SeCPRI: IPsec based Security layer for eCPRI

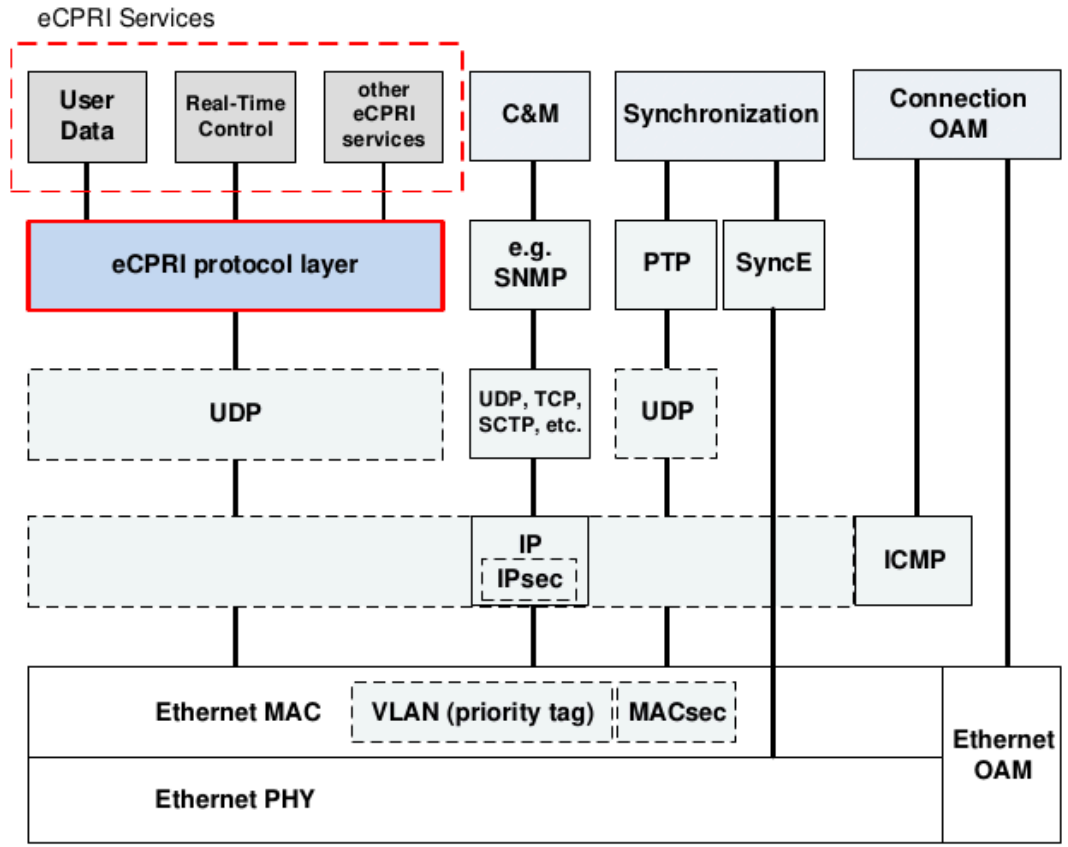
IPsec Workshop Madrid July 17 2025

Daniel Migault

I. IPsec: a potential candidate to secure IP fronthaul

The Common Public Radio Interface (CPRI) eCPRI the interface between the O-RU and O-DU.

- 3 Planes: User Plane (U-Plane), Control Plan (C-Plane) and Synchronization Plane (S-Plane)
- carried over ethernet or IP/UDP.
- secured with MACsec or IPsec.



> The modern 5G fronthaul, which connects the base stations to radio units in cellular networks, is designed to deliver microsecond-level performance guarantees using **Ethernet-based protocols**. Unfortunately, due to **potential performance overheads**, as well as misconceptions about the low risk and impact of possible attacks, **integrity protection is not considered a mandatory feature in the 5G fronthaul standards**.

[1] On the Criticality of Integrity Protection in 5G Fronthaul Networks

IPsec/IKEv2 is:

1. an open standard benefiting from multiple open source implementations benefiting from a huge experience in deployment.
2. Highly flexible with multiple extensions including Diet-ESP

To deliver microsecond-level performance guarantees, SeCPRI leverages IPsec flexibility to optimize security for eCPRI.

With high latency requirements and simplicity of developments, eCPRI messages are mostly small size messages sent as soon as possible.

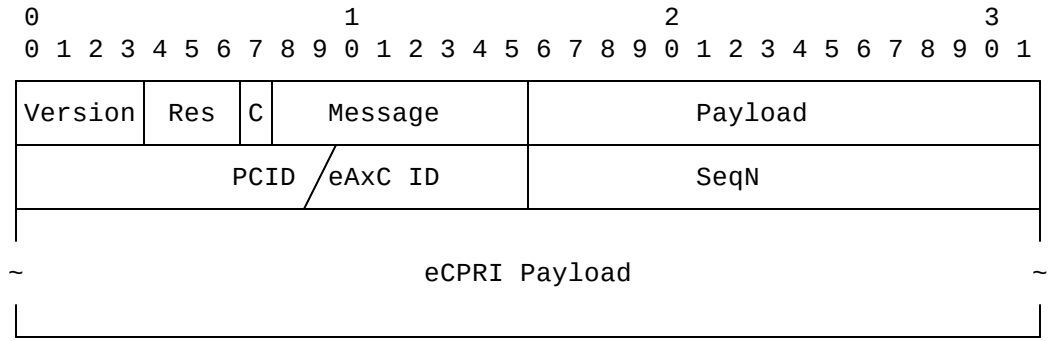
- This makes eCPRI communication highly sensitive to any security overhead

SeCPRI, MACsec, IPsec respectively results in a $\approx$ 14, 32, and 35.75 bytes overhead

SeCPRI is highly flexible and can be characterized by:

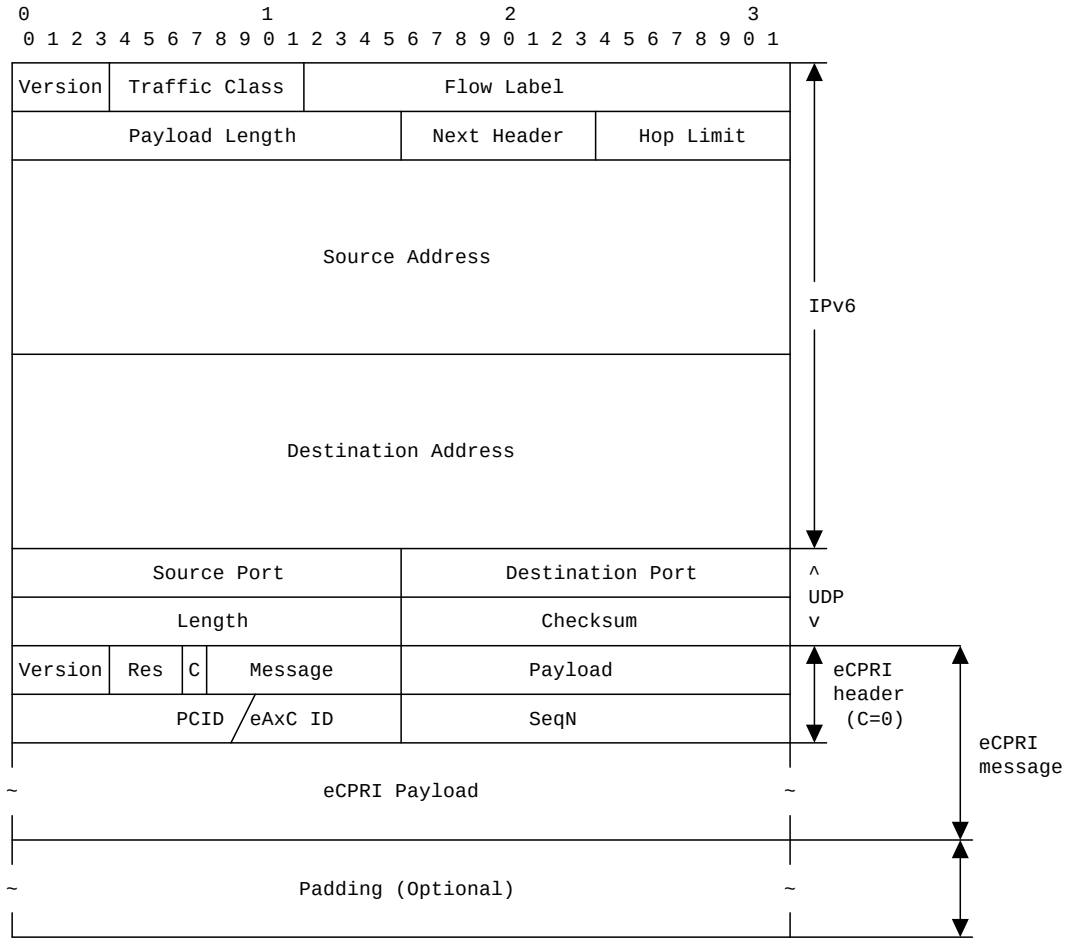
1. Compressing any part of an encrypted eCPRI message
2. Optimizing eCPRI processing:
    - organizing encrypted and clear text eCPRI messages for batch processing

II.1 eCPRI message Description

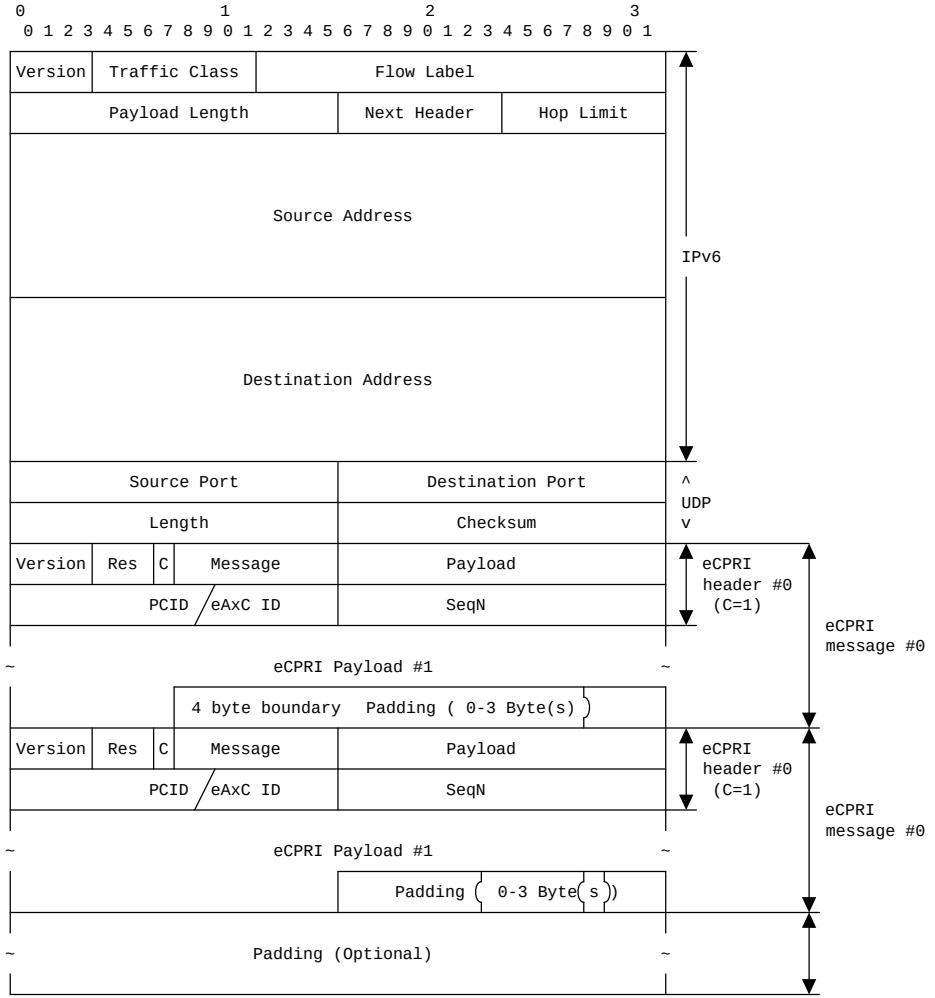


IP fronthaul (concatenated) IPv6/UDP/eCPRI or IPv6/UDP/[eCPRI, ..., eCPRI]

IPv6/UDP/eCPRI



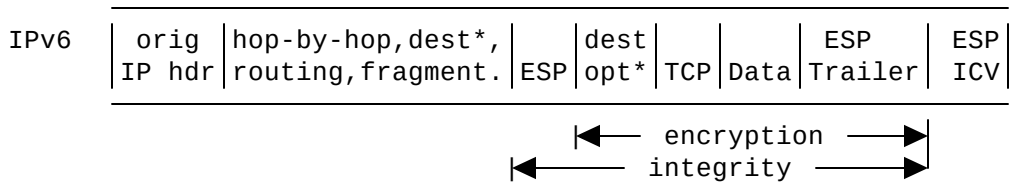
IPv6/UDP/[eCPRI, ..., eCPRI]



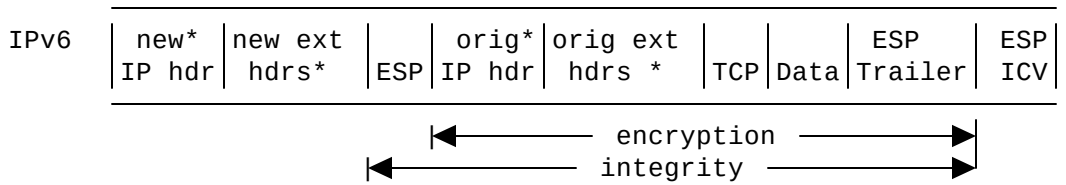
II.2 SeCPRI uses IPsec Transport mode

IP/UDP/eCPRI can be protected by IPsec in two modes:

Transport Mode



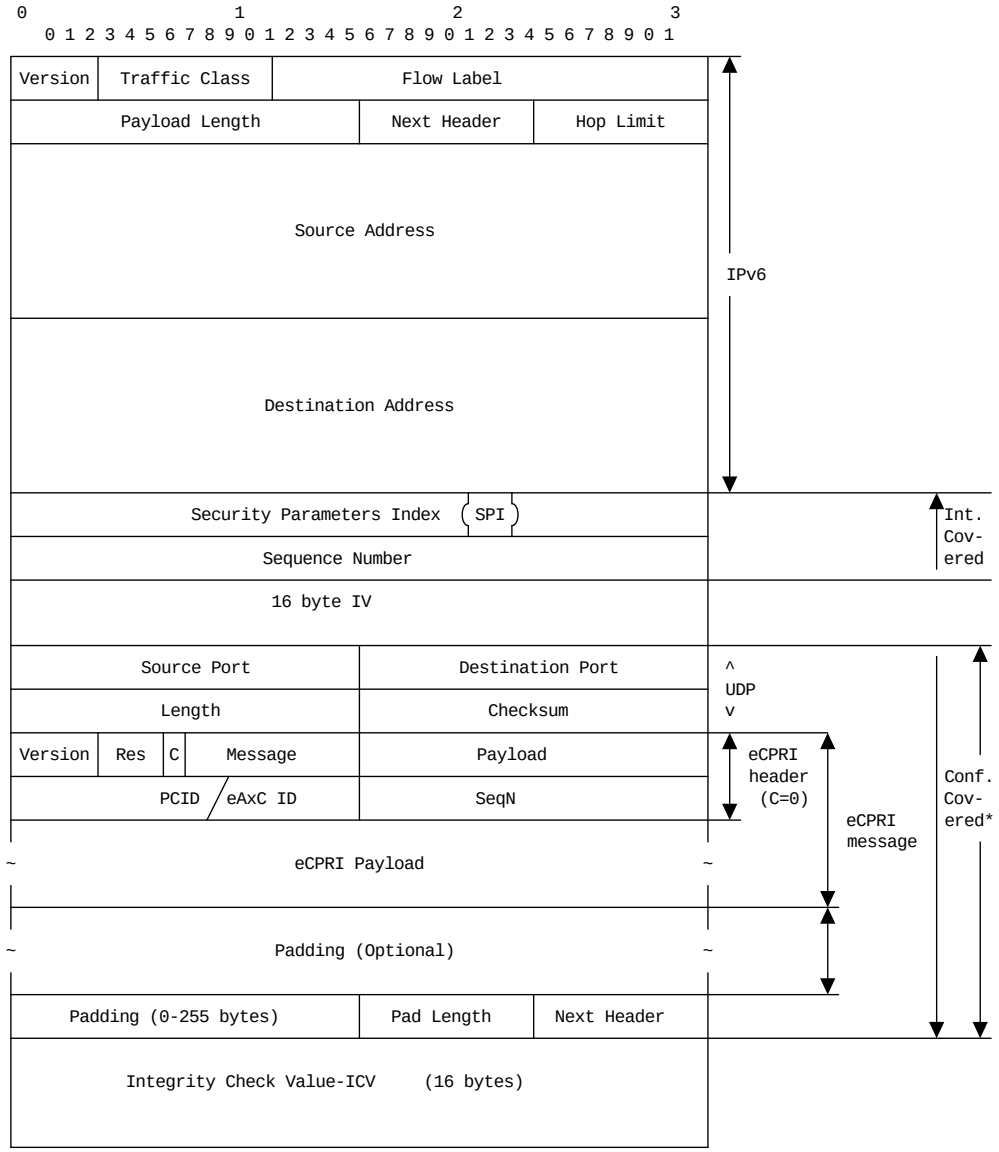
Tunnel Mode



SeCPRI only considers the Transport mode to ease the SeCPRI implementation:

- by preventing the compression of the inner IPv6 header

Standard ESP RFC4303 with ENCR_AES_GCM_16 RFC4601 used in Transport mode is as follows:

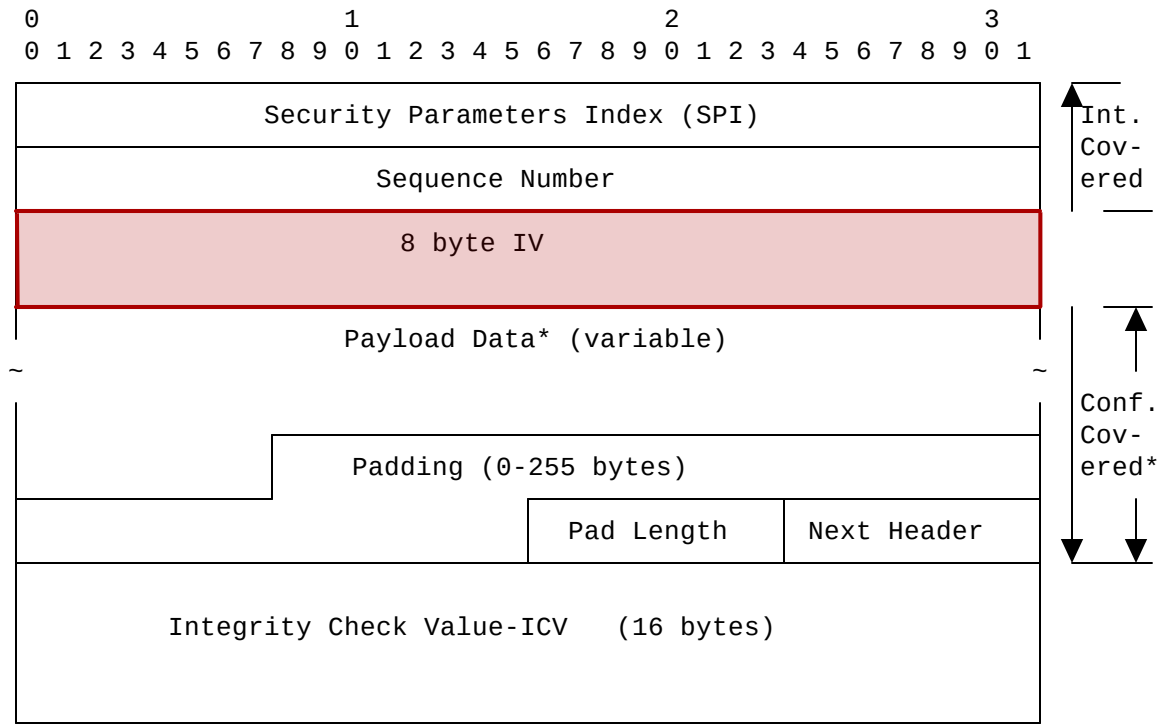


III. Compressing the encrypted eCPRI message: ESP/UDP/eCPRI or ESP/UDP[eCPRI, ..., eCPRI]

- III.1 SeCPRI uses ENCR_AES_GCM_16_IIV to compress the IV
- III.2 SeCPRI leverages Diet-ESP
- III.3 EEC
- III.4 CTEC
- III.5 IIPC

III.1 SeCPRI uses ENCR_AES_GCM_16_IIV to compress the IV

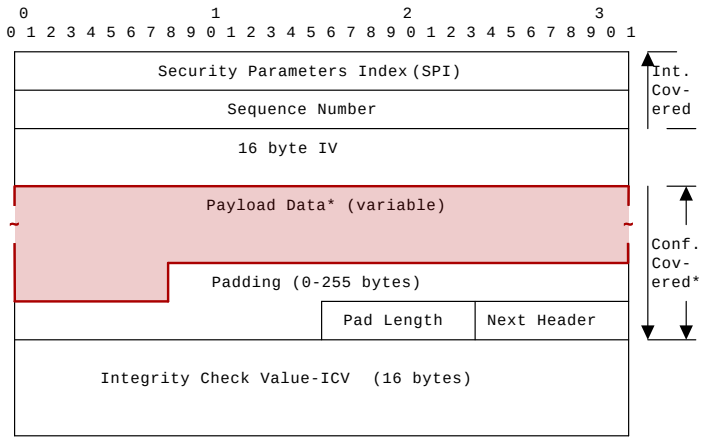
SeCPRI compresses the IV field with ENCR_AES_GCM_16_IIV RFC8750



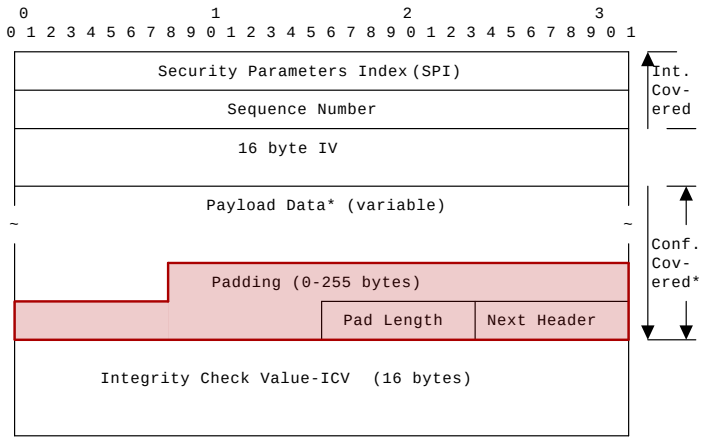
III.2 SeCPRI leverages Diet-ESP

Diet-ESP defines 3 compressors:

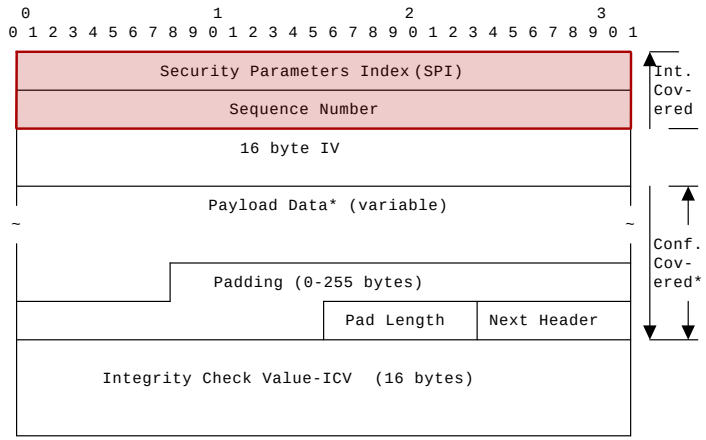
Inner IP packet (IIP):



ClearText ESP (CTE):

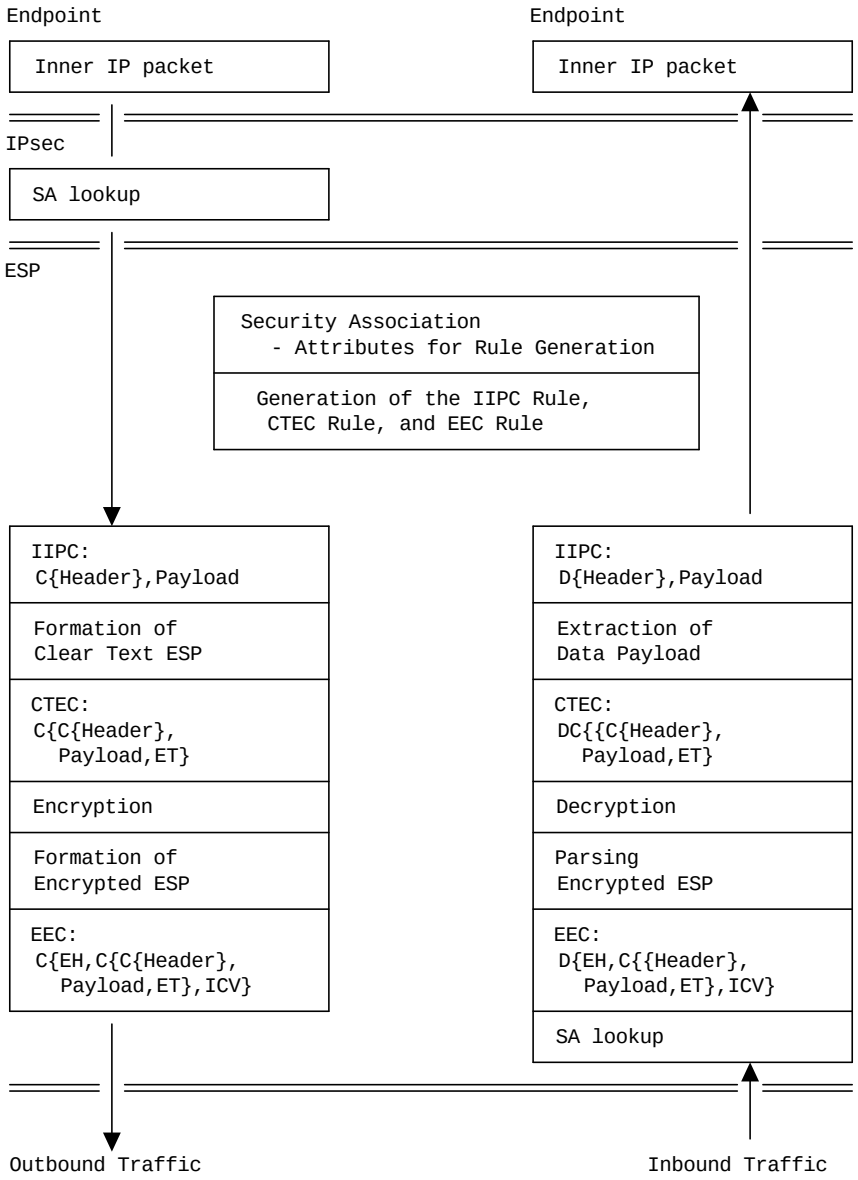


Encrypted ESP (EE):



Diet-ESP is a compression framework.

- It defines 3 Compressors (C.):
  - Inner IP C. (IIPC),
  - Clear Text ESP C. (CTEC),
  - Encrypted ESP C. (EEC)
- C designates the Compressed Header for the fields inside
- EH refers to the ESP Header
- ET refers to the ESP Trailer



III.3 EEC

SPI and SN fields to a number of bytes

III.4 CTEC

SeCPRI asssumes the ESP Trailer is compressed

- `alignment` is set to 8 bits
- `esp_trailer` is set to Optional

III.5 IIPC

SeCPRI considers UDP specific ports so that UDP can be compressed

- `ts_proto` : UDP
- `ts_port_src_start` = `ts_port_src_end` and `ts_port_dst_start` = `ts_port_dst_end`

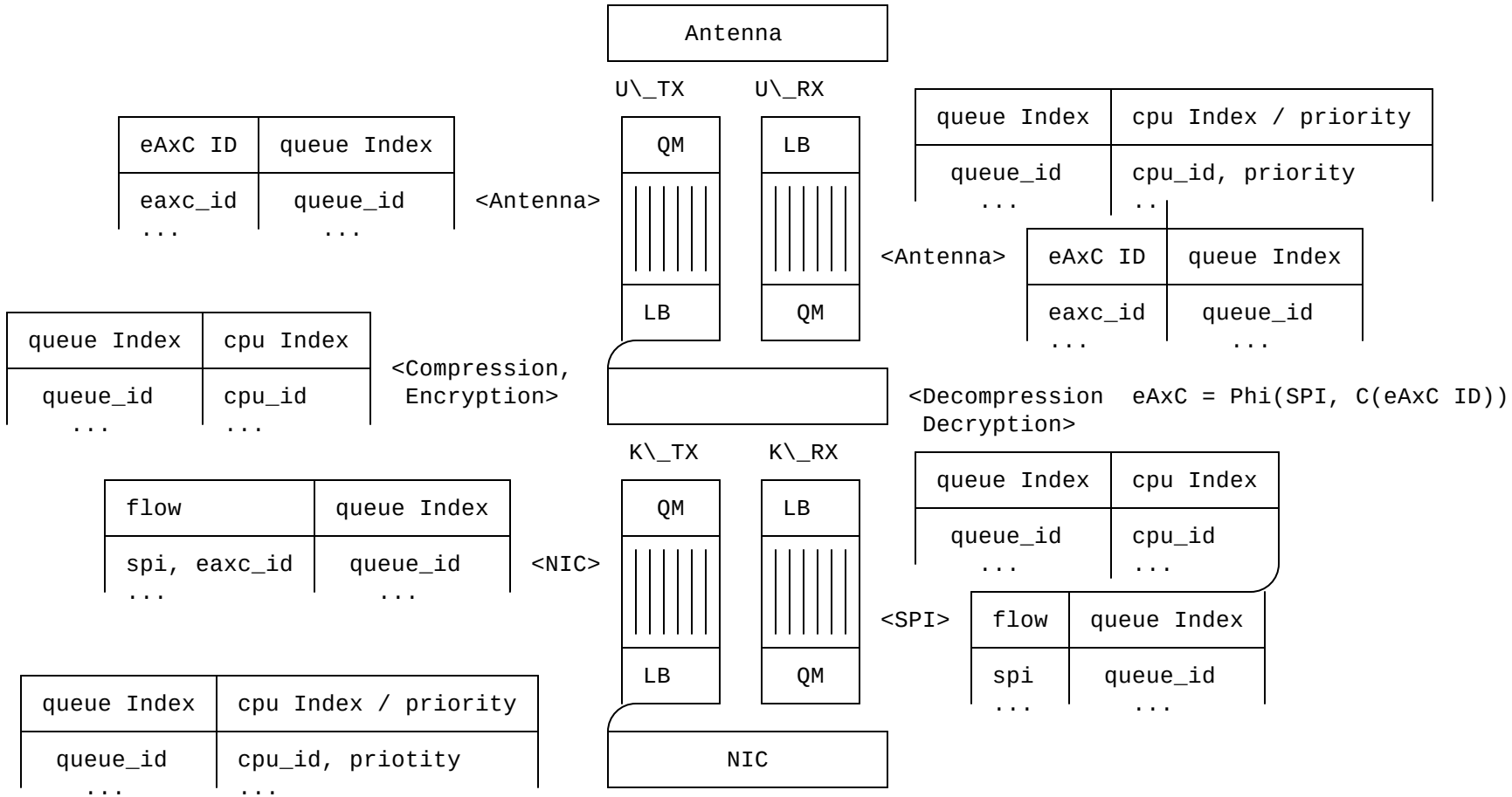
Diet-ESP defines a specific IIPC Profiles for eCPRI

IV. Optimization of eCPRI processing

eCPRI messages are processed according to their eAxC ID …

- but eAxC ID is not visible when the eCPRI message is encrypted

With SeCPRI, encrypted eCPRI message are correctly steered according to the SPI.


Antenna
U_TX
U_RX
QM
LB
LB
QM
eAxC ID
queue Index
eaxc_id
...
queue_id
...
<Antenna>
queue Index
cpu Index / priority
queue_id
...
cpu_id, priority
..
<Antenna>
eAxC ID
queue Index
eaxc_id
...
queue_id
...
queue Index
cpu Index
queue_id
...
cpu_id
...
<Compression,
Encryption>
<Decompression
Decryption>
eAxC = Phi(SPI, C(eAxC ID))
K_TX
K_RX
QM
LB
LB
QM
flow
queue Index
spi, eaxc_id
...
queue_id
...
<NIC>
queue Index
cpu Index
queue_id
...
cpu_id
...
<SPI>
flow
queue Index
spi
...
queue_id
...
queue Index
cpu Index / priority
queue_id
...
cpu_id, priotity
...
NIC

Lemme: A queue mapper does not introduce in-stream out-of-order packets by reducing or increasing the number of queues when streams are *partitioned* over both incoming and outgoing queues.

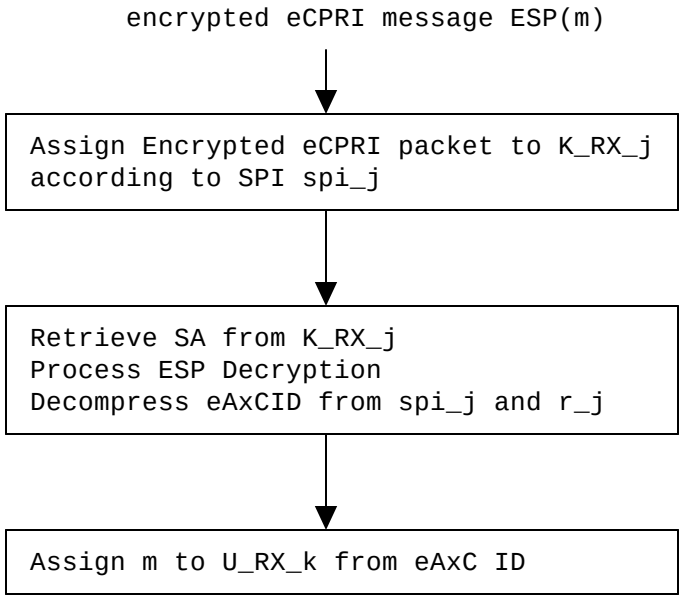
Let $\mathcal{A}$ be all possible eAxC IDs partitioned into $\mathcal{A}_{spi_j}$.

$$\mathcal{A} = \bigcup_{spi_j \in SPI_i} \mathcal{A}_{spi_j} with\ \mathcal{A}_{spi_j} \cap \mathcal{A}_{spi_k} = \emptyset\ \forall (j,k) \in |SPI_i| j \neq k$$

eAxC ID is expressed via:

- the SPI which designates $\mathcal{A}_{spi_j}$
- its rank r_j within $\mathcal{A}_{spi_j}$

Inbound



Outbound