Bachelor's Thesis

Enhancements for MACsec providing transparent Layer-2 encryption

Jan Sönke Huster

March 26, 2018

TU Dresden
Faculty of Computer Science
Institute of Systems Architecture
Chair of Privacy and Data Security

Professor: Prof. Dr. Thorsten Strufe
Supervisors: Dr.-Ing. Stefan Köpsell
Dipl. Inf. Tim Lackorzynski

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Datenschutz und Datensicherheit

AUFGABENSTELLUNG FÜR DIE BACHELORARBEIT

Name, Vorname des Studenten: *Jan Sönke Huster*

Immatrikulationsnummer: *4084528*

Studiengang: *Bachelor Informatik (2012)*

Thema (deutsch): **Verbesserungen bezüglich transparenter Layer-2 Verschlüsselung mit MACsec**

Thema (englisch): **Enhancements for MACsec providing transparent Layer-2 encryption**

Zielstellung:

Mit MACsec (IEEE 802.1AE) steht ein Protokoll zur Verfügung, dass eine Integritätsprüfung sowie Verschlüsselung des Datenverkehrs auf Schicht 2 ermöglicht. Dabei werden die zu übertragenden Ethernet-Pakete mit einem MAC geschützt und gegebenenfalls verschlüsselt. Problematisch bei der aktuellen Spezifikation und Implementierung von MACsec ist, dass durch die MACsec-Header und dem angehangenen Message Authentication Code das zu übertragende Paket größer wird. Dabei kann es passieren, dass die MTU überschritten wird und der Ethernet-Frame nicht übertragen werden kann.

In der Bachelorarbeit soll daher eine Lösung entwickelt und evaluiert werden, die eine transparente MACsec-Verschlüsselung erlaubt, d.h., es ist davon auszugehen, dass weder die auf der Leitung übertragbare Ethernet-Frame-Größe vergrößert werden kann, noch dass die Größe der gesichert zu übertragenen Ethernet-Frames verkleinert werden kann. Folglich ist eine Fragmentierung umzusetzen. Neben dem naiven Ansatz, aus jedem zu großen Ethernet-Frame zwei verschlüsselte MACsec Pakete zu erzeugen, sind auch effizientere Fragmentierungsverfahren zu untersuchen.

In einer Evaluation ist zu zeigen, dass die entwickelte Lösung das Erwartete aus funktionaler Sicht leistet, wobei gleichzeitig zu argumentieren ist, dass die Sicherheit des erweiterten MACsec-Protokolls nicht schlechter ist als die Sicherheit des originalen MACsec-Protokolls. Ferner ist zu untersuchen, welchen Einfluss die Fragmentierung auf Performanceparameter wie Verzögerungszeit und Bandbreite (übertragene Datenmenge) hat. Als Baseline ist eine nicht fragmentierte Übertragung mit Hilfe von Jumbo-Frames anzusetzen.

Für eine erfolgreiche Bearbeitung des Themas sind folgende Teilaufgaben zu erfüllen:

- Erweiterung des MACsec-Protokolls um transparente Fragmentierung von zu übertragenden Ethernet-Frames
- Implementierung dieses Protokolls im Linux-Kernel
- Evaluierung der entwickelten Lösung bezüglich Funktionalität, Sicherheit und Performance

Betreuer: Dr.-Ing. Stefan Köpsell

Verantwortlicher Hochschullehrer: Prof. Thorsten Strufe

Institut: Systemarchitektur

Beginn am: 08.01.2018 Einzureichen am: 26.03.2018

8.01.18 .....

Datum, Unterschrift der/des Studierenden

.....

Unterschrift des betreuenden Hochschullehrers

Verteiler: Prüfungsamt (Original); Student (Original – Deckblatt der Bachelorarbeit); HSL + Sekr. (Kopie); elektr. Daten → Sekretariat

Erklärung / Declaration

Hiermit erkläre ich, dass ich diese Arbeit selbstständig erstellt und keine anderen als die angegebenen Hilfsmittel benutzt habe.

I hereby declare that I completed this work on my own and have not used any resources other than those noted.

Dresden, March 26, 2018

Jan Sönke Huster

Contents

List of Figures

List of Tables

1 Introduction

As Industry 4.0 becomes more popular, networking in factories has gotten more important. However, current factories are insecure as a study of the german Federal Ministry of Economic Affairs and Energy regarding IT security in Industry 4.0 shows [Bac+16]. The FastVPN project[1] adresses this problem space.

It provides transparent network bridges—FastVPN boxes—, which are deployed between industry machines and the factory network. Thus the network communication is encapsulated, furthermore, network security inbetween the FastVPN boxes is improved. This environment is displayed in figure 1.1.

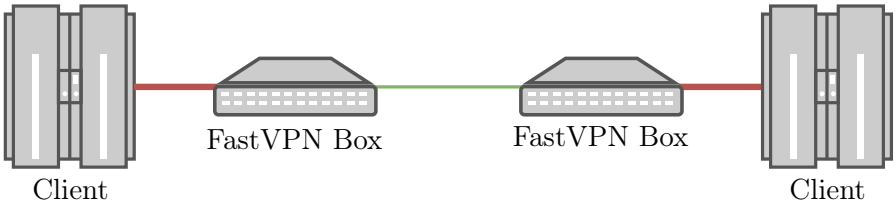


Figure 1.1: Insecure network communication is protected by FastVPN boxes. The is done by applying MACsec. The Maximum Transmission Unit between the boxes is smaller than the one between client and box.

A protocol used by the FastVPN box to improve network security is *Medium Access Control Security* (MACsec), a standard of the *Institute of Electrical and Electronics Engineers* (IEEE). MACsec provides confidentiality and integrity on the data link layer. It encrypts network communication and protects against unauthorized modification by appending a message authentication code. The downside of MACsec is the reduction of the maximum packet size—the *Maximum Transmission Unit* (MTU)— because additional information are added to each frame which is tunneled through the FastVPN box.

If machines cannot discover that the MTU is decreased on the path, the packets are oversized at one point. The standard behavior is to drop these oversized packets. As a result, machines without this discovery mechanism are not able to communicate through the FastVPN box.

A solution to this problem can be the use of jumbo frames, which allow higher MTU sizes than the standard of 1500 bytes. By enabling jumbo frames between the FastVPN boxes, the MTU can be increased so that the overhead introduced by MACsec is equalized. But to enable jumbo frames modern network hardware between the FastVPN boxes is required.

[1] `https://en.fast-zwanzig20.de/industrie/fast-vpn-2/`

The aim of the FastVPN project is the deployment in factories. Machines as well as network hardware in factories often have a long life span, so the FastVPN project cannot assume that the configuration of jumbo frames is feasible. Furthermore, it is not assumed that the machines are discovering the decreased MTU.

Therefore, it requires a solution to use MACsec without a reduction of the MTU and the configuration of jumbo frames. The aim of this thesis is to develop a transparent fragmentation solution in combination with MACsec to solve this problem. As the reduction of the MTU is comparatively small to the frame size, a simple split produces one full and one very small frame. Hence, possible optimizations are examined.

This thesis begins with an introduction into computer networks, security and MACsec. Moreover, protocols which support fragmentation and possible optimizations are examined. Afterwards, several approaches to implement fragmentation when using MACsec are proposed and discussed. The most promising solution is then specified in detail and implemented for the use with linux. Finally, the implementation of the proposed fragmentation scheme is evaluated regarding security and performance. Therefore, it is compared with the solution of using jumbo frames.

2 Related Work

2.1 Computer Networks

The following sections give a short introduction into computer network theory and provide an explanation to the terms used in this thesis. If not otherwise noticed, the information of this section is taken from [TW11].

OSI model The OSI model is a reference model for computer network architecture proposed by the *International Standards Organisation* (ISO) in [94].

It defines seven layers which are stacked on top of each other like displayed in figure 2.1. Every layer is just able to communicate with its neighbouring layers and performs a well defined function, such as layer 1 (physical layer) transmits bytes on a medium.

When an application transmits data to another host the data is traveling the path from the top layer (application layer) to the bottom layer, which is the physical layer. Each layer on this path performs its function on the data. The recipient receives the data and each layer processes it, from the lowest back to the top layer. Also the devices in between the network have a layered architecture and are processing the transmitted data in their layers. Those devices in the network which determine the path of transmitted data ("routing") usually have three or two layers as shown in figure 2.1.

Data received by a layer from the layer above is named *Service Data Unit* (SDU). The processed data, which a layer is passing down to the next layer, is called *Protocol Data Unit* (PDU). PDUs have different names, depending on the layer. On layer 2 they are named "frames" and on layer 3 they are named "packets". A generic term is "datagram" [MRM94].

Relevant layers for this thesis are the network layer (layer 3) and the data link layer (layer 2).

Layer 3 The network layer is determining the route of a packet and makes communication with other networks possible. Furthermore, it adapts the SDU to the MTU, which is the maximum size a data packet can have. Therefore, it breaks the SDU in several fragments and sends them as single packets. The receiving network layer is able to reassemble the fragments and passes them as a whole to the overlaying layer. This is possible because each layer is able to put more information into each SDU by transforming it or prepending and appending data. Usually a layer prepends a header with information on the SDU on transmission. When it receives a PDU to pass it to an upper layer, the information is processed and stripped. One of the usual protocols that work on layer 3 is the *Internet Protocol* (IP). The fragmentation process of IP is described in section 2.4.1.

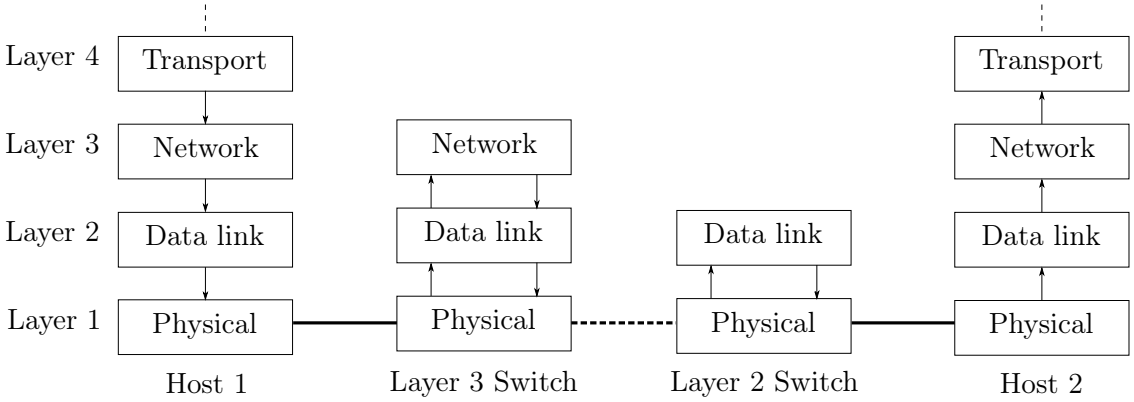


Figure 2.1: The first 4 layers of the *Open Systems Interconnection* (OSI) model. Host 1 sends a message to Host 2, one layer 3 and one layer 2 switch are involved.

Preamble	Destination	Source	Type	Payload	Pad	Checksum

Figure 2.2: Ethernet Protocol Data Unit

Layer 2 The data link layer communicates in the same subnetwork and consists of two sublayers. The Logical Link Control sublayer enables multiplexing. On shared channels the *Medium Access Control* (MAC) sublayer is checking if the channel is already in use and if a frame can be transmitted. Furthermore, it passes data to the physical layer. The SDU is put into a frame, source and destination addresses are added, as well as a checksum for error detection. In a cable-based *Local Area Network* (LAN) usually Ethernet is used at the data link layer. A LAN is a network which operates in a small area, such as one building.

Switched Ethernet Ethernet is standardized by the IEEE 802.3[2] working group. The terms "IEEE 802.3" and "Ethernet" are interchangeable. Classical Ethernet used a shared medium to transmit data, which means the physical cable was used by more than one participant at a time. This evolved to Switched Ethernet, here a dedicated cable per transmission channel is used.

Data is sent in frames and each frame has a preamble (7 bytes), source (6 byte), destination (6 byte) and ethertype (2 byte) prepended as shown in figure 2.2. The preamble is a constant value to synchronize the clocks of sender and receiver. Source and destination addresses are 6 bytes long and unique, so every participant can derive if he is the receiver of the frame. The ethertype field (2 bytes) is used to distinguish different protocols, so the receiver knows what type of data the payload is. Ethertypes can be requested by anyone at the IEEE. The Ethertype for MACsec is 0x88E5. Furthermore, a checksum (4 bytes) is appended, so the receiver is able to detect errors that happend on transmission. If an error is detected the frame is dropped, so Ethernet does not guarantee

[2] `http://www.ieee802.org/3/`

a successful transmission. An Ethernet frame has to have a minimum size of 64 bytes and a maximum size of 1522 bytes without the preamble, therefore, the payload can usually have a maximum size of 1500 bytes. If it has less than 46 bytes, padding has to be used.

Switches & Router A switch is a device which has a direct connection to several participants or other switches and forwards received frames to the destination. A layer 2 switch uses the information in the Ethernet frame header to determine the destination. As it works on layer 2, it can just forward packets in the same network. A layer 3 switch—or router—is able to route packets and uses the information of the IP to determine the destination. Therefore, it is able to route between different networks or fragment packets.

MTU The *Maximum Transmission Unit* (MTU) is the maximum size of a data unit to be transmitted. As prior mentioned, the Ethernet MTU is for example 1500 bytes, for Wireless LAN it is 2272 bytes [Gro16]. The protocol used in layer 3—which is in practice mostly IP—fragments the SDUs to match the MTU.

The Path MTU is the minimum MTU on the path a packet travels. To determine the Path MTU a process named *Path MTU Discovery* (PMTUD) is used, which is specified in [MD90].

Jumbo Frames Additionally, there is the possibility of raising the MTU up to 9000 bytes while using at least Gigabit Ethernet. Frames with a size higher than 1500 bytes are called Jumbo Frames. This is not standardized, but supported by most vendors of network interfaces.

Transmission Control Protocol *Transmission Control Protocol* (TCP) is a protocol which resides on layer 4—the transport layer—of the OSI model. It was first specified in RFC 793 [Pos81c]. It builds a connection over a network and provides a reliable transmission of packets. To achieve reliability, the receiver sends an acknowledgement for received packets. If these are not received by the sender after a certain time the packets are resend.

Internet Control Message Protocol The *Internet Control Message Protocol* (ICMP) is a companion protocol for IP. Its first specification was in [Pos81a]. If for example the size of an IP packet is bigger than the MTU, an ICMP message is sent to report this to the sender. Furthermore, it is used to send "echo" messages to check if a machine is reachable which then answers with an "echo reply" message. This can be used to measure the *Round-trip time* (RTT), which is the time from sending a packet until receiving the acknowledgement.

2.2 Security

The following sections introduce the computer security terms, which are needed for this thesis.

Security Goals There are three main security goals: confidentiality, integrity and availability [PP15]. Confidentiality means that information is kept private and just accessable by authorized entities. Integrity means that unauthorized modification of information is detectable and availability means that a service or data is available and usable [Pfi12]. These goals can be achieved through different techniques, confidentiality for example can be achieved by using encryption.

Symmetric Encryption Symmetric encryption describes the class of encryption algorithms, where the same secret is used for encryption and decryption. The communication partners that want to achieve confidentiality, therefore, they need a shared secret [Pfi12].

Message Authentication Code A message authentication code is a symmetric system to check if a message has been manipulated by an unauthorized entity. Message authentication codes help to achieve integrity. Authorized entities need a shared secret to create and verify a message authentication code [Pfi12].

Denial-of-Service A denial of service attack is an attack targeting the availability of data or a service. One example for a Denial-of-Service attack is the flooding of a network with unneccessary information to overload it [PP15].

Replay attack Another attack is the replay attack. Here the attacker delays or repeats valid messages between two parties to gain information [PP15].

2.3 IEEE 802.1AE MACsec

The Standard for *Medium Access Control Security* (MACsec) has been published by the IEEE in 2006 [Gro06]. MACsec is a protocol on layer 2 that provides confidentiality, integrity and data origin authenticity to mitigate network attacks. Furthermore, it prevents replay attacks.

2.3.1 Process

The entity which is operating MACsec is named *MAC Security Entity* (SecY). For the operation of MACsec cryptographic keys are needed because symmetric encryption and message authentication codes are used. The entity that provides these is the *MAC Security Key Agreement Entity* (KaY). How the KaY works is not specified in the MACsec standard but in IEEE 802.1X [Gro10]. It discovers other KaYs that are attached to the same LAN, mutually authenticates and authorizes them. Thereby secure relationships are

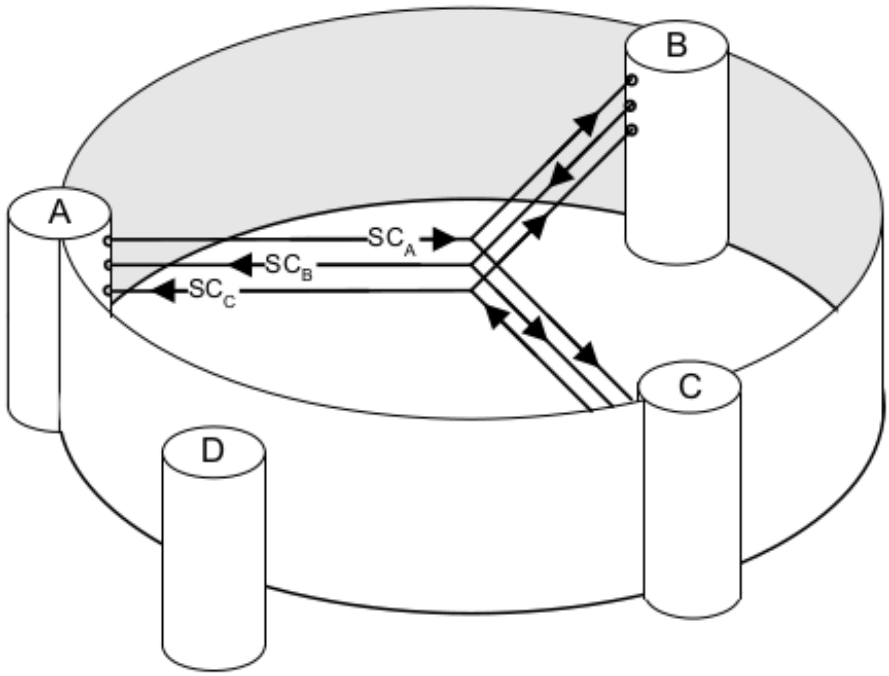


Figure 2.3: The three stations A, B and C form one Secure Connectivity Association. This CA is supported by three Secure Channels, SC_A, SC_B and SC_C. Station D has no secure connection to the other stations.
Figure from the MACsec standard [Gro06, p.25]

created, which the KaY maintains. The participants of these secure relationships form one *Secure Connectivity Association* (CA). A SecY can only be a member of one CA.

The CA is a group of SecYs. All members of a CA use the same cipher suite, which defaults to GCM-AES-128. With the amendment of 2011 GCM-AES-256 was added [Gro11]. The communication of the CA members takes place through *Secure Channels* (SCs).

A SC is a channel from one to many members within a CA. SCs support the secure frame transmission using symmetric cryptography. To distinguish different SCs every SC has an *Secure Channel Identifier* (SCI). A SC is supported by successive *Secure Associations* (SAs).

Each SA uses an individual *Secure Association Key* (SAK), which is provided by the KaY. To replace the cryptographic keys on a regular basis, the SecY can replace a SA with its successor. To identify the used SA the SCI and an *Association Number* (AN) are used.

Figure 2.3 displays four stations which are attached to the same LAN. Three of them build a CA, which was established by KaYs. For secure communication three SCs are build. The cryptographic key which is used on each SC depends on the current SA.

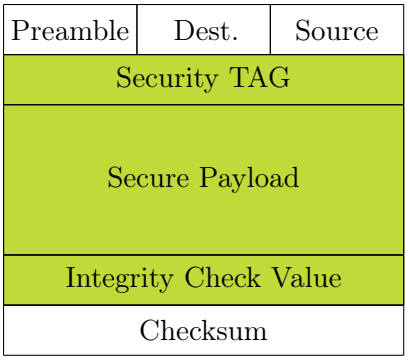


Figure 2.4: MACsec Protocol Data Unit (green) encapsulated in Ethernet frame.

2.3.2 MACsec Protocol Data Unit

MACsec works on a frame-by-frame basis. To provide connectionless confidentiality, integrity, origin authenticity and to mitigate replay attacks, the Ethernet frame is modified. The entities using MACsec have to be attached on the same LAN as it works on layer 2.

Furthermore, an *Integrity Check Value* (ICV) is appended, to detect modifications and, therefore, protect the integrity of the *Security TAG* (SecTAG) and payload.

The structure of a MACsec frame is shown in figure 2.4. As it is a modified Ethernet frame just the new fields are discussed, which are:

- Security TAG (8 or 16 bytes)
- Secure Payload
- Integrity Check Value (8 or 16 bytes)

Security TAG

The SecTAG is displayed in 2.5 and comprises of

- MACsec Ethertype (2 bytes constant value 0x88E5)
- *TAG Control Information* (TCI) (6 bit)
- *Association Number* (AN) (2 bit)
- *Short Length* (SL) (1 byte)
- *Packet Number* (PN) (4 byte)
- *Secure Channel Identifier* (SCI) (8 byte, optional)

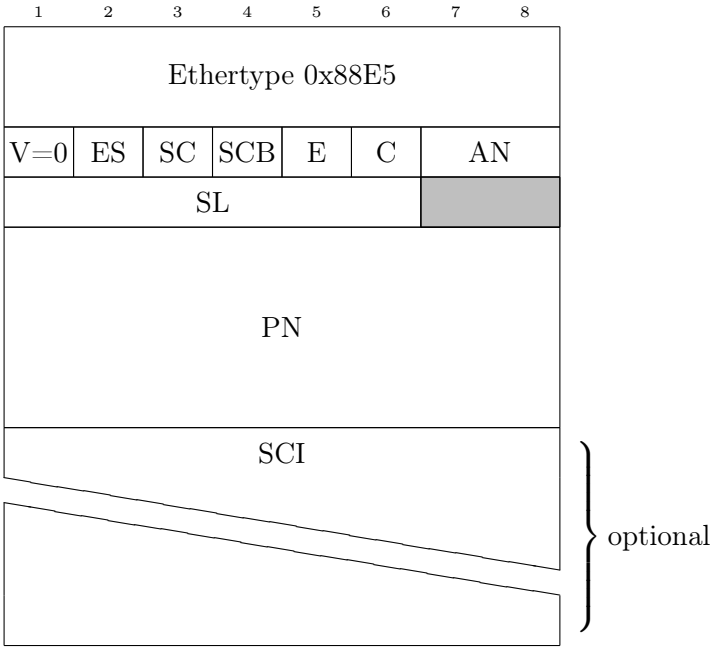


Figure 2.5: SecTAG of an MACsec Protocol Data Unit.

The *TAG Control Information* (TCI) field comprises of the following bits:

- Bit 1: Version (0 for [Gro06])
- Bit 2: MAC Source Address parameter bit (set if the first 6 bytes of the source address are equal to SCI)
- Bit 3: Explicit SCI bit (set if SCI is encoded)
- Bit 4: Usage of EPON Single Copy Broadcast without explicit SCI bit
- Bit 5: Encryption bit (set if secure payload is encrypted)
- Bit 6: Changed-Text bit (set if secure payload is not equal to payload of SDU)

The two bit AN is needed to identify different SA, which needs to be known use the correct cryptographic key.

If the length of the secure payload is shorter than 48 bytes, the *Short Length* (SL) field describes the length. Otherwise it is null. This is needed because frames that undercut the minimum ethernet frame size are padded. If this happens, the receiver needs the length of the secure payload to locate the ICV. The seventh and eighth bit are currently not used and always zero.

To provide protection from replay attacks the *Packet Number* (PN) is an incrementing value for each frame secured with MACsec. Moreover, it is part of the *Initialization Vector*

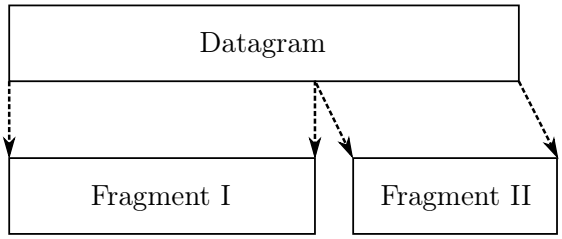


Figure 2.6: A datagram is fragmented into two fragments

(IV) for the Cipher Suite, which is needed for confidentiality and integrity, concatenated with the SCI.

The SCI is used to identify the SA in combination with the AN. Moreover, it is needed to identify the SecY which sent the frame in the network. If the SecY uses a point-to-point connection the use of the SCI is not necessary.

Secure Payload

The secure payload field contains the unencrypted or encrypted data of the SDU, depending on the configuration of MACsec. It's integrity is always protected, independent of whether encryption is configured or not.

Integrity Check Value

The ICV is a message authentication code, therefore, it is used to verify the integrity of the whole frame. It can be 8 bytes and 16 bytes long, this depends on the used cipher suite. The calculation of the ICV includes the source- and destination address, the SecTAG and the secure payload.

2.4 Fragmentation in other Protocols

In the context of computer networking fragmentation happens when data units are too large to be sent through the network. If this happens, the datagram is split into several fragments as shown in figure 2.6. These are transmitted independently. The receiver rebuilds the datagram out of the received fragments, this process is called reassembly.

In the following sections, three different protocols are examined regarding fragmentation. The IP is widespread and realizes fragmentation on layer 3. *Wireless LAN* (WLAN) is a common layer 2 protocol, which implements fragmentation. Furthermore, *Long Term Evolution* (LTE) realizes fragmentation and concatenation on layer 2. Concatenation is the process of appending several datagrams to one joint datagram.

2.4.1 Internet Protocol Fragmentation

The IP fragments internet datagrams into packets matching the path MTU as already mentioned in section 2.1. The protocol is specified in [Pos81b].

Three fields of the IP header are relevant for the fragmentation and reassembly process: the *More Fragments* (MF) bit, the Fragment-Offset field and the Identification field.

The Identification field is two bytes long and unique for the source-destination pair. It is determined by the originating protocol that sends internet datagrams and is used to identify the fragmented datagram. A MF bit indicates that the packet is not the last fragment of the fragmented internet datagram. The fragments position in the datagram is described by a 13 bit long Fragment-Offset field. It indicates the number of preceding 8 bytes long blocks. So a Fragment-Offset field with the value of one means that the preceding data is 8 bytes long. This field allows to reassemble fragments even if they do not arrive in the transmitted order.

Regarding figure 2.6 the first packet—carrying Fragment I in its payload—has the MF bit set and the Fragment-Offset is zero. The second packet—carrying Fragment II in its payload—has the MF bit set to zero and the Fragment-Offset is set to the position of this fragment in the datagram. The Identification field is the same for both packets, as both contain fragments of the same datagram.

The detailed process of the IP fragmentation and reassembly process is described in [Pos81b, section 2.3]

2.4.2 IEEE 802.11 Wireless LAN

WLAN is a standard for "wireless connectivity [...] within a local area" [Gro16, sec. 1.1]. The standard defines several physical layers and a MAC layer which contains fragmentation. It is specified by the IEEE in [Gro16].

Due to the fact that WLAN uses a shared medium, collision probability and the probability of interferences is higher compared to Ethernet. Fragmentation can be used to improve the reliability of WLAN [TW11, sec. 4.4.3].

Similar to fragmentation in IP, which is described prior to this section, the header of a WLAN frame has a MF bit. In addition to that, a Sequence Control field exists, which comprises of a Fragment number field and a Sequence number field. The Sequence number field identifies the datagram that is fragmented; the Fragment number field identifies the individual fragments. The first fragment has the Fragment number field set to zero. It is incremented with every following fragment. The Sequence number field is the same for every fragment that is part of the same datagram. It is comparable to the Identification field of IP.

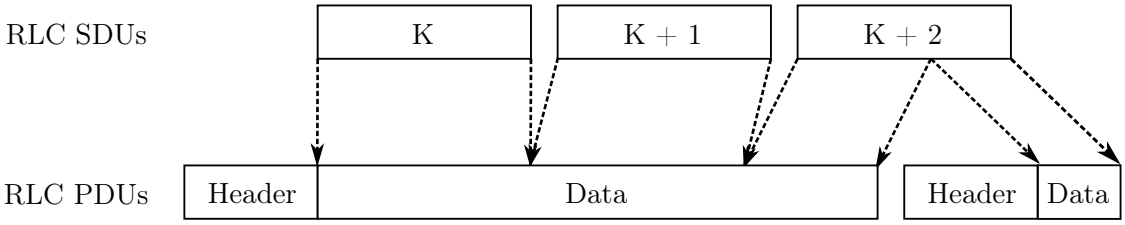


Figure 2.7: Three *Radio Link Control* (RLC) SDUs are transformed into two RLC PDUs. K, K+1 and the first fragment of K+2 are concatenated, K+2 is additionally fragmented.[3]

2.4.3 Long Term Evolution

The LTE standard is defined by the 3GPP—"a collaboration between telecommunications associations" [TW11]—in various specifications [3GP06]. It is a wireless comunication technology used for mobile devices that has fragmentation and concatenation implemented. This means that—additionally to fragmentation—the payload of several datagrams can be concatenated into one frame. Fragmentation in the LTE standard is named segmentation, but this thesis continues the usage of the term fragmentation to avoid confusion.

The process of fragmentation and concatenation in LTE is shown in figure 2.7. The RLC layer, which is defined in [3GP17], is responsible for fragmentation and concatenation in LTE. This layer lays between the physical layer and the layer which includes IP, like layer 2 of the OSI model.

The RLC layer in LTE has different modes but to understand fragmentation and concatenation in LTE this distinction is not relevant and, therefore, not deepened here. In the following the header of the *Unacknowledged Mode* (UM) PDU is described to understand how fragmentation and concatenation work in general in LTE.

Unacknowledged Mode Data PDU

The header of an *Unacknowledged Mode Data* (UMD) PDU comprises of two parts: a fixed and an optional extension part. Furthermore, the PDU has one data field which can contain several SDUs as shown in figure 2.7.

Fixed part The fixed part of the header is displayed in figure 2.8. It contains a *Framing Info* (FI) field, an *Extension* (E) bit and a *Sequence Number* (SN) field.

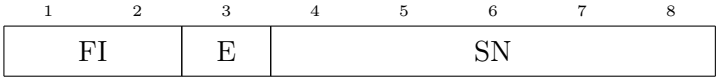


Figure 2.8: The fixed part of an UMD header.

The FI field shows, whether the first and last RLC SDUs are fragmented or not. The first bit is set, if the first byte of the data field is not equal to the first byte of a RLC

[3] Figure from `http://www.tutorialspoint.com/lte/lte_layers_data_flow.htm`

SDU. The second bit is set, if the last byte of the data field is not equal to the last byte of a RLC SDU.

For example, the FI field of the left RLC PDU of figure 2.7 is 01 because the first SDU is not fragmented, but the last.

If the E bit of the header is set, an extension part is following. Otherwise the data field follows.

The SN field is the identifier of an UMD PDU and is incremented for the following PDU. This is necessary to reassemble the correct SDU fragments together.

Extension part If more than one data field is present—this means concatenation is used—one or more extension parts exist in the PDU header. This extension part, displayed in figure 2.9, contains an E bit and a *Length Indicator* (LI) field. The E bit indicates, whether another extension part or the data field follows. The LI field contains the length of the associated data field.

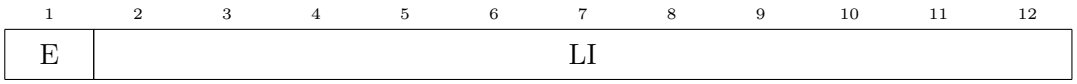


Figure 2.9: The extension part of an UMD header.

If concatenation happens, for every—except the first—concatenated SDU an extension part is added to the header. This means that for k concatenated SDUs the header comprises of one fixed part and $k-1$ extension parts. The nth extension part is associated with the $n+1$th SDU. For example, in figure 2.7 the header of the left RLC PDU has one fixed part and two extension parts.

The recipient is able to reassemble the fragmented and concatenated SDUs with the header information. With the information of the SN field, the order can be reconstructed. By reading the extension header, the PDU can be split up into the different SDUs. The FI field describes, wether these SDUs are fragments or not, if so the related fragments are concatenated.

3 Fragmentation and Concatenation Approaches

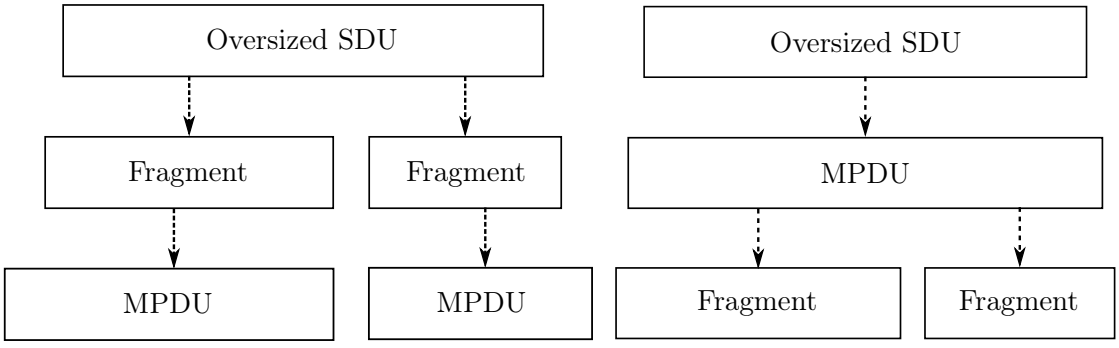
The problem described in the introduction arises when using MACsec and the configuration of Jumbo Frames is not possible. But it can also appear in different scenarios with different protocols. It occurs when an SDU should be processed which is bigger than the MTU, when using MACsec bigger than 1468 bytes. This can happen because the sender is not aware that the MTU undercuts the standard Ethernet MTU of 1500 byte. Following the current MACsec standard it cannot be processed and is dropped [Gro06].

When thinking about a transparent fragmentation and concatenation protocol to solve this problem, there is the decision to make whether fragmentation should happen before or after applying MACsec. This means, whether the SDU is fragmented or the *MACsec Protocol Data Unit* (MPDU) is fragmented.

In the following paragraphs, both approaches are drafted, evaluated and discussed.

3.1 Fragmentation Approaches

In figure 3.1 the process of fragmenting the SDU and applying MACsec on each fragment (3.1a) and the process of first applying MACsec on the SDU and fragmenting the MPDU (3.1b) are compared.



(a) SDU fragmentation (b) MPDU fragmentation

Figure 3.1: SDU (3.1a) and MPDU (3.1b) fragmentation approaches

3.1.1 Fragmenting the SDU

The oversized SDU is fragmented into two fragments, each smaller or equal to the MACsec MTU of 1468 bytes. After that, MACsec is applied to both fragments. The indication of fragmentation can work similar to the IP protocol or WLAN, using a *More Fragments* (MF) bit in the MACsec header.

When the destination receives the first frame, it verifies the MACsec frame. After that the destination checks for the MF bit. If it is set, the destination waits for the next MACsec frame to arrive and assembles both secured payloads after successfully verifying the second MACsec frame.

Without a field to identify the fragmented SDU, the protocol relies on the consecutive transmission of both fragments. Otherwise, two payloads which do not belong together are assembled. But as MACsec works on Point-to-Point connections, frames can not overtake another and the consecutive transmission can be guaranteed by the sender.

If the—of the undercut MTU unaware—sender sends a frame with a size of 1500 byte, after applying MACsec the first fragment has a size of 1500 bytes. The second one then has a size of 64 bytes. The overall data sent is 1564 bytes.

3.1.2 Fragmenting the MPDU

Using this approach MACsec is applied to the oversized SDU. After that, the MACsec frame is fragmented and both fragments are sent to its destination. When the destination receives the first fragment it waits for the second one. After receiving the second fragment both are assembled and verified by MACsec. The fragmentation can be indicated with a MF bit as in the previous approach.

If the—of the undercut MTU unaware—sender sends a frame with a size of 1500 byte, the MPDU has a size of 1532 byte. The first fragment then has a size of 1500 bytes, the second one of 32 bytes. Considering the minimum size of an ethernet frame, the second fragment has to be padded to a length of 46 bytes. The overall data sent is 1546 bytes.

3.2 Concatenation Approaches

If the first fragment has the maximum size of 1500 byte, the second one is much smaller, regardless which of both approaches is used for fragmentation. In the MPDU approach it is even to small to be sent without padding. To utilize the maximum space in the second fragment, concatenation like in the LTE protocol can be implemented as optimization. Therefore, the sender checks the network transmission buffer for remaining SDUs and concatenates SDUs with the same destination. The frame header has to specify the end of a segment in the payload, so that the destination is able to split the segments. Furthermore, it has to indicate whether the last segment is fragmented.

For both approaches, a drafted concatenation design is described in the following paragraphs.

Assuming that concatenation creates a higher latency, its usage should be optional.

3.2.1 Concatenation of SDUs

In this approach, concatentation must be managed inside the MPDU header.

As an arbitrary number of SDUs can be concatenated, the header must have a variable size. Moreover, for every except one SDU the length must be present. To represent the maximum length of a SDU, considering a MTU of 1500 bytes, 11 bits are needed.

The SecTAG needs to carry several additonal information. It must indicate, if concatenation is used and present the length of all but one concatenated SDUs. Furthermore, the use of fragmentation must be indicated. To have a variable header, depending on the number of concatenated SDUs, an extension header can be used like in the LTE, which is explained in section 2.4.3.

Therefore, a modification of the SecTAG is proposed as shown in figure 3.2. The unmodified SecTAG has two reserved bits that follow the short length field, which are used. These bits are replaced by:

- *More Fragments* (MF) bit
- *Extension* (E) bit

The MF bit indicates, whether the last SDU is fragmented or not. The E bit indicates, whether an extension header follows.

An extension header comprises of:

- Segment length field (11 bit)
- Reserved field (4 bit), to achieve a byte-aligned header
- Extension bit

For every except the first SDU an extension header is present. Extension headers are appended to the SecTAG. The segment length field is 11 bit long and contains the length of the corresponding SDU. The first extension header corresponds to the second SDU. As—according to the IEEE MACsec standard [Gro06]—the header must be byte-aligned, it contains 4 reserved bits, which follow the segment length. These are followed by the E bit, which indicates whether another extension header follows.

When the destination receives a PDU, it verifies the MACsec frame and checks the first E bit. If it is set, it splits the segments according to the extension headers. Afterwards, it checks the MF bit and if applicable buffers the last, fragmented SDU. If the buffer contains a fragment, the first SDU is appended to it.

As the SecTAG is protected against modifications by the ICV, the security should not weaken. Furthermore, it must be ensured that all concatenated SDUs belong to the same *Secure Channel* (SC).

3.2.2 Concatenation of MPDUs

Concatenation for this approach would use a header similar to the extension header of the SDU approach which is described in figure 3.2.

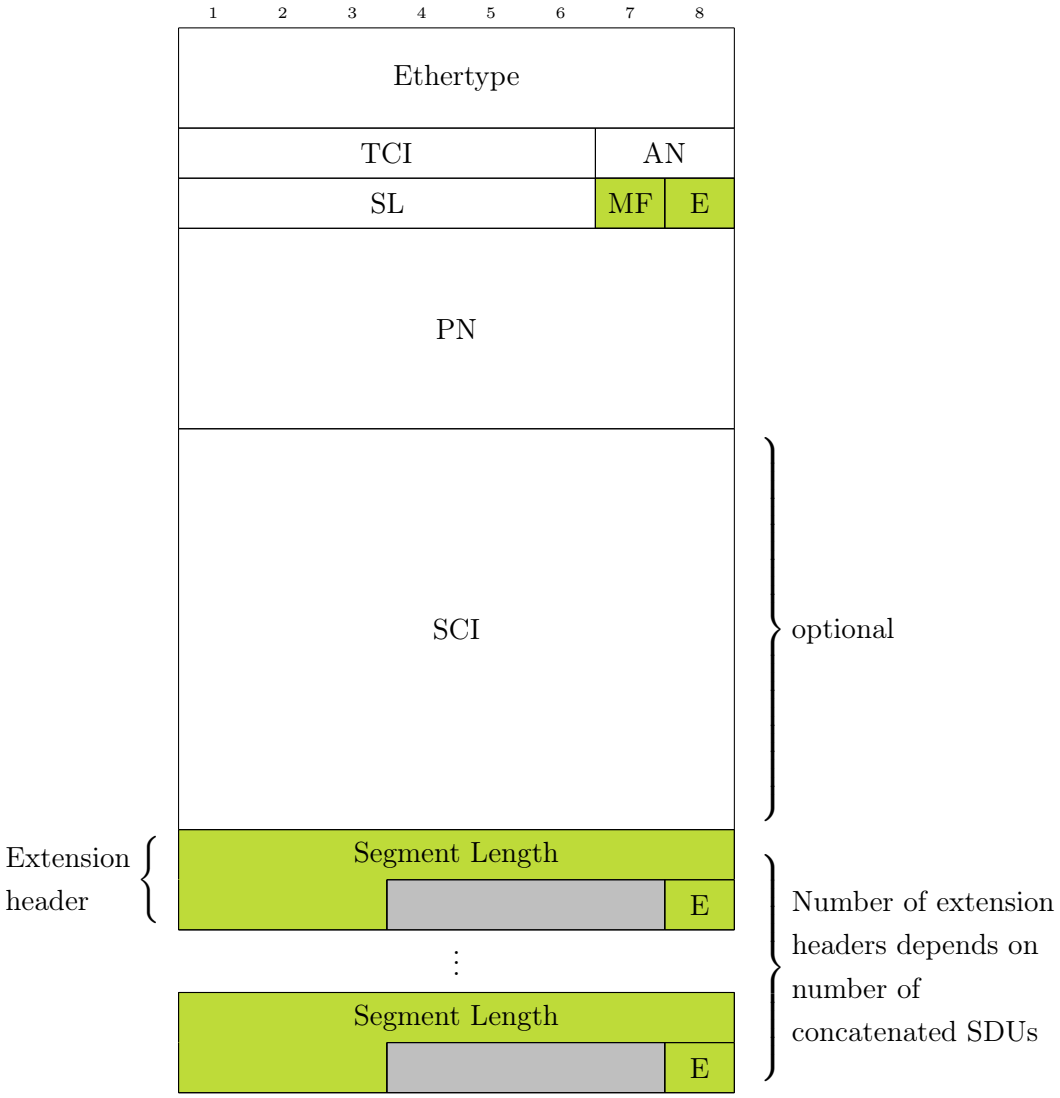


Figure 3.2: Draft for modified SecTAG to support fragmentation and concatenation. New fields are highlighted green.

MACsec is applied to every SDU. After that, the MPDUs are concatenated together, using the extension header described in section 3.2.1, and sent in one frame. Furthermore, the last extension header must additionally contain a MF bit to support fragmentation.

The destination reads all extension headers and splits all MPDUs. After that, it verifies all MACsec frames.

3.3 Evaluation

To evaluate both approaches, this thesis assumes the two cases of "mouse flows" and "elephant flows" as introduced by L. Guo and I. Matta in [GM01]. An elephant flow describes a large data flow, a mice flow a small one.

For the elephant flow this thesis assumes that the network buffer is full with SDUs with the size of the Ethernet MTU of 1500 bytes. In the case of the mouse flow it is assumed that one SDU is sent without the necessity of fragmentation or concatenation.

This thesis compares:

1. Data overhead
2. Cryptographic overhead
3. Latency
4. Security

3.3.1 Data overhead

As baseline to compare to, MACsec with Jumbo Frames is used, which results in 32 bytes overhead for each case. This solution is considered to be optimal because the MACsec header must not be increased but it is not applicable for the given scenario as explained in the introduction.

To calculate the overhead for each approach a function is developed which has the number of SDUs that should be sent as input and outputs the total size of the sent headers. After that the gradients of the functions are compared.

Jumbo Frames

The jumbo frame solution creates an overhead of 32 bytes per SDU, regardless of whether it is a mice or an elephant flow.

To the best of the authors knowledge there are no layer 2 concatenation solutions. Therefore, this thesis uses normal MACsec as baseline for concatenation solutions. The header size is 32 bytes per SDU.

SDU fragmentation and concatenation

For this approach the header size comprises of 32 bytes per frame and additionally two bytes for every concatenated segment from the second segment onwards.

Elephants The SDUs have the maximum size, so there are never more than two data segments in one frame. Assuming every frame contains two data segments, the header for this approach is 34 bytes per frame, therefore, the MACsec MTU decreases to 1466 bytes.

$f(n)$ in 3.1 calculates the total size of headers for n SDUs using the approach of fragmenting SDUs.

$$\text{SDU} = 1500 \text{ byte} \quad \text{MTU} = 1466 \text{ byte} \quad \text{MACsec} = 34 \text{ byte}$$

$$f(n) = \underbrace{n \cdot \frac{\text{SDU}}{\text{MTU}}}_{\text{number of frames}} \cdot \text{MACsec} \qquad n \in \mathbb{N} \tag{3.1}$$

The number of frames that have to be sent is the total payload of all SDUs divided by the MTU of the approach. Multiplied with the header size it is the total size of sent headers.

$$f(n) = n \cdot \frac{1500}{1466} \cdot \ 34 \text{ byte} \qquad n \in \mathbb{N} \tag{3.2}$$

$$= 34.79 \text{ byte} \cdot n \tag{3.3}$$

As n is the number of SDUs of a size of 1500 bytes, the average overhead per SDU is 34.79 bytes.

$$\frac{34.79}{32} \cdot 100\% = 108.71\% \tag{3.4}$$

As step 3.4 of the calculation shows, compared with the overhead of the jumbo frame solution, the overhead of this approach is 8.71% bigger.

Mice As the mice case is an SDU which is not fragmented, the overhead is the 32 bytes long header, which is the same overhead a jumbo frame solution has.

MPDU fragmentation and concatenation

For the MPDU approach there is a 2 bytes long header for every except one concatenated segment and a 32 bytes MACsec header per SDU.

Elephants In this case there is a 2 bytes long header for each frame because there are never more than two SDUs in one frame for the elephant case. Additionally, one MACsec header per SDU is added.

$$\text{MPDU} = 1532 \text{ byte} \quad \text{MTU} = 1498 \text{ byte} \quad \text{Header} = 2 \text{ byte} \quad \text{MACsec} = 32 \text{ byte}$$

	Header size in byte	
Approach	Elephant flow	Mice flow
Jumbo Frames	32.00	32.00
SDU	34.79	32.00
MPDU	34.05	32.00

Table 3.1: A comparison of the average resulting header sizes using different approaches for fragmenting and concatenating oversized SDUs while using MACsec

To calculate the transmitted header size, $g(n)$ in 3.5 adds the size of all fragmentation headers—2 bytes per frame—with the size of all MACsec headers.

$$g(n) = \underbrace{n \cdot \frac{\text{MPDU}}{\text{MTU}}}_{\text{number of frames}} \cdot \text{Header} + n \cdot \text{MACsec} \qquad n \in \mathbb{N} \tag{3.5}$$

$$g(n) = n \cdot \frac{1532}{1498} \cdot 2 \text{ byte} + n \cdot 32 \text{ byte} \qquad n \in \mathbb{N} \tag{3.6}$$

$$= 34.05 \text{ byte} \cdot n \tag{3.7}$$

$$\frac{34.05}{32} \cdot 100\% = 106.41\% \tag{3.8}$$

Step 3.8 shows that, compared with the jumbo frame solution, this approach creates an overhead of 6.41%.

Mice In case of a mice flow the MPDU has a size where it is not fragmented and the network stack is empty, so concatenation is not applicable. This approach would then send an unmodified MACsec frame. The overhead is one 32 bytes MACsec header, which is compared to the jumbo frame solution 6.25% higher.

Comparison

A comparison of the average header sizes of the different approaches per processed SDU is displayed in table 3.1. In the mice case, all approaches create the same header size of 32 byte. In the elephant case, the MPDU approach generates a smaller header compared to the SDU approach. The jumbo frame solution is not usable in the given setup, but would be the best of these approaches regarding data overhead.

3.3.2 Cryptographic overhead

When using MACsec two cryptographic processes are done: The payload is encrypted and the ICV is calculated, which includes the header and payload.

For the encryption part there arises no overhead for both approaches, as length of the encrypted payload is the same compared to usual MACsec.

For the MPDU approach the number of cryptographical operations remain unchanged when calculating the ICV because the MPDUs is created as usual.

If fragmentation has to be applied, with the SDU approach the cryptographical operations for the ICV increase because a second header has to be processed.

Regarding concatenation, the cryptographical operations for calculating the ICV decrease when using the first approach, because several SDUs are concatenated and share the same MACsec header. For the second approach this does not apply, as every SDU has still one MACsec header.

As the cryptographic operations remain the same for the approach of fragmenting MPDU and minimal increases or decreases for the other, the cryptographic overhead is considered as negligible.

3.3.3 Latency

The latency is assumed to increase compared to jumbo frames because in both approaches the data is sent in two instead of one frame, so the receipient has to wait for the second frame to arrive before processing the data. But as there is to the best of the authors knowledge no other solution, this is considered as negligible.

Concatenation can be an issue for latency, as it has to check the network stack for other SDUs, but as this is optional it is not considered here.

3.3.4 Security

MACsec secures against a wide range of network attacks. In the following section both approaches are analyzed regarding a decrease of security provided by MACsec.

SDU fragmentation and concatenation

As this approach sends modified MACsec frames, it has to be evaluated if the modifications influence the security. The new header fields are inside the SecTAG which is protected by the ICV. Therefore, the new header fields are also protected.

MPDU fragmentation and concatenation

This approach fragments and concatenates unmodified MPDUs which are conventionally processed by MACsec. The additional segment field header is not protected because it is not part of the MPDU.

Furthermore, the source and destination fields of the ethernet frame are not longer protected by the ICV. Thereby an unauthenticated attacker is able to sent frames with

Approach	Data overhead		Cryptograpy	Latency	Security
	Elephant flow	Mouse flow			
SDU	108.71%	100%	→	↘	→
MPDU	106.41%	100%	→	↘	↘

Table 3.2: Comparison of the SDU and MPDU approach

the More-Fragments bit set and a spoofed source address. As this is not protected the destination would interpret the following valid and authenticated frame as the second fragment. After concatenation the ICV would be invalid, the unauthorized as well as the authorized frame would be dropped. This decreases the availability for authenticated users and is a security issue.

Comparison

The first approach provides the same security as regular MACsec. The second approach rapidly decreases the security provided by MACsec.

3.3.5 Outcome

The previous sections compared the approach of fragmenting and concatenating SDUs to the approach of fragmenting and concatenating MPDUs regarding data and cryptographic overhead as well as security and latency. The results are displayed in table 3.2.

The approach of processing MPDUs is insecure as shown in section 3.3.4. This thesis proposes an enhancement for MACsec, which secures layer 2 traffic. A solution which weakens the security that MACsec provides is not contemplable.

Therefore, this thesis focuses on the improvment, implementation and evaluation of the approach of fragmenting and concatenating SDUs.

4 Proposed Solution

Fragmenting SDUs solves the MACsec MTU problem and is superior compared to MPDU fragmentation as discussed in the previous chapter. Therefore, this chapter specifies the proposed modifications of MACsec to realize transparent SDU fragmentation and concatenation. Afterwards, the implementation is described.

4.1 Specification

4.1.1 Transmission Process

The following transmission flow is displayed as flowchart in figure 4.1 for a better understandability.

When a frame has to be sent using MACsec it must be checked at the beginning whether fragmentation has to be applied. This is the case if the size of the SDU exceeds the maximum payload size of MACsec regarding the MTU of the underneath layer. If the standard MTU of 1500 bytes is set, the maximum payload size is 1468 bytes.

The fragmentation process must happen before applying MACsec. Therefore, the SDU must be split. The first fragment must be sent with the *More Fragments* (MF) bit set. It must be ensured that the second fragment is processed next by MACsec.

If the size of the SDU undercuts the maximum payload size of MACsec, concatenation must be applied if enabled. Therefore, the queue of MACsec has to be checked for additional SDUs that belong to the same *Secure Channel* (SC). If it contains SDUs, they are dequeued and appended to the payload until the maximum payload size is matched or exceeded. Source and Destination addresses of these SDUs are cut. It must be considered that for each additional SDU the maximum payload is decreased by two bytes because an extension header must be added. The last SDU may be fragmented if otherwise the maximum payload size is exceeded. The lengths of the additional SDUs are saved for the creation of the SecTAG.

When creating the SecTAG for each additional SDU an extension header is appended to the SecTAG. The extension header contains the size of the corresponding SDU. The nth extension header corresponds to the $n + 1$th SDU. The Extension bit is set for all but the last extension headers.

After the fragmentation and concatenation process, MACsec is applied as usual.

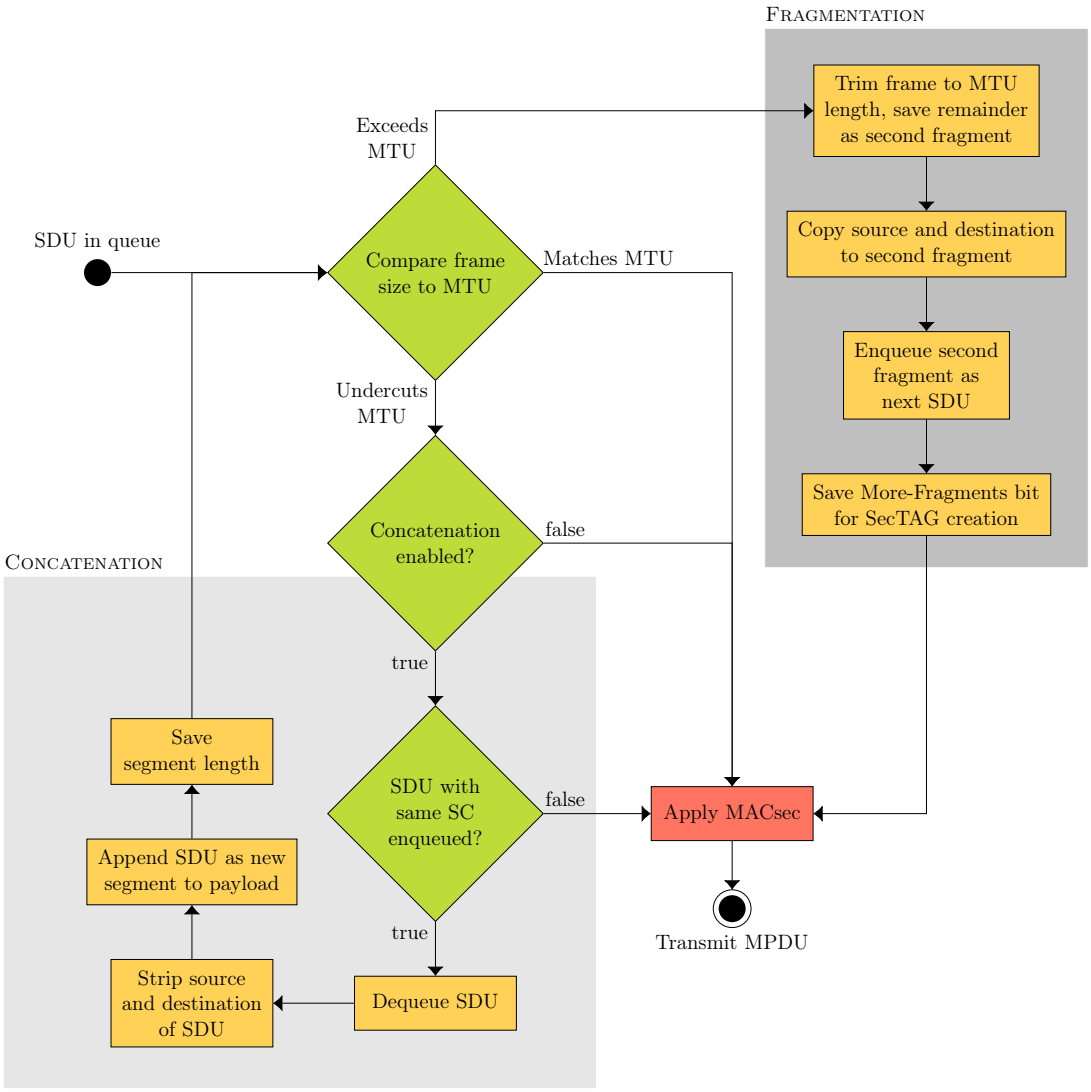


Figure 4.1: Transmission flowchart for proposed modifications

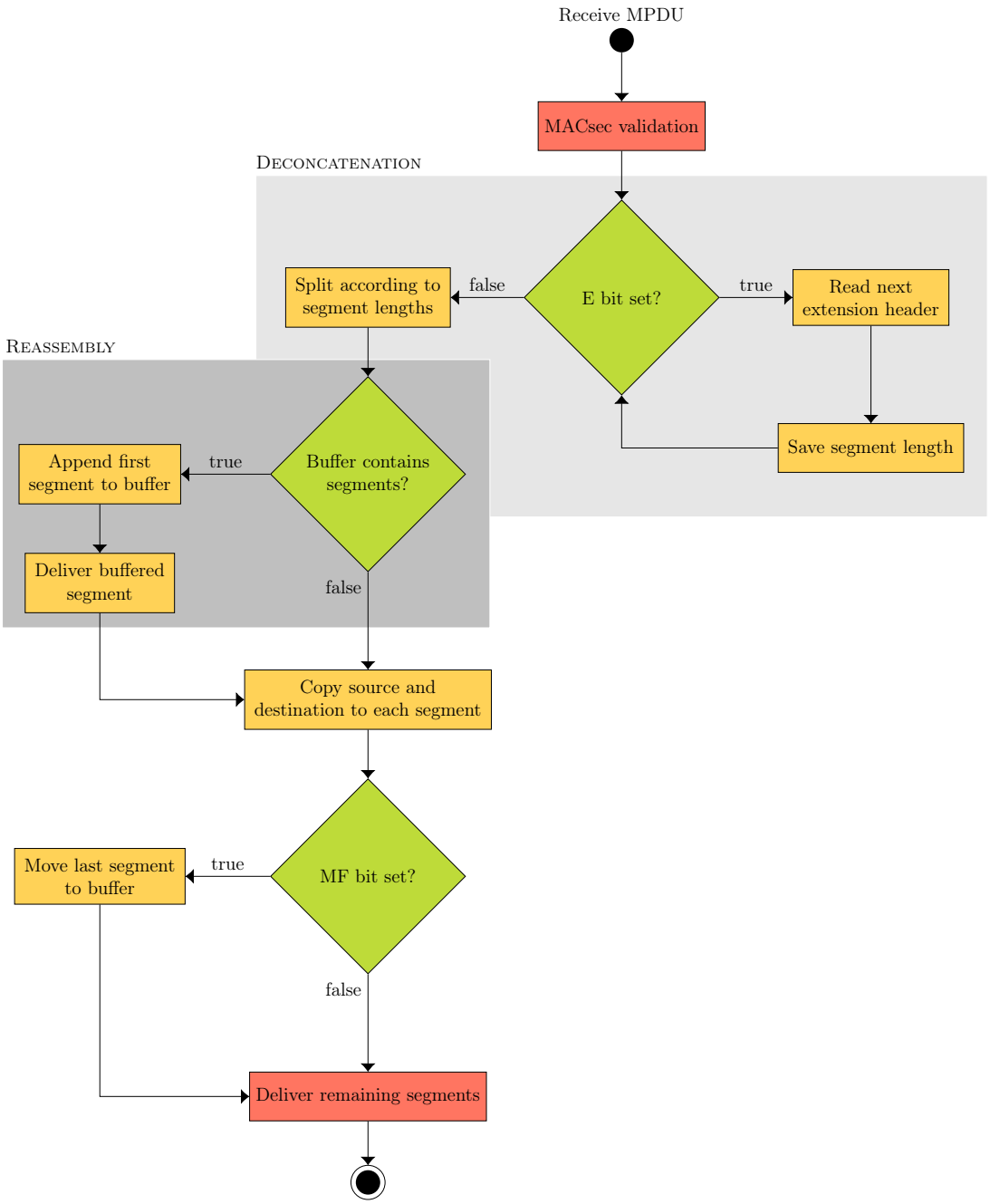


Figure 4.2: Receive flowchart for proposed modifications

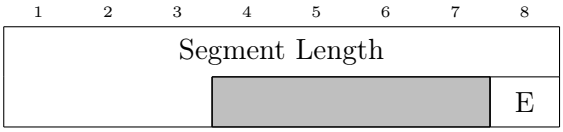


Figure 4.3: Extension header of modified MACsec

4.1.2 Receiving Process

The following receive flow is displayed as flowchart in figure 4.2.

When a frame is received it must be validated as specified by the original MACsec standard. Except to this validation is the check if both reserved bits of the SL field are set as these are now the E and MF bit.

If the validation is successful, it is checked whether the E bit is set. If the E bit is set, all extension headers are processed and the payload is cut into several SDUs. The length of each one is derived by the extension headers. The Source- and Destination addresses are prepended to each SDU. If the E bit is not set, the payload contains only one SDU.

If the fragmentation buffer for the current SC contains any data, the first SDU must be prepended to this data and the whole, new SDU must be delivered. If the MF bit is set, the last SDU has to be saved into the buffer for the current SC.

Finally, all remaining SDUs are delivered.

4.1.3 MACsec Protocol Data Unit

The modified MPDU is displayed in figure 3.2. The first unused bit after the Short Length field is the MF bit. This bit indicates whether the payload is fragmented and the next frame contains the next fragment. The second unused bit after the Short Length field is the E bit. This bit indicates whether an extension header is appended to the SecTAG.

The extension header comprises of an 11 bit long *Segment Length* (SEGL) field and an E bit. These are seperated by 4 unused bits. The *Segment Length* (SEGL) field contains the length of the corresponding SDU. The nth extension header corresponds to the $n + 1$th SDU. If the E bit of an extension header is set, another extension header follows. Otherwise, the payload follows.

4.2 Implementation

In 2016 the MACsec kernel driver implemented by Sabrina Dubroca was merged into the linux kernel[4]. The design specified in the previous chapter is implemented based on the kernel version 4.15.6[5].

In [Dub16] Dubroca describes the implementation details of the MACsec driver. MACsec is implemented as a virtual network device. There are two important entry

[4] `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=c09440f7dcb304002dfced8c0fea289eb25f2da0`

[5] `https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable-rc.git/commit/?h=v4.15.6&id=1a7aef62b47b00630e62a268d647f54ec93fb38c`

```
struct macsec_eth_header {
    struct ethhdr eth;
    u8  tci_an;
    u8  short_length:6
        more_fragments:1
        extended:1;
    __be32 packet_number;
    u8 secure_channel_id[8];
} __packed;

```

Figure 4.4: MACsec header struct

points regarding the proposed changes. When a new frame is sent via the MACsec device, the function `macsec_start_xmit` is executed with the frame as parameter. When a frame is received by the MACsec device, the `rx_handler macsec_handle_frame` function is executed.

Frames in the linux kernel are represented by `struct sk_buff`, which is an abbreviation of socket buffer. This struct contains multiple pointers to the actual data.

The MACsec frame header is represented by the `struct macsec_eth_header`. To reflect the proposed changes of the MPDU, it now contains the More Fragments and Extension bits as well as an arbitrary number of extension headers as shown in the implementation excerpt 4.4.

The MTU of the MACsec device is calculated by subtracting the size of the SecTAG and ICV from the MTU of the underlaying network device. To reflect the increased MTU, a constant— `#define ADDITIONAL_MTU 1468` —is introduced and added to this value.

4.2.1 Fragmentation

The fragmentation is implemented in the transmission handler function `macsec_start_xmit`. Therefore, the size of the socket buffer is checked. If its current size added to the additional MACsec size is larger than the MTU of the underlaying device a fragment has to be created. Therefore, a second socket buffer is allocated. The ethernet header is copied and the payload is split up between both frames. After that, each step is executed on both socket buffers if fragmentation occurs. To ensure the correct order of both frames, the second socket buffer is transmitted directly after the first socket buffer.

```
1  struct macsec_fragmentation_buffer {
2    sci_t sci;
3    struct macsec_fragmentation_buffer *next;
4    struct macsec_fragmentation_buffer *prev;
5    unsigned char *data;
6    unsigned int len;
7    unsigned short proto;
8  }
9
```

Figure 4.5: Fragment buffer data structure

4.2.2 Reassembly

The reassembly takes place after the MACsec frame is validated and the SecTAG and ICV are stripped. If the *More Fragments* (MF) bit is set, the payload, its length and the original Ethertype are saved in a double linked list as shown in 4.5. As every *Secure Channel* (SC) has its own buffer the network device is still able to receive frames of different SCs.

When a frame arrives that does not have the MF bit set, it is checked whether data is buffered for this SC. If this is the case, the data of the fragmentation buffer is prepended to the payload of the frame. Additionally, the Ethertype is set according to the saved one. After the buffer is cleared, the frame is delivered.

4.2.3 Challenges

While implementing the proposed modifications of MACsec, several issues arised. The three most significant ones are described in the following paragraphs.

Order of frames The protocol relies on the correct order of frames. As both endpoints are connected directly, it must be ensured that the network interface retains the order on transmission. Therefore, the new MACsec driver transmits both fragments successively to the network interface.

Each network interface has a configured algorithm which manages the queue. This group of algorithms is named *Queueing Disciplines* (Qdiscs) [Hub+02]. A simple Qdisc is *First in, First out* (FIFO)—packets are sent in the order they arrive. There are also Qdiscs, which do not retain the order of frames as they arrive, but prefer some with certain properties. One of them is FQ_CoDel[6], which drops packets with a high queing delay.

[6] http://man7.org/linux/man-pages/man8/tc-fq_codel.8.html

The Qdisc FQ_CoDel has been set for the network interface on the used test system. This led to high retransmission and error rates while using "iperf3".

As the virtual device should not change the configuration of the underlaying network interface, it does not set a Qdisc. The only solution that may seem reasonable is to print a warning to kernel log, if a Qdisc differs from FIFO, as this may decrease the performance of MACsec fragmentation significantly.

Concatenation The implementation of the concatenation algorithm has several requirements, two of them are the possibility to access and manipulate the queue of the MACsec device. By default the MACsec device has no queue but after configuring a Qdisc—which is described in the previous section—it was possible to access it. When concatenation of several SDUs happens, it may occur that the last SDU must additionally be fragmented. If this happens, it must be guaranteed that the last fragment of this SDU is sent in the next MACsec frame. Thus, this fragment must be the next one processed by the MACsec device and the queue must be manipulated to achieve that. After some investigation, it was possible to dequeue and enqueue SDUs, but not to manipulate the order of the queue. For time reasons, it was not possible to implement this manipulation and hence the current implementation does not support concatentation.

Fragmentation of TCP packets If the socket buffer has to be fragmented, it is split into two parts. Therefore, the original buffer is trimmed and the remaining payload is moved to another position, where the fragment points to.

To test the implementation the command line tools `ping` —which sends ICMP packets— and `iperf3` —which sends TCP packets—where used. In the beginning of each test everything worked but when `iperf3` was used after a varying amount of time the kernel crashed.

After many debugging attempts, the reliability of TCP turned out to be a problem. TCP retransmits a packet when no acknowledge is received. Therefore, the kernel keeps a pointer of the socket buffer, so on retransmission the frame must not be recreated. If the socket buffer was fragmented by MACsec the kernel expected a longer one which led to a kernel panic.

To solve this problem, the implementation now checks, whether a reference to the socket buffer is still hold by anyone. If this is the case the socket buffer and its data is cloned, so if TCP now retransmits a packet it is not modified. This is recommended by the documentation[7].

[7] `https://www.fsl.cs.sunysb.edu/kernel-api/re501.html`

5 Evaluation

The problem of a decreased MTU when using MACsec is solved with the proposed modifications of chapter 3, which were implemented as described in section 4.2.

To assess, if these modifications lead to significant performance changes, several measurments were made. These are compared with the solution of using jumbo frames, which is also a possibility to mitigate the problem of a lower MTU. This solution is considered to be optimal but not applicable in the given scenario.

As the modified driver is sending two frames instead of one considering a large frame, *Round-trip time* (RTT) might be affected. Furthermore, an additional MACsec header must be sent, which might affect the bandwidth. As a third measurement the CPU usage during the bandwidth test is recorded, to see if the fragmentation and reassembly has a significant impact on it.

For the evaluation two computers are directly connected to each other with a Cat5e network cable. Table 5.1 contains the hardware specifications of both machines.

Processor	Intel i5-4590 3.3GHz
RAM	16GB
Network interface	Fast Ethernet & Gigabit Ethernet
Operating System	Arch Linux[8]with linux kernel 4.15.7

Table 5.1: Specifications of the machines used for the evaluation

5.1 Experiment

Evaluated is the RTT and bandwidth. For the measurement of the RTT the command line tool "ping" is used. The bandwidth is measured with "iperf3"[9].

The test cases are:

- Ethernet with MTU of 1468 bytes, 1500 bytes and 2936 bytes
- MACsec with MTU of 1468 bytes, 1500 bytes and 2936 bytes by enabling Jumbo Frames
- Modified MACsec with MTU of 1468 bytes, 1500 bytes and 2936 bytes

[8] `https://www.archlinux.org/`
[9] `http://software.es.net/iperf/`

The MTU of 1468 bytes is chosen because it is the standard MACsec MTU. With this MTU the modified MACsec does not do any fragmentation. The MTU of 1500 bytes is chosen because it is the standard ethernet MTU. And with an MTU of 2936 bytes the modified MACsec version sends two full frames, instead of one full and one small frame as in the setting with an MTU of 1500 bytes.

The measurement is automated with a shell script, which runs on one of both machines. In the following, this machine is referred to as sender and the other one as receipient.

In the beginning, the sender sets the configuration according to the current test case. After that, it configures the receipient and starts an iperf3 server via ssh. A single bandwidth measurement performs a 10 second long iperf3 test, which builds up a TCP connection and measures the average bandwidth. This test is repeated 1000 times for each test case, except the Ethernet case with a MTU of 2936 bytes where it is repeated 100 times. The ethernet test case is interesting for comparability but plays a minor role for the assessment of the MACsec modifications. For the RTT measurment an adaptive ping is started, which sends and receives 50000 ICMP messages. Usually, ping sends one echo request every second but with the adaptive ping it sends the next request as soon as the last was acknowledged.

5.2 Measurements

5.2.1 Round-trip time

The results of the RTT are displayed in figure 5.1 and table 7.1. It is noticeable that two test cases which used jumbo frames—Ethernet and MACsec with an MTU of 2936 bytes—have a significant higher RTT (0.44ms to 0.52ms) and higher variance, compared with the remaining test cases (0.14ms to 0.17ms).

The reasons for this are not further researched in this thesis but there are several publications which detect a high latency when using jumbo frames [Com05].

To make the results of the other test cases more visible, both described cases are removed in figure 5.2.

The RTT of Ethernet with an MTU of 1468 bytes (0.139ms) respectively 1500 bytes (0.14ms) is lower than the RTT of the test cases where MACsec is used. It can be assumed that this is a result of the cryptographic operations when using MACsec.

It is noticeable that the modified MACsec (0.155ms) has a similar RTT as the unmodified MACsec (0.154ms) with a frame size of 1468 bytes. Here, no fragmentation happens. When the modified MACsec does fragmentation with a frame size of 1500 bytes, the RTT increases to 0.166ms. Compared to MACsec with jumbo frames and a MTU of 1500 bytes (0.155ms) this is an increase of 7.1%. It is assumed that this is the result of having to send two frames instead of one. The modified MACsec with an MTU of 2936 bytes has an increased RTT (0.174ms), which might be the result of a bigger frame which has to be sent. But compared to MACsec with jumbo frames and an MTU of 2936 bytes it is significant faster.

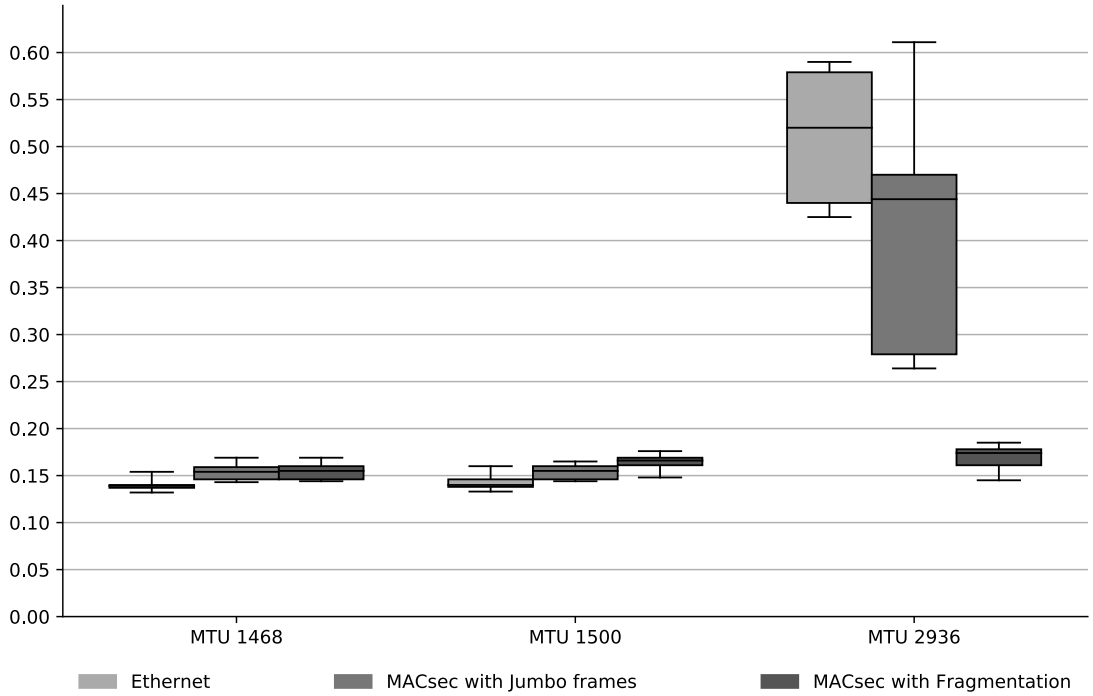


Figure 5.1: RTT measurements for the different test cases.
$n = 50000$

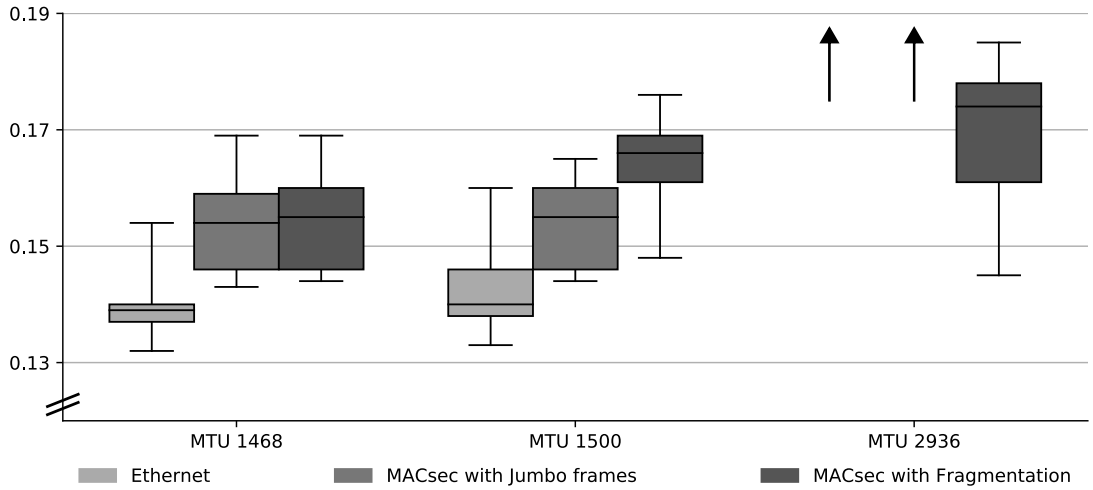


Figure 5.2: Clipped RTT measurements for the different test cases, full results visible in figure 5.1.
$n = 50000$

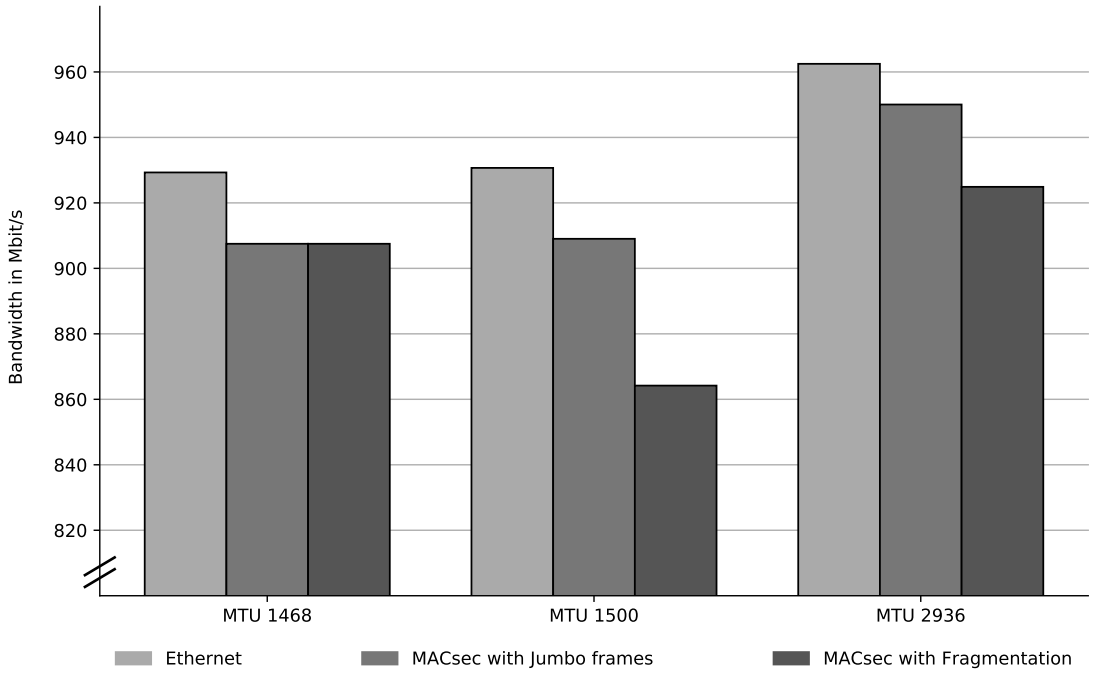


Figure 5.3: Bandwidth measurements for the different test cases.
$n = 1000$, except for Ethernet with MTU 2936 $n = 100$

5.2.2 Bandwidth

The results of the bandwidth test are shown in figure 5.3 and table 7.2.

The data shows that, independent of the test case, the variance is small. All test cases where no MACsec is used reach a higher bandwidth compared with the other test cases which had the same MTU. This decrease when using MACsec might be the result of the additional 32 bytes long header, which has to be sent in each frame.

MACsec (907.4 Mbit/s) and the modified MACsec (907.38 Mbit/s) reach a similar bandwidth when using a frame size of 1468 bytes. When the modified MACsec does fragmentation, the bandwidth decreases compared to MACsec with the same MTU of 1500 bytes respectively 2936 bytes. In the fragmentation cases a second Ethernet and MACsec header have to be sent which lead to this decrease. The modified MACsec reaches 95% respectively 97% of the bandwidth that MACsec with the same MTU creates.

5.2.3 CPU Usage

The results of the CPU usage during the bandwidth test are shown in figure 5.4 and table 7.3. As the data is generated during the bandwidth test, the CPU usage also depends on the size of this data.

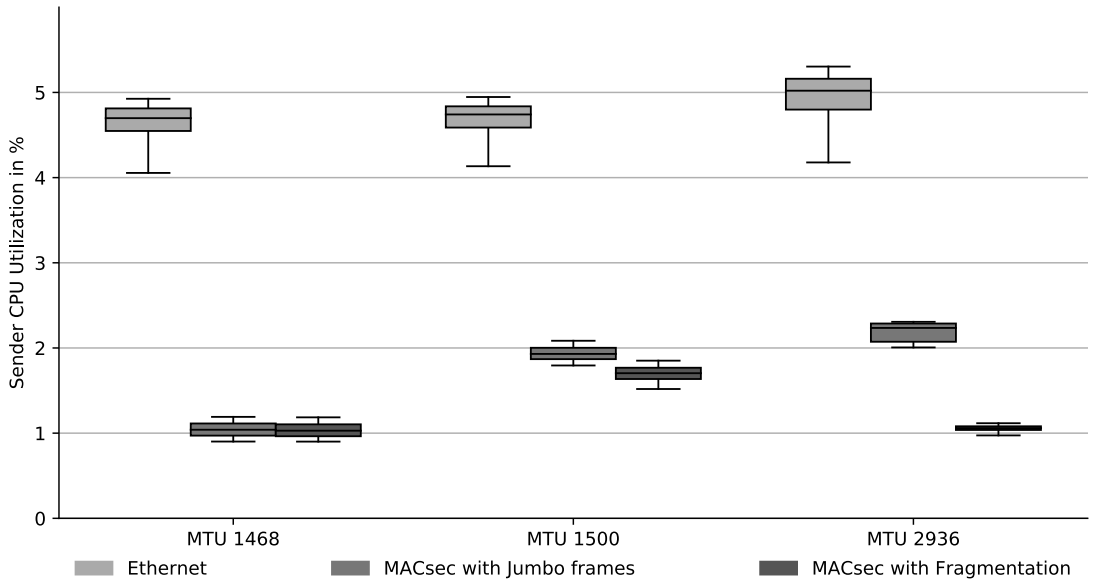


Figure 5.4: CPU usage during the different test cases regarding the bandwidth test. $n = 1000$, except for Ethernet with MTU 2936 $n = 100$

During the tests without MACsec, the CPU usage is significantly higher (5.55% to 5.16%) than during the tests where MACsec is applied (0.96% to 2.29%). Furthermore, it is noticeable that the CPU usage during the tests with the modified MACsec is in the same order of magnitude as during the use of MACsec.

There is at least no obvious reason, why during the tests without the usage of MACsec the CPU was used significantly more. These were the cases, where more data for the bandwidth test was generated due to a better performance as evaluated in section 5.2.2. For investigation of this reason, a normalization of the CPU usage result was made, which is displayed in figure 5.5. But even here the results are significantly worse. The results of the CPU usage measurement need more investigation.

5.3 Security Evaluation

This thesis assumes that MACsec is secure. It provides integrity, confidentiality, data origin authenticity, and protection against replay attacks.

The proposed modifications alter the SecTAG. Two reserved bits are replaced by the More Fragments and Extension bit. Furthermore, additional extension headers may be appended to the SecTAG.

These modifications—which are explained in further detail in chapter 4—are evaluated regarding security in the following paragraphs.

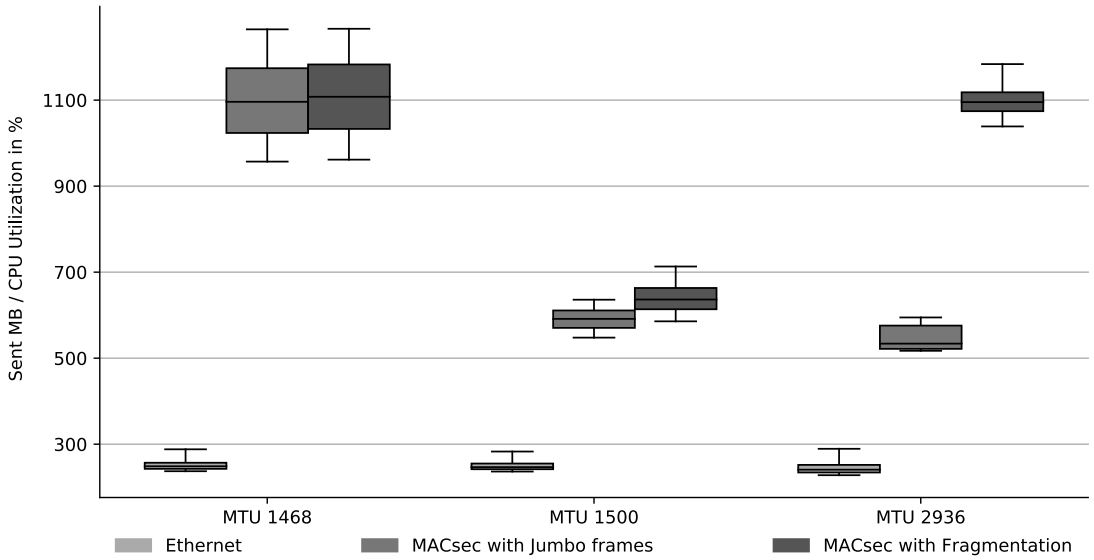


Figure 5.5: Sent megabytes divided by CPU usage in percent during bandwidth test. $n = 1000$, except for Ethernet with MTU 2936 $n = 100$

5.3.1 Confidentiality

MACsec provides confidentiality by encrypting the secure payload. This does not include the SecTAG.

If fragmentation is applied, the SDU is split and each part is the secure payload of one MPDU. If concatenation is applied, several SDUs build the secure payload of one MPDU. As the secure payload is constructed before MACsec is applied, it still provides the same confidentiality regarding the encryption of the transmitted data. The encryption process as well as the used algorithms or keys are not modified.

The SecTAG is extended with information of whether a fragment follows and the length of each concatenated SDU. The number and corresponding length of transmitted SDUs is obtainable for an attacker, when the unmodified MACsec is used. Hence, no additional information is disclosed.

Therefore, confidentiality is not weakened by the proposed modifications.

5.3.2 Integrity

MACsec provides integrity by building a message authentication code named ICV. The ICV includes the source and destination adresses, the SecTAG and the secure payload.

Fragmented as well as concatenated SDUs are always contained in the secure payload of an MPDU and, therefore, integrity protected by the ICV. The modifications of the SecTAG are also protected by the ICV. The process defined in chapter 4 specifies that on transmit the modifications and the secure payload are built before calculating the ICV. Furthermore, on receive the ICV is verified before the data is processed. Thus, an outside

attacker has no possibility to modify data undetectable. This is visible in the flowcharts describing the transmission (figure 4.1) and receive (figure 4.2) process.

Therefore, the integrity is not weakened by the proposed modifications. The data origin authenticity is guaranteed by integrity and authentication through the MAC Security Key Agreement Entity. As integrity is not weakened and the MAC Security Key Agreement Entity is not part of MACsec standard data origin authenticity is still ensured.

5.3.3 Availability

The proposed modifications of MACsec specify that received data is verified by MACsec before the process of reassembly and deconcatenation occurs. Therefore, outside attackers cannot exploit the introduced behavior, as it is assumed that MACsec is secure. On transmission and receive, the proposed modifications perform similar to MACsec as shown in the previous sections. Inside attackers which participate legitimately in the communication can exploit the introduced behavior regarding availability, for example by sending frames which have the MF bit set. This leads to a reassembly of not associated fragments and therefore illegable data is delivered. But as MACsec does not prevent insider attacks, this is not considered.

Therefore, the availability regarding outside attackers is not weakened by the proposed modifications.

5.3.4 Replay protection

MACsec provides replay protection by using incrementing packet numbers. Packets with a number which was already received are dropped[10]. This behavior is not changed by the modifications, the check of the packet number is done before the modifications are applied. Furthermore, the field containing and the build of the packet number is not changed.

Therefore, the replay protection of MACsec is not weakened by the proposed modifications.

[10] The exact behavior is configurable

5.4 Results

The results of the performance evaluation and security evaluation are satisfying.

The RTT results are as expected if fragmentation is applied the RTT just slightly increases. Additionally, the results of the bandwidth test shows that the modified MACsec still reaches 95% of the results of MACsec with jumbo frames. The measurements of the CPU usage are not completely comprehensible but the results of using modified MACsec are in the same order of magnitude as unmodified MACsec. The security evaluation shows that the proposed modifications maintain the security level of MACsec.

The proposed modifications appear to successfully solve the given problem without decreasing the performance significantly, as demonstrated by the prior evaluation. A working implementation as a linux kernel module is provided. This outcome leads to the conclusion that functionality of the proposed solution is given.

6 Conclusion

The goal of this thesis was to develop a transparent fragmentation solution while using MACsec to mitigate the problems of a decreased MTU. It is required to not mitigate the problem by the use of jumbo frames or fragmentation in upper layers. Three established fragmentation processes in computer networks were presented. Two general approaches for the given problem were deduced and discussed. The approach of fragmenting MPDUs turned out to be vulnerable, so the approach of fragmenting SDUs was choosen. To optimize this solution a concatenation scheme was developed. The fragmentation solution was implemented in C for the linux kernel. The concatenation scheme was not implemented. The implemented algorithm in MACsec was then evaluated regarding performance and security. For evaluation of performance the implementation was deployed to two physical machines. The results of this evaluation were compared to the solution of using jumbo frames, which is considered as an optimal solution. Here, the proposed solution appeared to be successful, as the performance results were just slightly below the results of the optimum. Furthermore, the security evaluation showed that the proposed solution is secure.

The solution of fragmenting SDUs solves the problem of a decreased MTU when using MACsec. It performs well—as the evaluation showed—and maintains the security which is established by MACsec. The developed improvement of a concatenation process appeared to be an optimization, which could be implemented and evaluated by future work. Moreover, the field of other improvements and optimizations for MACsec can be researched. The behavior when using Jumbo Frames seems to be an interesting topic which can be investigated, as the evaluation detected some notable deviations from expected behavior.

7 Appendix

7.1 Measured Data

The following tables contain the measured data of the experiment described in chapter 5.

		Percentile in ms		
Test Case	MTU in	25th	50th	75th
Ethernet	1468	0.137	0.139	0.140
	1500	0.138	0.140	0.146
	2936	0.440	0.520	0.579
MACsec	1468	0.146	0.154	0.159
	1500	0.146	0.155	0.160
	2936	0.279	0.444	0.470
Modified MACsec	1468	0.146	0.155	0.160
	1500	0.161	0.166	0.169
	2936	0.161	0.174	0.178

Table 7.1: Round-trip time measurement

		Percentile in Mbit/s		
Test Case	MTU in	25th	50th	75th
Ethernet	1468	929.23	929.30	929.40
	1500	930.60	930.69	930.75
	2936	962.46	962.48	962.49
MACsec	1468	907.40	907.51	907.64
	1500	908.94	909.03	909.15
	2936	950.01	950.05	950.26
Modified MACsec	1468	907.38	907.51	907.64
	1500	863.95	864.19	864.41
	2936	924.75	924.90	925.06

Table 7.2: Bandwidth measurement

		Percentile in %		
Test Case	MTU in byte	25th	50th	75th
Ethernet	1468	4.55	4.70	4.81
	1500	4.59	4.74	4.84
	2936	4.80	5.02	5.16
MACsec	1468	0.97	1.04	1.11
	1500	1.87	1.93	2.00
	2936	2.07	2.24	2.29
Modified MACsec	1468	0.96	1.03	1.10
	1500	1.64	1.70	1.77
	2936	1.04	1.06	1.08

Table 7.3: CPU usage in percent on the sender machine during the bandwidth test

Acronyms

AN Association Number. 7–10, 17

CA Secure Connectivity Association. VIII, 7

E Extension. 12, 13, 16, 17, 23, 26, 27, 34

FI Framing Info. 12, 13

FIFO First in, First out. 28, 29

ICMP Internet Control Message Protocol. 5, 31

ICV Integrity Check Value. 8–10, 16, 21, 27, 28, 35

IEEE Institute of Electrical and Electronics Engineers. VII, 1, 4, 6, 11, 16

IP Internet Protocol. 3, 5, 10–12, 15

ISO International Standards Organisation. 3

IV Initialization Vector. 9

KaY MAC Security Key Agreement Entity. 6, 7, 36

LAN Local Area Network. 4, 6–8

LI Length Indicator. 13

LTE Long Term Evolution. 10, 12, 15, 16

MAC Medium Access Control. 4, 11

MACsec Medium Access Control Security. VII, VIII, 1, 2, 4, 6–10, 14–16, 18–23, 26–31, 33–38

MF More Fragments. 11, 15–18, 23, 26–28, 34, 36

MPDU MACsec Protocol Data Unit. VIII, 8, 9, 14–16, 18–23, 26, 27, 35, 38

MTU Maximum Transmission Unit. 1–3, 5, 11, 14–16, 18, 19, 23, 27, 30, 31, 33, 38

OSI Open Systems Interconnection. VIII, 3–5, 12

PDU Protocol Data Unit. VIII, 3, 4, 12, 13, 16

PMTUD Path MTU Discovery. 5

PN Packet Number. 8, 9, 17

Qdisc Queueing Discipline. 28, 29

RLC Radio Link Control. 12, 13

RTT Round-trip time. VIII, IX, 5, 30–32, 37, 39

SA Secure Association. 7, 9, 10

SAK Secure Association Key. 7

SC Secure Channel. 7, 16, 23, 26, 28

SCI Secure Channel Identifier. 7–10

SDU Service Data Unit. VIII, 3–5, 9, 10, 12–23, 26, 29, 35, 38

SecTAG Security TAG. VIII, 8–10, 16, 17, 21, 23, 26–28, 34, 35

SecY MAC Security Entity. 6, 7, 10

SEGL Segment Length. 26

SL Short Length. 8, 9, 17, 26

SN Sequence Number. 12, 13

TCI TAG Control Information. 8, 9, 17

TCP Transmission Control Protocol. 5, 29, 31

UM Unacknowledged Mode. 12, 42

UMD Unacknowledged Mode Data. VIII, 12, 13

WLAN Wireless LAN. 5, 10, 11, 15

Bibliography

[3GP06] 3GPP. *Release 8*. Technical Specifications (TS). 3rd Generation Partnership Project (3GPP), 2006. URL: https://portal.3gpp.org/Specifications.aspx?q=1&releases=182.

[3GP17] 3GPP. *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification (3GPP TS 36.322 version 14.0.0 Release 14)*. Tech. rep. 2017. URL: http://www.etsi.org/deliver/etsi_ts/136300_136399/136322/14.00.00_60/ts_136322v140000p.pdf.

[94] *IEC 7498-1 1994: Information Technology–Open Systems Interconnection–Basic Reference Model: The Basic Model*. Standard. Geneva, CH: International Organization for Standardization, 1994. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip.

[Bac+16] Daniel Bachlechner et al. "IT-Sicherheit für die Industrie 4.0". In: (2016). URL: http://www.bmwi.de/Redaktion/DE/Publikationen/Studien/it-sicherheit-fuer-industrie-4-0.html.

[Com05] Chelsio Communications. "Ethernet Jumbo Frames The Good, the Bad and the Ugly". In: (2005). URL: https://www.chelsio.com/assetlibrary/solutions/Chelsio_Jumbo_Enet_Frames.pdf.

[Dub16] Sabrina Dubroca. "MACsec Encryption for the wired LAN". In: Netdev1.1. Feb. 2016. URL: https://www.netdevconf.org/1.1/proceedings/slides/dubroca-macsec-encryption-wire-lan.pdf.

[GM01] Liang Guo and I. Matta. "The war between mice and elephants". In: *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*. Nov. 2001, pp. 180–188. DOI: 10.1109/ICNP.2001.992898.

[Gro06] WG802.1 - Higher Layer LAN Protocols Working Group. *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security*. Tech. rep. Aug. 2006, pp. 1–150. DOI: 10.1109/IEEESTD.2006.245590.

[Gro10] WG802.1 - Higher Layer LAN Protocols Working Group. "IEEE Standard for Local and metropolitan area networks–Port-Based Network Access Control". In: *IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004)* (Feb. 2010). Ed. by Mick Seaman, pp. 1–205. DOI: 10.1109/IEEESTD.2010.5409813.

[Gro11] WG802.1 - Higher Layer LAN Protocols Working Group. “IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Security Amendment 1: Galois Counter Mode–Advanced Encryption Standard– 256 (GCM-AES-256) Cipher Suite”. In: *IEEE Std 802.1AEbn-2011 (Amendment to IEEE Std 802.1AE-2006)* (Oct. 2011), pp. 1–52. DOI: 10.1109/IEEESTD.2011.6047536.

[Gro16] WG802.11 - Wireless LAN Working Group. *IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Tech. rep. Dec. 2016, pp. 1–3534. DOI: 10.1109/IEEESTD.2016.7786995.

[Hub+02] Bert Hubert et al. “Linux advanced routing & traffic control”. In: *Ottawa Linux Symposium*. 2002, p. 213.

[MD90] Jeffrey Mogul and Steve Deering. *Path MTU discovery*. RFC 1191. RFC Editor, Nov. 1990. URL: http://www.rfc-editor.org/rfc/rfc1191.txt.

[MRM94] A. Marine, J. Reynolds, and G. Malkin. *FYI on Questions and Answers - Answers to Commonly asked "New Internet User" Questions*. RFC 1594. RFC Editor, Mar. 1994.

[Pfi12] Andreas Pfitzmann. “Security in IT networks: Multilateral security in distributed and by distributed systems”. In: *Lecture Notes, Technische Universitaet Dresden* (2012). URL: https://dud.inf.tu-dresden.de/~sk13/TUD-Web-CMS/LV/SaC-II/Script.pdf.

[Pos81a] J. Postel. *Internet Control Message Protocol*. STD 5. RFC Editor, Sept. 1981. URL: http://www.rfc-editor.org/rfc/rfc792.txt.

[Pos81b] Jon Postel. *Internet Protocol*. STD 5. RFC Editor, Sept. 1981. URL: http://www.rfc-editor.org/rfc/rfc791.txt.

[Pos81c] Jon Postel. *Transmission Control Protocol*. STD 7. RFC Editor, Sept. 1981. URL: http://www.rfc-editor.org/rfc/rfc793.txt.

[PP15] Jonathan Pfleeger Charles P. Margulies and Shari Lawrence Pfleeger. *Security in computing*. Upper Saddle River, NJ : 2015. URL: http://slubdd.de/katalog?TN_libero_mab216370339.

[TW11] Andrew S. Tanenbaum and David Wetherall. *Computer networks*. 5. ed., international ed. Boston [u.a.] : Pearson, 2011. ISBN: 0132553171. URL: http://slubdd.de/katalog?TN_libero_mab215325534.