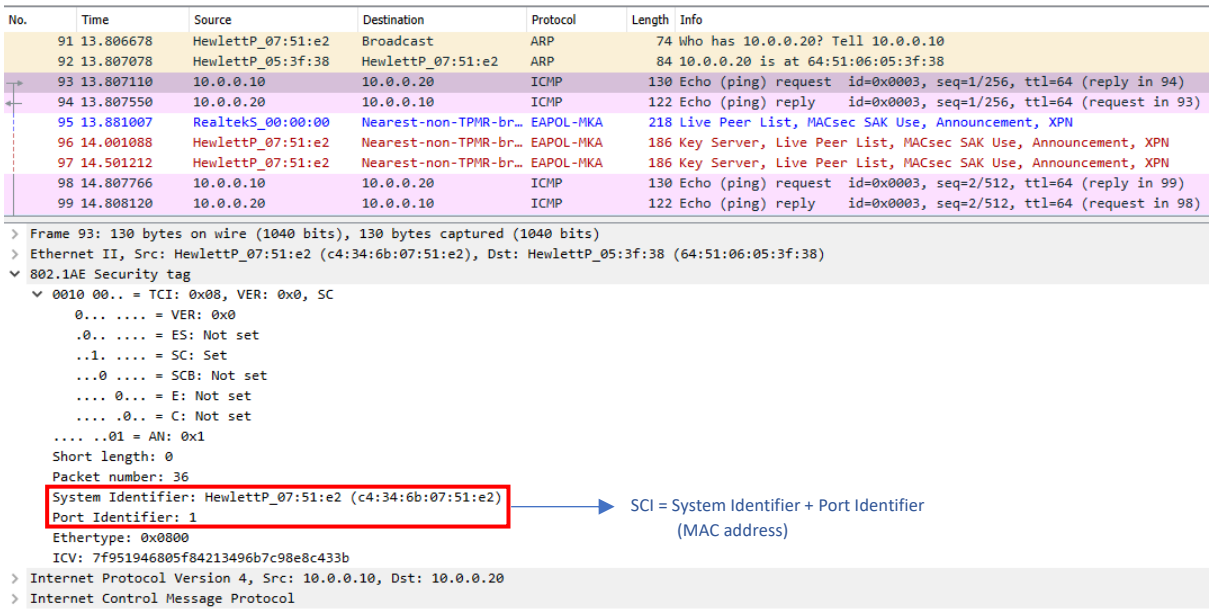


Figure 40. Succeed MACsec communication

10.5.3 Test 2: End-to-End MACsec

In this particular test scenario, the switch ports have not been configured with MACsec. The only configuration that has been implemented is Technica MKA Daemon on both Linux systems, with the same CAK and CKN.

It is important to note that this end-to-end MACsec communication setup is not recommended. Therefore, no further tests will be conducted using this setup.

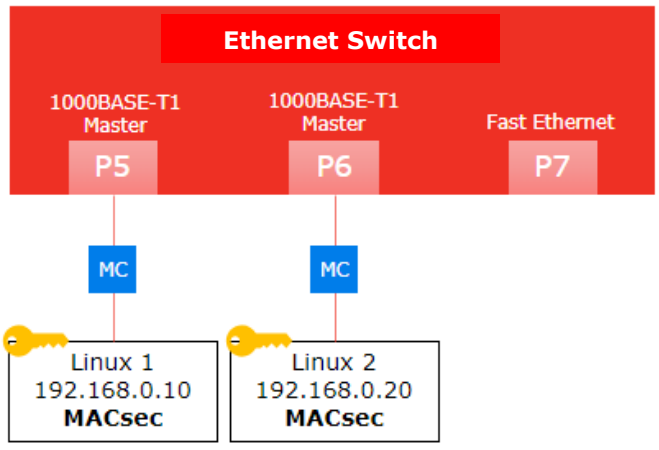


Figure 41. Setup: Linux - Switch - Linux

Test results

However, it has been observed that no MKA frames are being transmitted by the switch, which indicates that MACsec is not configured correctly. The Ethernet Switch supplier has confirmed that this is the expected behavior under these circumstances.

10.5.4 Test 3: MACsec only Between Switches

In the following test the setup is based on two switches connected through an ethernet cable, each switch is at the same time connected to a Linux PC.

In our test, the Linux PC and the switch simulate an Electronic Control Unit (ECU). To ensure secure communication between the ECUs, enabling MACsec is essential.

As the Linux PC is working as the host of the switch, it is not necessary to protect this link. This is because both the Linux PC and the switch are part of the same ECU and any external attempts to hack into the system would not be possible. Its connection will be protected by intra-ECU security measures. Therefore, the test where MACsec is only enabled between switches is a real setup implemented in the automotive industry.

From all the tests done, this is the most interesting scenario because a real application connection would be like the one done in test 3.1, enabling MACsec only between switches.

- **Test 3.1: MACsec only between switches:**

This is the most interesting scenario, because a real application connection would be like the one done in this test.

In this setup only the Port 6 from both switches is MACsec configured. EAPOL-MKA frames are only transmitted between the switch, only the packets on that link will be protected with MACsec.

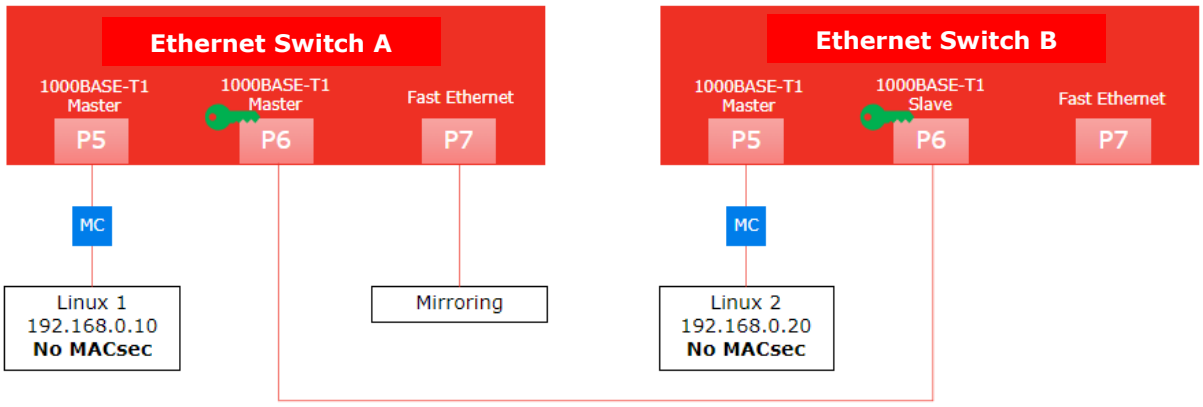


Figure 42. Linux - Switch (MACsec) – Switch (MACsec) - Linux

To be able to observe the frames sent between them, I do mirroring from the Port 7 of the switch A. P6 mirrored traffic shows MKA, but traffic is not MACsec tagged. This is the expected behavior as the mirroring is done after the port decrypts the data sent. Each port has specialized hardware responsible for handling the encryption and decryption of MACsec traffic.

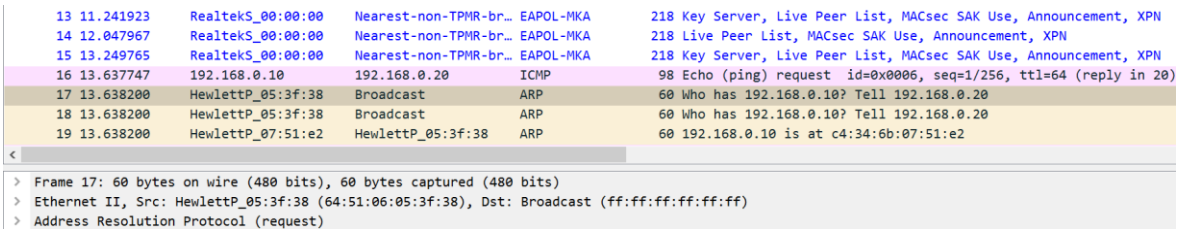


Figure 43. Port 6 traffic mirrored

- **Test 3.2: MACsec everywhere:**

In this setup, MACsec is configured in all the links. Although this is not going to be a real test case, it will guarantee that the frame is MACsec protected from the sender until its destination.

EAPOL-MKA frames are transmitted from Linux 1 to switch A, from switch A to switch B and from switch B to Linux 2, the packets transmitted in all those links from this setup will be MACsec protected.

In order to observe the frames sent between them, I do mirroring from the Port 7 of the switch A. P5 mirrored traffic shows MKA frames MACsec tagged. This is the expected behavior as port 5 is receiving MACsec tagged traffic form Linux 2.

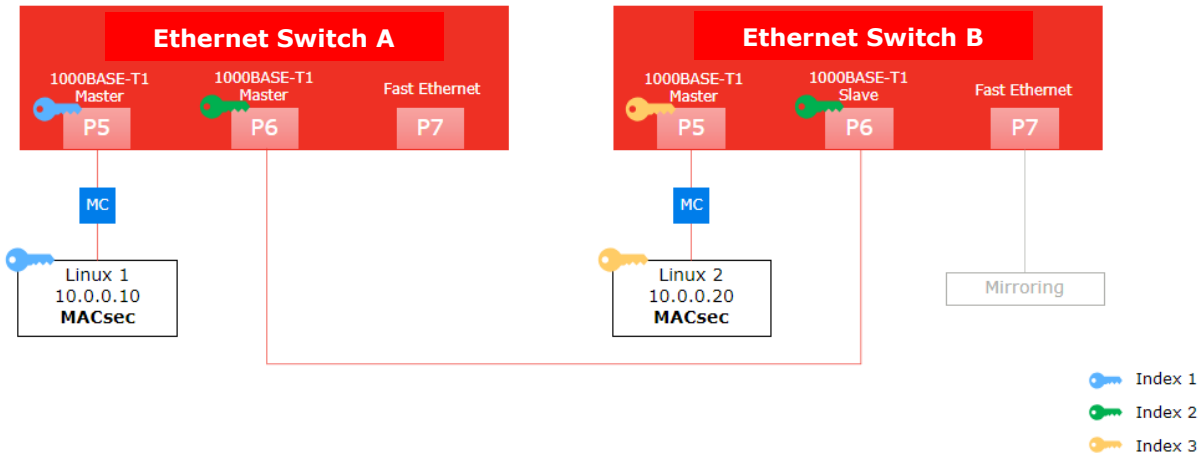


Figure 44. Linux (MACsec) - Switch (MACsec) - Switch (MACsec) - Linux (MACsec)

In the next image, it can be observed that Linux with IP address 10.0.0.20 successfully sends a ping request message to 10.0.0.10 and receives a ping reply message in return. These messages are MACsec tagged, as expected.

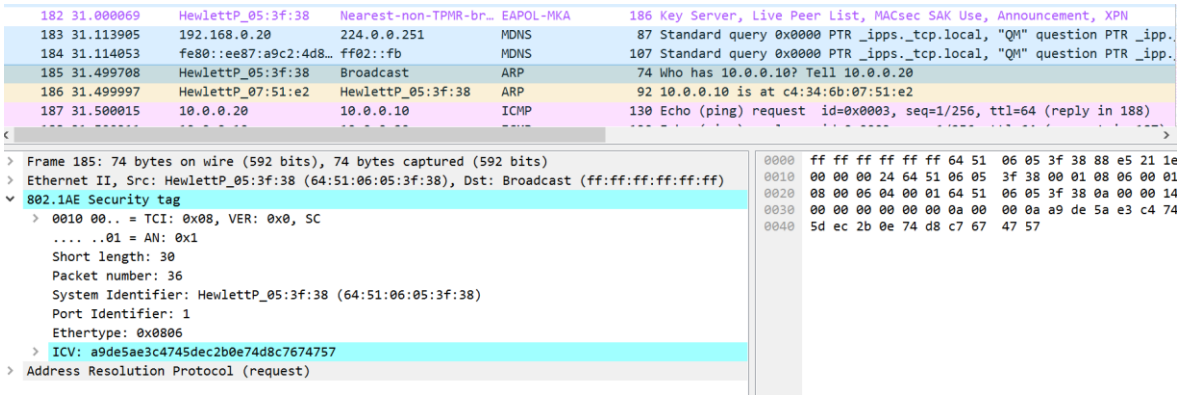


Figure 45. MACsec tagged frame

10.6 Performance analysis

A performance analysis is done to compare the packets error rate.

The iperf command, a Linux tool for measuring network efficiency, is used for this test. Within a network, it can also transmit packets between two devices. In this scenario, the goal of implementing iperf is to saturate the network in order to verify that the Switch configuration is operating as expected.

- Linux 1: Iperf server
- Linux 2: Iperf UDP client

```
iperf -u -s  #Linux1
iperf -u -c 10.0.0.10 -t 10 -b 0   #Linux2
```

The performance analysis has been done on the following tests setups:

- Test 0.0. Linux (No switch, no MACsec)
- Test 3.1. MACsec between switches
- Test 3.2. MACsec on all links
- Test 3.3. No MACsec (two switches)

	Transfer (GB)	Bitrate (mb/s)	Jitter (ms)	Lost/total datagrams (%)
Test 0.0	1,1	944	0,020	0,0034
Test 3.3	1,1	945	0,025	0,0041
Test 3.1	1,07	917	0,020	3,0000
Test 3.2	1,07	916	0,016	0,2300

Table 6. Performance análisis I

The analysis was conducted using the maximum bitrate allowed by the switch, which is 1 Gbps. It was observed that the maximum rate achieved was 945 Mbps with this bandwidth. Upon analyzing the setup in test 0.0, it was demonstrated that the Linux PCs did not have an impact on the number of lost datagrams. The bitrate and the number of lost datagrams were almost the same for both test case 0.0 and test case 3.3, indicating that adding switches does not increment the error rate.

During test 3.1, it was observed that the number of lost datagrams was higher than expected, with a lost datagram percentage of 3 %. However, once MACsec was enabled in all the links (test 3.2), the lost datagrams decreased to 0.23 %. Additionally, the results indicated that the bitrate was almost the same for both setups, test 3.1 and 3.2.

	Transfer (MB)	Bitrate (mb/s)	Jitter (ms)	Lost/total datagrams (%)
Test 3.1	334	279	0,01	0,00041
Test 3.1	334	279	0.007	0

Table 7. Performance analysis II

To observe if reducing the bandwidth would also reduce the error rate, test 3.1 was repeated using a lower bandwidth. The bandwidth was reduced until the packet loss was equivalent to when MACsec was not configured. When the bandwidth was set to 280 Mbps, the maximum rate achieved was 279 Mbps, and the error rate became almost 0 %, as expected.

However, the next chapter is going to study why such a high error rate is obtained when MACsec is configured in only one of the links in the setup.

To understand why a reduction of 70% in the maximum bandwidth, from 1 Gbps, is necessary to avoid

It is necessary to understand why with a maximum bandwidth of 1 Gbps, it is needed to reduce the bandwidth by 70 % to avoid error rate. It is necessary to find out if the error is in the measurement taken or if the Ethernet switch affects with the measurement. For this reason, contacting with the supplier is necessary to reach a correct conclusion about the results obtained in the measurements made.

10.6.1 Performance analysis on test case 1.2

In this test case, the objective is to analyze the reason behind the unexpected error rate observed in the previous chapter when MACsec was configured only between the switches. To do this analysis, a simple setup has been implemented.

The MACsec (Media Access Control Security) protocol is configured and enabled for the communication link between Linux 1 and the switch. This means that all traffic transmitted between Linux 1 and the switch is encrypted and authenticated using MACsec. On the other hand, the connection between Linux 2 and the switch does not utilize MACsec for securing the communication link.

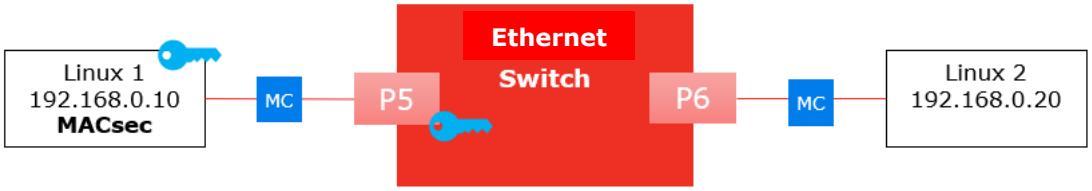


Figure 46. Test case 1.2

In the recent test, it is observed that the messages sent from Linux1 to Linux2 does not introduce error rate, whereas when messages are sent from the other direction an error rate of 3.1 % is introduced. This is due to the 32 Bytes introduced in port 5, as its link with Linux 1 is MACsec protected. When using MACsec, a 32-byte tag is added.

When using MACsec, a 32-byte tag is added on the packet length.

Case 1:

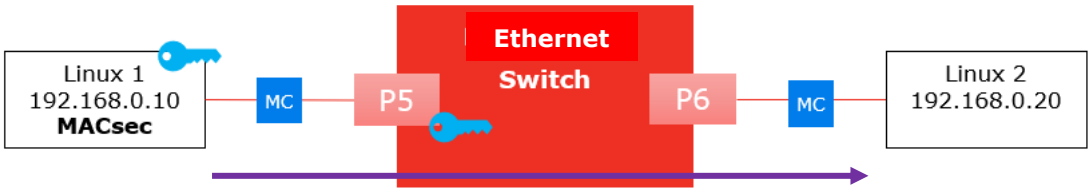


Figure 47. Case 1

Case 2:

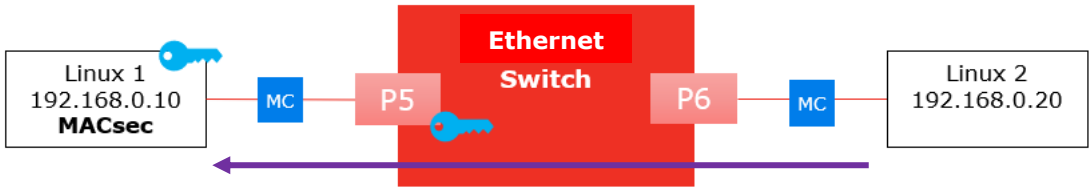


Figure 48. Case 2

In the second scenario, data is being transmitted from Linux 2 to Linux 1. The source does not have MACsec configured, so the packet length is set to its maximum value because no extra bytes are expected. When the frame arrives at port 5, which is configured with MACsec, it adds 32 bytes to the frame length, resulting in a packet size longer than expected. This leads to saturation and packet loss.

The test results showed that one packet out of 33 was lost, indicating a loss rate of 3 %.

The packet length was measured, resulting in a packet size of 1490 bytes for MACsec-tagged packets and 1458 bytes for non-MACsec-tagged packets. This difference in packet size explains the variance in error rates between the two scenarios.

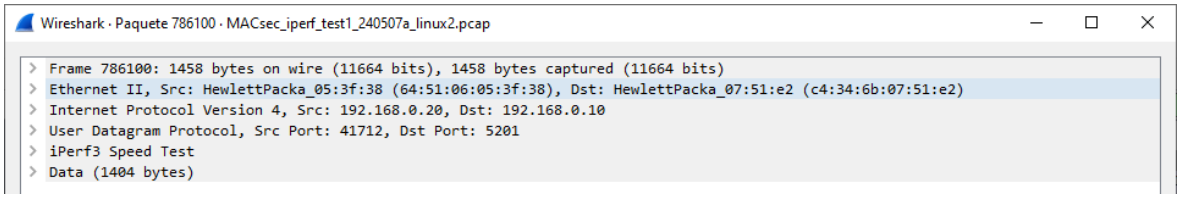


Figure 49. Non-MACsec tagged packets

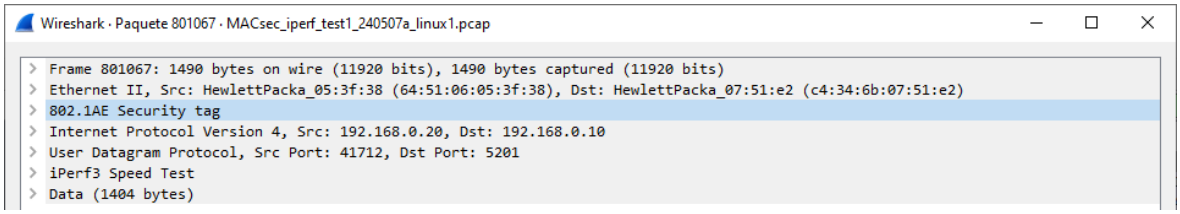


Figure 50. MACsec-tagged packets

11. Conclusions

After the successful completion of the project and the analysis of the final results, several significant conclusions have been derived. This chapter aims to provide a detailed explanation of these results and conclusions, as well as outline essential future steps necessary to gain deeper knowledge in MACsec and MKA protocols.

All objectives set at the beginning of the project have been successfully achieved.

The first objective involved analyzing the MACsec Key Agreement protocol (MKA) by studying the 802.1X standard, where this protocol is specified. This analysis has been successfully completed, and the necessary documentation has been written, it can be find in the first part of this project. I would like to emphasize the amount of knowledge acquired during this research. Understanding how this protocol works was essential for implementing it in different setups explained in the various chapters of the project. Therefore, it can be concluded that the first objective has been achieved.

Regarding the second objective, MKA has been successfully configured in the test cases. This configuration required prior research on how to be done. To select the best implementation, extensive research was conducted to compare different information and finally select the one that best fits our requirements, which involves using Pre-Shared Keys.

The third objective involves conducting a detailed analysis of the results obtained in the test case where Pre-Shared Keys were used. This analysis was carried out to understand in detail all the components involved in the implementation of the MKA protocol and all the parameters that are part of it. This analysis has been successfully completed, leading to significant conclusions about the different aspects of the protocol. The necessary documentation has been written and can be found in chapter 9.2.

The fourth objective aimed to implement 802.1X/EAP, which relies on the Extensible Authentication Protocol (EAP) and requires digital certificates to trust participating nodes in secure MACsec communication. While digital certificates and a Certification Authority were successfully generated, the implementation of 802.1X/EAP itself was not achieved due to the requirement for a RADIUS server and an authenticator device, which necessitated further knowledge beyond the scope of this project.

The implementation of the 802.1X/EAP require further study, making it an interesting future step. The EAP protocol is extensible and complex; thus, a new future project could explore the makeup of this protocol and its implementation in an automotive application.

The fifth objective involved implementing the MKA protocol using PSK in a real automotive application, utilizing an Automotive Switch. Despite encountering challenges that required close collaboration with the supplier, the implementation of a real automotive application was successfully achieved, fulfilling the main goal of the project.

I want to highlight the opportunity I had to engage in an advanced project, which included collaborating with an Automotive Switch. One of the project's core requirements was the implementation of MACsec, and my involvement in this protocol’s implementation required me to contribute to this aspect.

Moving forward, additional research and collaboration with Realtek will be required to conduct performance analysis and reach conclusions regarding the error rate observed during the implementation of MACsec in the setup simulating a real automotive application.

Through this project, I have gained expertise and a better understanding of MACsec and MKA protocols. I also had the opportunity to work with state-of-the-art technologies currently used in the automotive industry. This experience has allowed me to work in an innovative field and grow professionally. I have improved my technical, professional, and personal skills, and gained experience and knowledge about MACsec implementation in a real automotive application. This has given me insight into its future perspectives and impact on upcoming automotive projects.

12. Annexes

12.1 Hop-by-hop communication

Hop-By-Hop architecture refers to the way data packets are transmitted form one network node to another. In the automotive industry this is the communication used.

In hop-by-hop communication, data packets are processed and forwarded by each intermediate network device (such as routers or switches) independently. At each hop, the device examines the packet header, makes forwarding decisions, and sends the packet to the next hop along the path. However, this approach doesn't consider the entire end-to-end communication and may not be suitable for all scenarios.

For example, if you need to transfer large files over a network, hop-by-hop communication could result in slower transfer times and increased latency. In this case, end-to-end communication may be more effective.

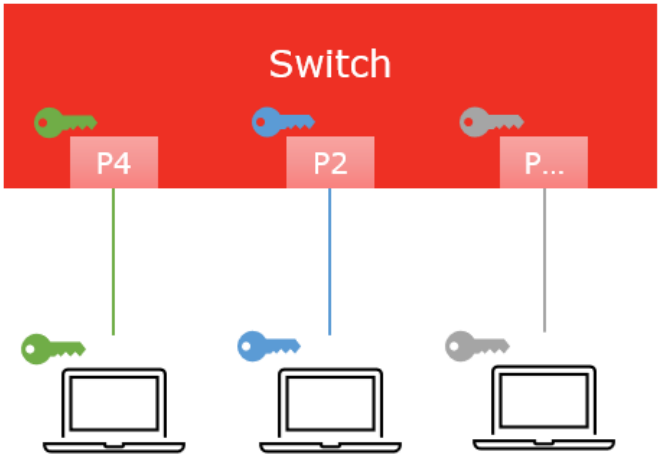


Figure 51. Hop-by-Hop communication

12.2 End-to-End communication

End-to-end communication focuses on ensuring correctness and reliability at the endpoints (source and destination) rather than relying heavily on intermediate devices. The network treats the communication as a whole, emphasizing the end hosts. Intermediate devices perform minimal processing and forwarding, leaving complex functions (such as error recovery, flow control, and security) to be implemented at the endpoints.

For instance, hop-by-hop communication may be more suitable for real-time applications such as VoIP (Voice over IP) where low latency is critical, while end-to-end communication may be a better fit for file transfers or email communication where reliability and data integrity are paramount.

The current mode has certain limitations, including scalability issues, multicast and broadcast traffic vulnerability, and complicated ECU integration. While end-to-end encryption may work for VPNs, the automotive industry requires hop-by-hop MACsec for better protection.

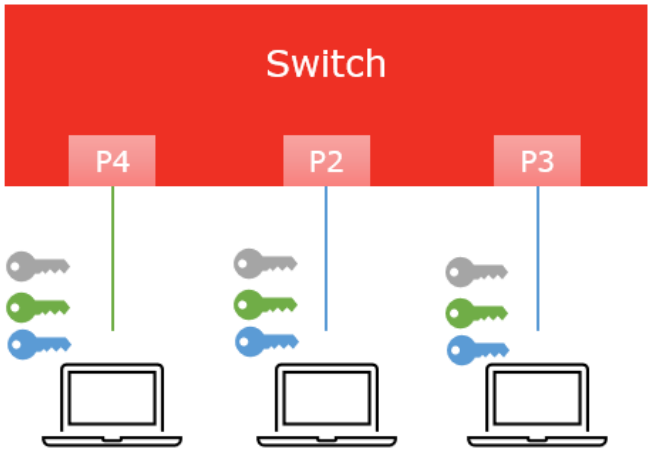


Figure 52. End-to-End communication

13. Referencies

1) DigitalOcean. (n.d.). How to set up and configure a certificate authority on Ubuntu 22.04. Retrieved February, 2024, from https://www.digitalocean.com/community/tutorials/how-to-set-up-and-configure-a-certificate-authority-on-ubuntu-22-04
2) GoLinuxCloud. (n.d.). Create certificate authority and sign a certificate with root CA. Retrieved February, 2024, from https://www.golinuxcloud.com/create-certificate-authority-and-sign certificate/
3) GoLinuxCloud. (n.d.). OpenSSL create self-signed certificate Linux with example. Retrieved February, 2024, from https://www.golinuxcloud.com/openssl-create-self-signed-certificate/
4) IBM. (n.d.). Generate root CA key. Retrieved March, 2024, from https://www.ibm.com/docs/en/runbook-automation?topic=certificate-generate-root-ca-key
5) Saylor Academy. (n.d.). Cryptography. Retrieved March, 2024, from https://learn.saylor.org/course/view.php?id=453&sectionid=16689
6) Huawei. (n.d.). MACsec configuration commands. Retrieved January, 2024, from https://support.huawei.com/enterprise/en/doc/EDOC1100278255/3137975f/macsec-configuration-commands
7) H3C. (n.d.). MACsec overview. Retrieved January, 2024, from https://www.h3c.com/en/d_201609/950843_294551_0.htm
8) Microsoft. (n.d.). Certificate requirements for EAP-TLS/PEAP. Retrieved February, 2024, from https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/certificate-requirements-eap-tls-peap
9) FreeBSD. (n.d.). WPA supplicant configuration. Retrieved January, 2024, from https://man.freebsd.org/cgi/man.cgi?wpa_supplicant.conf(5)
10) Tran, V. (n.d.). How to install FreeRADIUS on Linux. Retrieved March, 2024, from https://www.vulongtran.com/how-to-install-freeradius-radius-linux
11) VOCAL Technologies. (n.d.). AES GCM encryption algorithms. Retrieved May, 2024, from https://vocal.com/cryptography/gcm-and-gmac-authenticated-encryption-algorithms/
12) Cryptography Stack Exchange. (2022). How does AES GCM encryption work. Retrieved May, 2024, from https://crypto.stackexchange.com/questions/101106/how-does-aes-gcm-encryption-work
13) Cryptosys.net. (2023). AES-GCM authenticated encryption. Retrieved May, 2024, from https://www.cryptosys.net/pki/manpki/pki_aesgcmauthencryption.html
14) Nokia. (n.d.). MACsec. Retrieved January, 2024, from https://infocenter.nokia.com/public/7750SR217R1A/index.jsp?topic=%2Fcom.nokia.Interface_Configuration_Guide_21.7.R1%2Fmacsec_802-1ae_-ai9emdynxf.html
15) Cisco Systems. (n.d.). Configuring certificate-based MACsec with local authentication. Retrieved May, 2024, from https://community.cisco.com/t5/networking-knowledge-base/configuring-certificate-based-macsec-with-local-authentication/ta-p/4494657

16) CommScope. (n.d.). MACsec on Linux. Retrieved January, 2024, from https://docs.commscope.com/bundle/fastiron-08090-securityguide/page/GUID-877639DC-7DF2-486B-B266-B98A4924B106.html
17) Hewlett Packard Enterprise. (n.d.). MACsec configuration. Retrieved January, 2024, from https://techhub.hpe.com/eginfolib/networking/docs/switches/5130hi/5998-8419b_security_cg/content/496254775.htm
18) Technica Engineering. (n.d.). MKAdaemon. Retrieved April, 2024, from https://github.com/Technica-Engineering/MKAdaemon
19) IEEE Standards Association. (2018). IEEE 802.1AE-2018: IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Security
20) IEEE Standards Association. (2020). IEEE 802.1X-2020: IEEE Standard for Local and metropolitan area networks—Port Based Network Access Control
21) Centro Criptografico Nacional – Informe CCN-PYTEC nº9 – Recomendaciones de Seguridad para MACsec