N O R M A I T A L I A N A C E I

PROGETTO

Progetto
C. 1336

Data Scadenza Inchiesta
18-03-2024

Data Pubblicazione
2024-01

Classificazione
9-xxx

Titolo

Applicazioni ferroviarie, tranviarie, filoviarie e metropolitane – Sistemi di telecomunicazione, segnalamento ed elaborazione – PVS: Protocollo di comunicazione sicuro tra impianti di terra

Title

Railway applications – Signaling systems – PVS: Safety related transmission protocol for application between ground systems

Sommario

Questa Norma definisce i requisiti funzionali del Protocollo di comunicazione sicuro tra impianti di terra di sistemi di segnalamento, denominato PVS (Protocollo Vitale Standard).
PVS è specificato al fine di consentire lo scambio di dati in condizioni di sicurezza tra due applicazioni vitali generiche collegate a un canale di trasmissione non vitale.

Il documento è pubblicato in lingua inglese.

CEI COMITATO ELETTROTECNICO ITALIANO
AEIT FEDERAZIONE ITALIANA DI ELETTROTECNICA, ELETTRONICA, AUTOMAZIONE, INFORMATICA E TELECOMUNICAZIONI
CNR CONSIGLIO NAZIONALE DELLE RICERCHE

CONTENTS

41

FIGURES LIST

TABLE INDEX

Foreword

This document was prepared by the Italian Technical Committee CEI / CT9.

Introduction

This document is derived from the Technical Specification « PROTOCOLLO VITALE STANDARD », Rev.F, edited by RFI on 12/06/2017.

1 Scope

This document defines the functional requirements of PVS (Protocollo Vitale Standard).

PVS is designed to enable the exchange of data under safe conditions between two generic vital applications connected to a non-vital transmission channel.

It is assumed that the systems in which PVS protocol is applied may be subject to the risk of intentional attack.

Therefore, the use of cryptographic techniques for unauthorized access prevention is taken into consideration.

2 Normative references

[EN50129] EN 50129, *Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling*, November 2018.

[EN50159] EN 50159, *Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems*, September 2010.

3 Bibliography

[DAOLFARI89] M. Damiani, P. Olivo, M. Favalli, B. Riccò, *An Analytical Model for the Aliasing Probability in Signature Analysis Testing*, IEEE Transactions on Computer-Aided Design, vol. 8, no. 11, Nov. 1989

[ELSHSE98] Mahmoud S. Elsaholy, Samir I. Shaheen and Reda H. Seireg, *A Unified Analytical Expression for Aliasing Error Probability Using Single-Input External- and Internal-XOR LFSR*, IEEE Transactions on Computers, vol. 47, no. 12, December 1998.

[SARPUR80] D.V.Sarwate, M.B. Pursley, *Crosscorrelation properties of pseudorandom and related sequences*, Proc. of the IEEE, vol. 68, no. 5, p. 593-620, May 1980

[WILSLO76] F.J.Mac Williams, N.J.A.Sloane, *Pseudo-random sequences and arrays*, Proc. of the IEEE, p.1715-1730, Dec 1976

[DKNUTH69] D.E.Knuth, *The art of computer programming Vol. 2: Seminumerical algorithms*, Addison-Wesley, 1969

[NIST10] A.Rukhin, J.Soto, J. Nechatal, M.Smid,E.Barker,S.Leigh,M.Levenson, M.Vangel, D.Banks, A.Heckert, J.Dray, S.Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST, April 2010

(http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html)

4 Terms, definitions and abbreviations

For the purposes of this document, the following terms and definitions apply.

4.1 Terms and definitions

3.1
Instance
A specific realisation of the PVS protocol.

3.2
Node
one of the two elements involved in a communication instantiated by the PVS protocol.

3.3
ML-sequence
Maximum-length sequence, sequence of 2^m-1 states produced by a maximum-length linear feedback shift register.

4.2 Abbreviations

AEP	Aliasing Error Probability
AES	Advanced Encryption Standard
ALE	Adaptation & redundancy management Layer Entity
ALEPKT	ALE packet; PDU exchanged by ALE
AM	Application Message
AM+ACK	Application Message with ACK
AM+REQ	Application Message with Request of ACK
APL	Access Protection Layer
ApPDU	Application layer PDU
CENELEC	Comité Européen de Normalisation Électrotechnique
CMAC	Cipher-based Message Authentication Code
CRC	Cyclic Redundancy Check
DI	DIsconnect
DT	DaTa
EC	Execution Cycle
Ex	Expected Execution Cycle
IP	Internet Protocol
LFSR	Linear Feed-back Shift Register
MIS	Monotonic Increasing Sequences
MMI	Man Machine Interface
N/A	Not Applicable
NIST	National Institute of Standards and Technology – United States of America
NOP	No Operations – SAI frames (AM, AM+REQ, AM+ACK) with UserData field empty
PDU	Protocol Data Unit
PR-EC	Pseudo Random Execution Cycle
PR-EC-CHECK	Pseudo Random Execution Cycle Check
PR-EC-REM-RX	Pseudo Random Execution Cycle Remote Received
PR-Ex	Pseudo Random Expected Execution Cycle
PR-SN	Pseudo Random Sequence Number
PR-SN-CHECK	Pseudo Random Sequence Number Check
PR-SN-REM	Pseudo Random Sequence Number Remote received
PRS	Pseudo Random Sequences
RBC	Radio Block Center
RFI	Rete Ferroviaria Italiana
SAI	Safe Application Intermediate layer
Sa PDU	Safety layer PDU
SL	Safety Layer
SN	Sequence Number
TCP	Transmission Control Protocol
TCS	Train Control System
THR	Tolerable Hazard Rate
TS	Transport Service
UDP	User Datagram Protocol

5 General description

5.1 Objectives

The PVS protocol was designed to satisfy the following technical characteristic:

- the communication is based on a bi-directional data exchange;
- the protocol is independent from the physical medium and the related network equipment;
- the physical medium and the related network equipment can be redounded only for availability target;
- the nodes periodically exchange frames between them, but the emission period can be different;
- the protocol is protected against the threats identified in EN 50159;
- the protocol can be implemented both in vital computer with high-end and low-end processors;
- the protocol can be implemented in all safety architectures (e.g. 2oo2, 2oo3 or coded processor).

EN 50159 [Ref.1] classifies the transmission system in three categories:

- **Category 1** consists of systems which are under the control of the designer and fixed during their lifetime;
- **Category 2** consists of systems which are partly unknown or not fixed, however unauthorized access can be excluded;
- **Category 3** consists of systems which are not under the control of the designer, and where unauthorized access has to be considered.

The PVS protocol addresses all the three categories.

The solutions of Category 1 and 2 do not include any cryptographic algorithm; the management of Category 1, 2 or 3 networks can be chosen by configuration.

With reference to Fig.C.1 of EN 50159 (see Figure 5-1) the choice is:

- Category 1 and 2 transmission systems: messages type A0 (**blu line**) - only non-cryptographic safety code;
- Category 3 transmission systems: messages type B1 (**red line**) - only non-cryptographic safety code + cryptographic code.

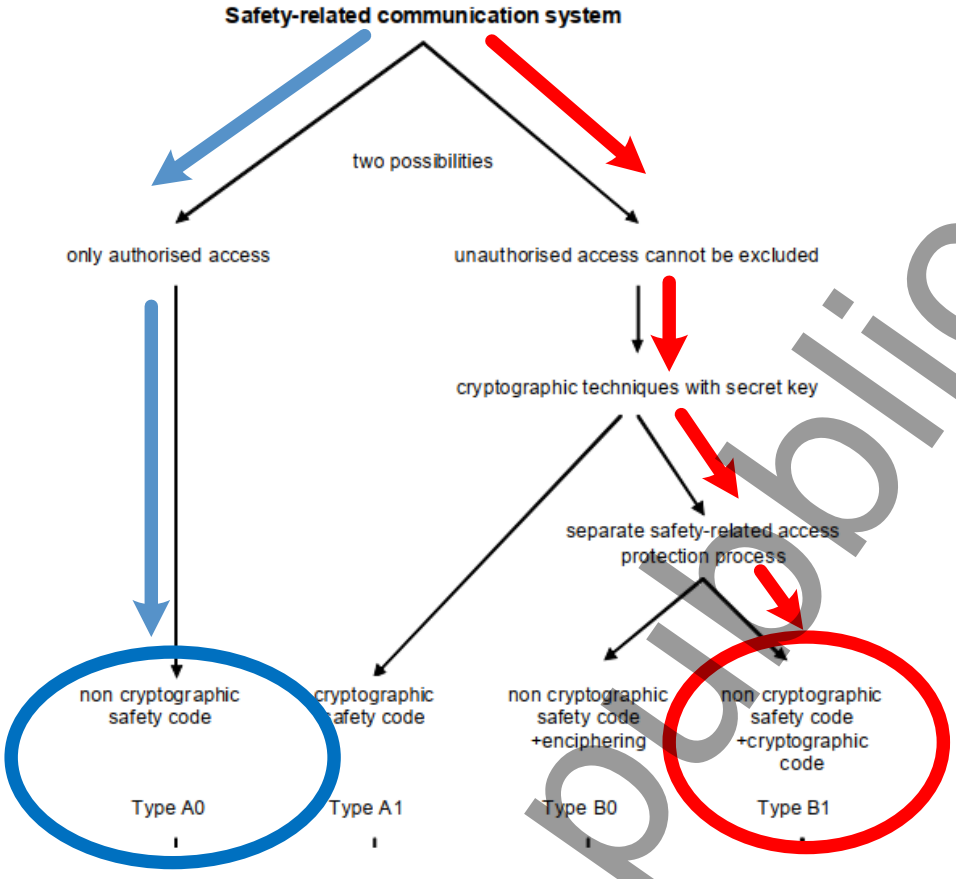


Figure 5-1 - Classification of the safety-related communication system

The cryptographic code can be implemented in the safety related equipment or externally, in accordance with the example given in Figure 5-2 (copying Figure C.7 of EN 50159).

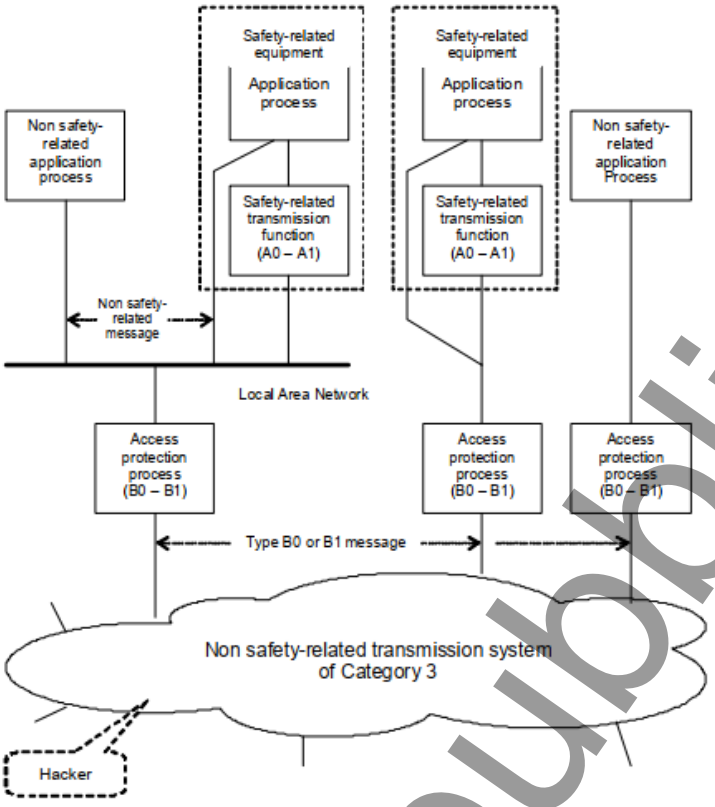


Figure C.7 – Communication between non safety-related and safety-related applications

Figure 5-2 - Use of a separate access protection layer

5.2 Protocol basic features

The PVS protocol is used to exchange data between two computers, where one of them is configured as Initiator and the other one as Responder.

5.3 Reference Architecture and layers description

5.3.1 Reference architecture

Reference architecture used in this specification is detailed in Figure 5-3.

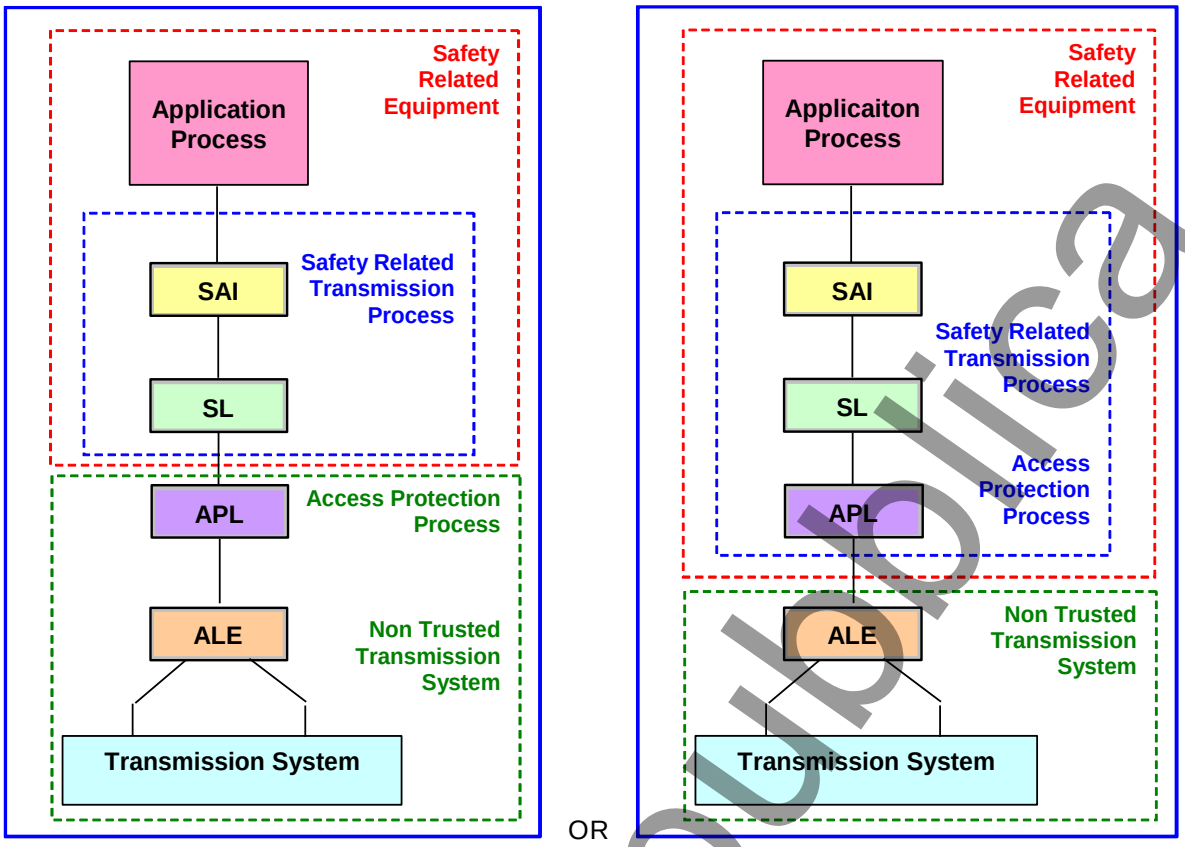


Figure 5-3 – reference architecture

With reference to Figure 5-3, the difference between the two schemes is in the APL location: in the second case it is located in the Safety Related Equipment, in the first one it is external.

In any case it is important to remark the requirement of EN 50159 §4 "Reference Architecture":

"safety-related cryptographic techniques which protect the safety-related message. These can either be realised by incorporating them in the safety-related equipment or having them outside of the safety-related equipment but checked by safety techniques. These techniques protect the safety related message in a Category 3 transmission system and are not needed in the case of a Category 1 or 2 transmission system."

5.3.2 Application Process

The Application Process is the layer in charge to convert the User Data received (Indication and Controls) in the internal vital computer data representation; it is the Application layer of the PVS protocol.

5.3.3 Safe Application Intermediate layer (SAI)

The SAI layer handles the interface towards the application and connection services; in addition it complements SL and APL for the defence mechanisms provided by EN 50159, ensuring:

- Protection against re-sequencing, repetition, insertion and deletion using a sequence number;
- "Freshness" and protection against "delay" by an Execution Cycle.

This mechanism is in accordance with what defined and validated in Subset-098 [Ref.3].

5.3.4 Safety Layer (SL)

SL is a layer positioned between SAI and APL.

SL ensures:

- Message integrity using non cryptographic safety code;
- Authenticity using node identifier and session identifiers.

5.3.5 Access Protection Layer (APL)

APL is an optional layer positioned between SL and ALE.

APL ensures:

- Access protection using AES cryptographic algorithm.

If APL is not executed on a safe architecture, its correct functioning shall be periodically verified (see Req_APL_001).

5.3.6 Adaptation & redundancy management Layer Entity (ALE)

ALE handles the redundancy (for availability purpose) operating on both communication physical links and ensures the correct interfacing towards the underlying transmission subsystems.

This layer's interface can be defined in such a way to be independent with respect to the transmission medium, to guarantee a suitable level of configurability to the Protocol Stack.

This document does not analyse the transmission media and the lower level protocol layers that could be adopted, but ALE must provide the correct interface with the Transmission System that can be chosen.

6 Physical level

In the context of this document any Physical level able to exchange the PVS message between the nodes can be considered.

7 Protocol level – Static description

7.1 Protocol data units for the different layers

With reference to the architecture depicted in Figure 5-3 this section is related to the specification of SAL, SL, APL and ALE blocks.

Figure 7-1 shows the encapsulation process of a PDU starting from the Application layer up to the Physical layer.

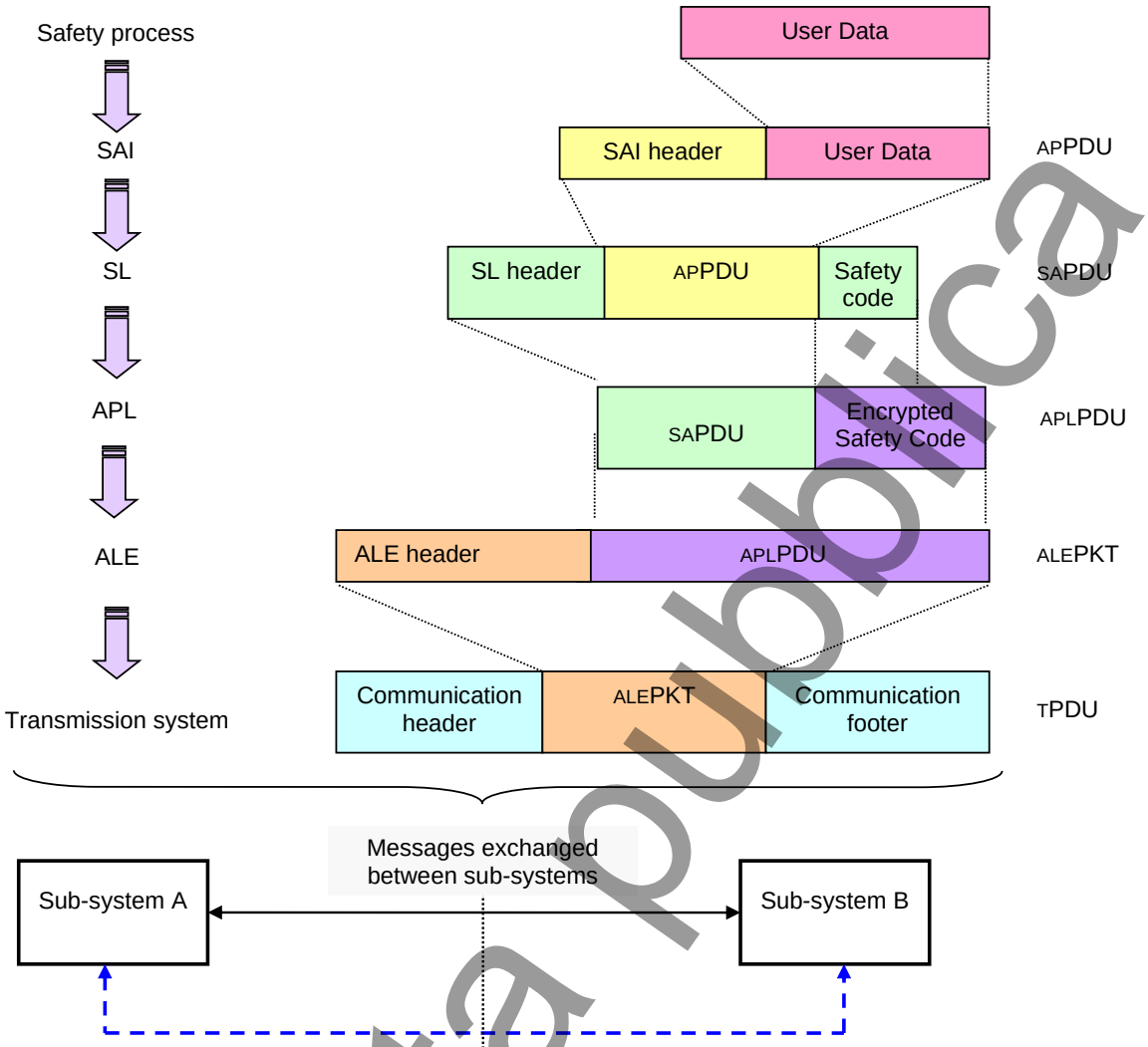


Figure 7-1 – PDU encapsulation process

7.2 ALE Layer

7.2.1 Functionalities

[Req_ALE_001]

The ALE layer shall implement either the UDP protocol (see Ref.6) or the TCP protocol (see Ref.7).

The selection shall be done at configuration level.

UDP is the recommended choice.

7.2.2 Adaptation Layer Packet (ALEPKT) format

7.2.2.1 General

The packets handled by the ALE are called ALEPKTs and are described below.

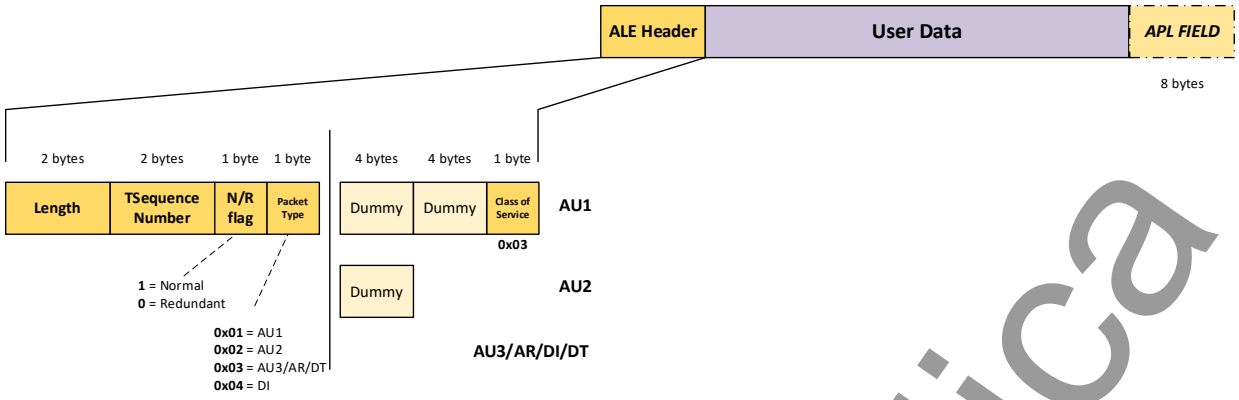


Figure 7-2 - ALE Packets format

[Req_ALE_002]

All ALEPKT fields shall be held in Big Endian byte order.

The format of ALEPKTS shall be as depicted in Figure 7-2.

[Req_ALE_003]

The ALE_HEADER shall be composed of the following fields:

- Packet Length (2 bytes): Length of the entire ALEPKT in bytes, excluding this 2-byte Packet Length field.
- TSequence Number (2 bytes): Transport Sequence Number is used by the receiving Adaptation Layer to manage duplicated frames and switching between two active connections, for communication redundancy purposes.
- N/R Flag (1 byte): it specifies the normal or the redundant links on which the ALEPKTs are sent. The attribution of "normal" and "redundant" to the links is fixed during the configuration phase. The links shall be connected normal-normal and redundant-redundant. N= 1; R=0
- Packet Type (1 byte): the type of ALEPKT.

7.2.2.2 AU1 ALEPKT format

The field dummy (8 bytes) is not used.

[Req_ALE_004]

The field Class of Service of AU1 ALEPKT shall contain the value 3.

Note: the value 3 corresponds to Class D, see Ref.2.

[Req_ALE_005]

Packet Type field of AU1 ALEPKT shall be 1.

[Req_ALE_006]

The packet AU1 ALEPKT shall contain in the SaPDU the AU1 SaPDU conforming to the format specified in Table 2 – AU1 SaPDU structure.

[Req_ALE_007]

The APL field (8 bytes) in the AU1 ALEPKT shall be present only if the APL layer is activated.

7.2.2.3 AU2 ALEPKT format

The field dummy (4 bytes) is not used.

[Req_ALE_008]

Packet Type field of AU2 ALEPKT shall be 2.

[Req_ALE_009]

The packet AU2 ALEPKT shall contain in the SaPDU the AU2 SaPDU conforming to the format specified in Table 4 – AU2 SaPDU structure.

[Req_ALE_010]

The APL field (8 bytes) in the AU2 ALEPKT shall be present only if the APL layer is activated.

7.2.2.4 DT ALEPKT format

[Req_ALE_011]

Packet Type field of DT ALEPKT shall be 3

[Req_ALE_012]

The packet DT ALEPKT shall contain in the SaPDU the AU3/AR/Data SaPDU.

[Req_ALE_013]

The APL field (8 bytes) in the DT ALEPKT shall be present only if the APL layer is activated.

7.2.2.5 DI ALEPKT format

[Req_ALE_014]

The format of DI ALEPKT shall be as described in Figure 7-2.

The Packet Type field of DI ALEPKT shall be 4.

7.3 APL - Access Protection Layer

7.3.1 Functionalities

APL provides cryptographic functions as a protection with respect to masquerade for vital messages transmitted on an open network and can be placed on a safe equipment or on a non-safe equipment (see §A.2 in Ref.1). In the last case, it is necessary to introduce diagnostic mechanisms of the APL.

The encryption techniques used are two, used in succession:

1) AES [Ref.4], which accepts 128-bit long data blocks as input, but which however transforms the input data making them unreadable and therefore making it difficult to control traffic on the network. The permissible lengths for the encryption key are 192 or 256 bits.

2) AES-CMAC [Ref.10], which accepts a message of any length as input and supplies a 128-bit long code as output.

7.3.2 APL – Reference architecture

A Safe communication based on open or closed network compliant with EN 50159 is possible with two different architectures.

1) APL not implemented in the endpoints, but in external non safety related computers (Figure 7-3);

2) APL implemented in the endpoints (Figure 7-4).

In case 2) the APL may be implemented either in the safety related part or in the non-safety related part of the endpoints.

Depending on whether the services of the APL are used or not, messages B1 or A0 will be used respectively.

If the APL is implemented in a separated non safety related computer (case 1)) or in the non-safety related part of endpoints (case 2)) it shall be checked by safety techniques.

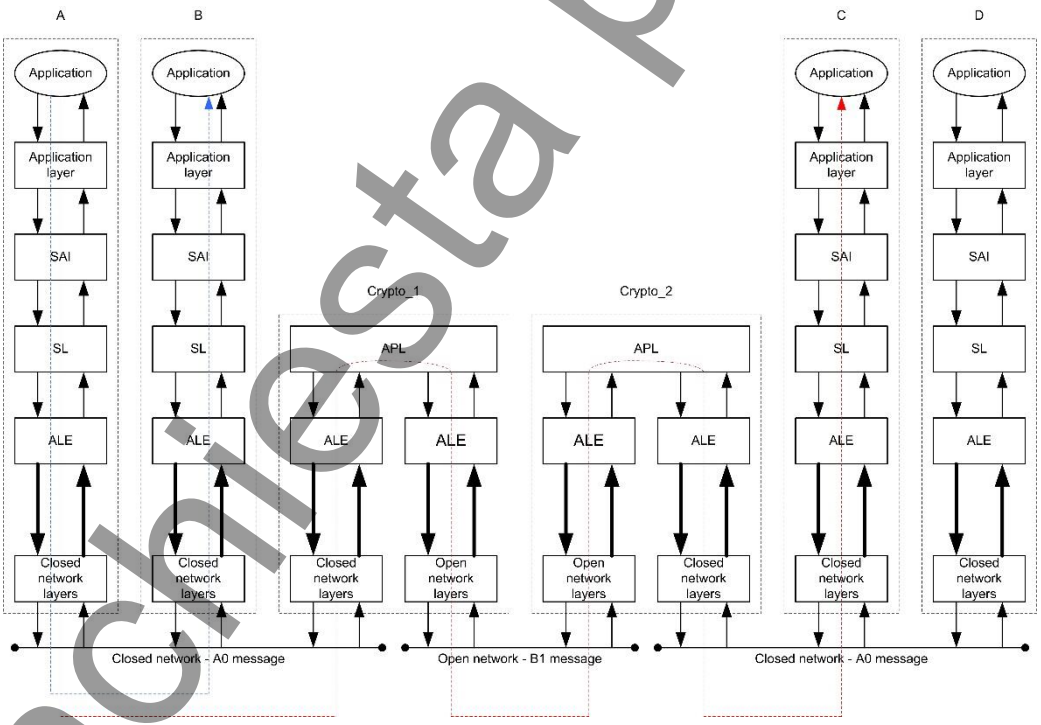


Figure 7-3 – Architecture example (Case 1)

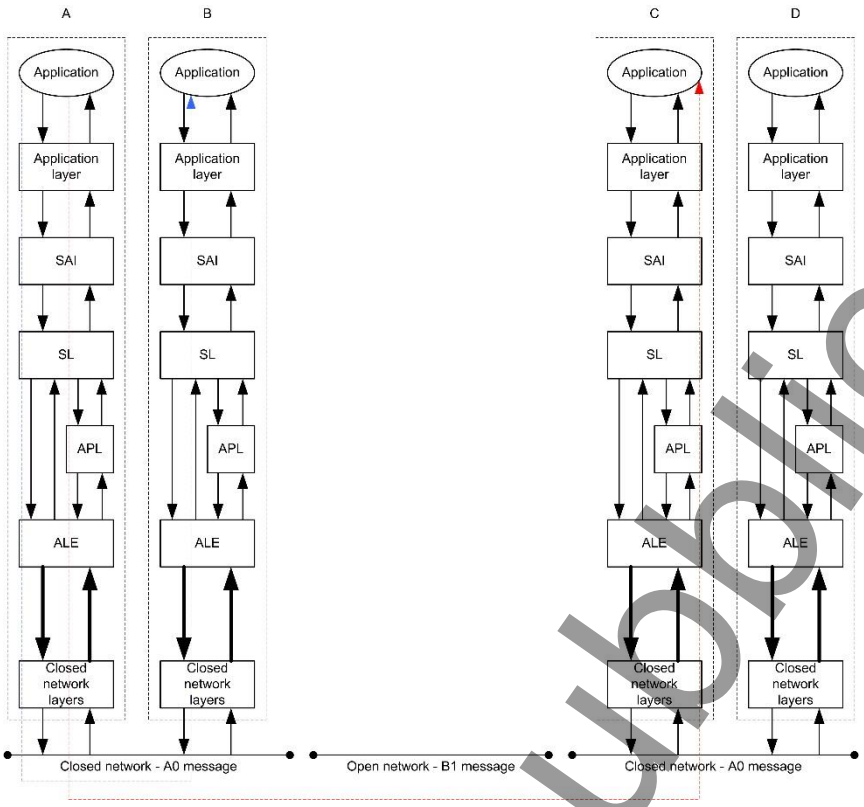


Figure 7-4 – Architecture example (Case 2)

Figure 7-3 and Figure 7-4 depicts a model structured as follows: four safety machines, A, B, C and D, are allocated two by two on different closed networks (up to Category 2). These networks are interconnected via an open network (Category 3). For communications internal to closed networks model A0 is sufficient, while model B1 is needed for open networks communications.

7.3.3 APL – requirements

[Req_APL_001]

When the APL is configured, its correct behaviour shall be verified; in case of failure the connection shall be released.

The main functions of APL to be tested shall be:

- the transmitter duplication of last 8 bytes of frames and encrypting, with configured key;
- the receiver decrypting, with configured key, of last 16 bytes of frames;
- the equality in groups of 8 of the decrypted last 16 bytes;
- the transmitter AES-CMAC computation of the whole frame except the last 8 bytes;
- the receiver AES-CMAC computation of the whole frame except the last 16 bytes.

Encrypting and decrypting

The chosen cryptographic techniques are AES (Advanced Encryption Standard, see Ref.4), it is a symmetric-key encryption standard with a fixed data block size of 128 bits and a key size of 192 or 256 bits and the AES-CMAC (Advanced Encryption Standard - Cipher-based Message Authentication Code, see Ref.10) with a key size of 128 bits.

[Req_APL_002]

If configured, APL shall use AES and AES-CMAC.

[Req_APL_003]

The admitted values for key size length shall be:

- 192 or 256 bits for AES;
- 128 bits for AES-CMAC.

7.4 SL Layer

7.4.1 Functionalities

SL ensures:

- Message integrity using a non-cryptographic safety code
- Authenticity using source identifiers, named nSaCEPID

7.4.2 Source identifiers (nSaCEPID)

nSaCEPID are used to safely identify the connection.

[Req_SL_001]

If a subsystem communicates with n remote subsystems:

- Two "n" connection identifier shall be provided: "n" nSaCEPID-LOC (local connection identifier) and "n" nSaCEPID-REM (remote connection identifier)
- All nSaCEPID shall be different.
- For each instance (connection) of a subsystem, a nSaCEPID-LOC shall be assigned univocally and this values shall be known only to the instance itself and to the remote instance to communicate.

The identification of the source using nSaCEPID is achieved by making the Safety Code of a packet to transmit function of nSaCEPID-REM. In this way only the receiver whose configuration contains that value will be able to verify the integrity of the packet and process it.

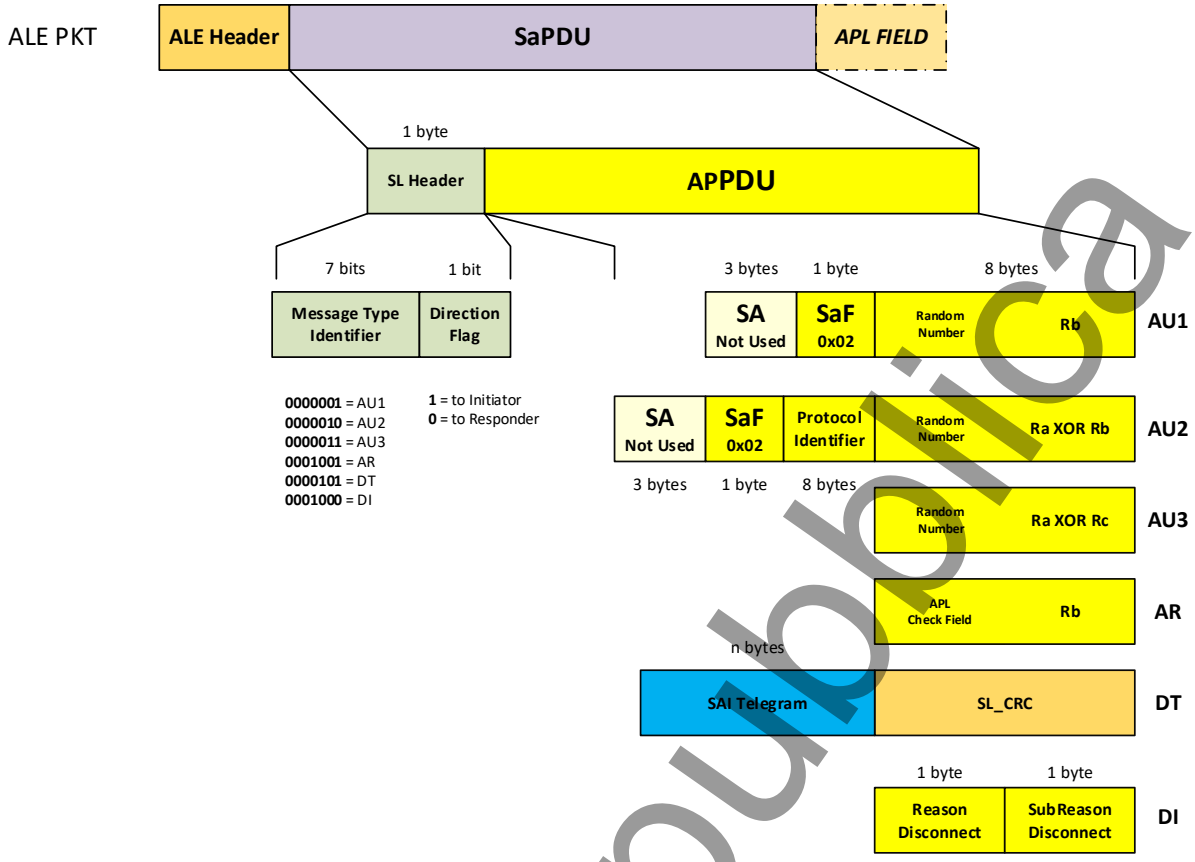
7.4.3 SaPDU types

7.4.3.1 General

The Message Types Identifiers used are the ones depicted in Figure 7-5 (they are inherited from Subset-037 Ref.2) the SaPDU may be grouped in three different sets, the first identifies the frames sent only by the Initiator, the second one those sent only by the Responder and the last one the frames sent by both, as shown in Table 1.

Message	Message Type Identifiers	Specific Initiator frames	Specific Responder frames	Common frames
AU1 (First Authentication SaPDU)	0001	X		
AU2 (Second Authentication SaPDU)	0010		X	
AU3 (Third Authentication SaPDU)	0011	X		
AR (Response to AU3 SaPDU)	1001		X	
DT (Data SaPDU)	0101			X
DI (Disconnect SaPDU)	1000			X

Table 1 – SaPDU types



401

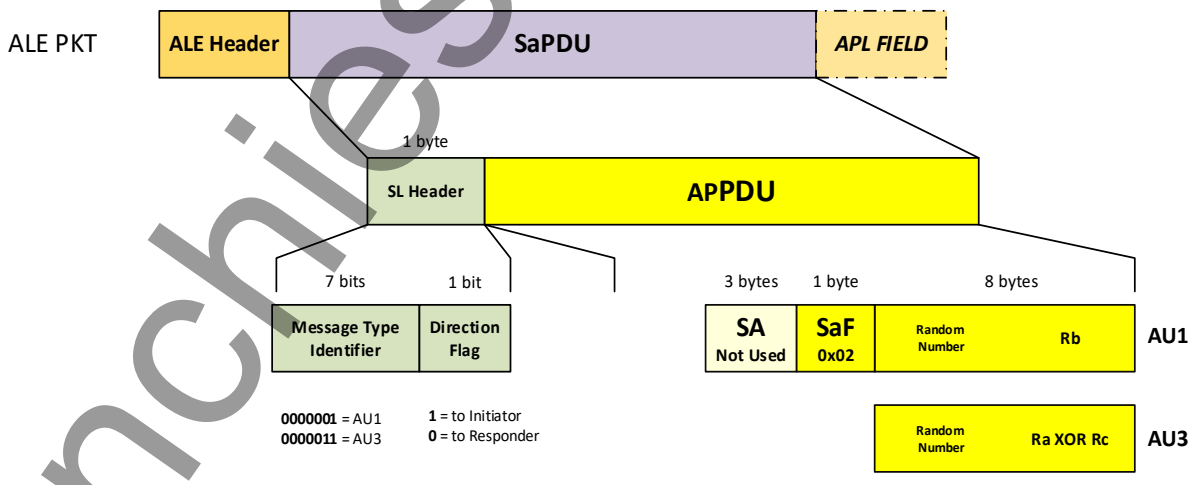
402 **Figure 7-5 – SaPDU**

403 **[Req_SL_002]**

404 All fields of SaPDUs shall be coded using big endian data representation.

405 **7.4.3.2 SaPDU specific Initiator frames**

406 The SaPDU_specific_Initiator_frames are the PDU sent only by the Initiator (see Figure 7-6).



407

408 **Figure 7-6 – SaPDU initiator frames**

409 The t_DirectionFlag is 1 bit when '0' indicates the direction to the Responder, when '1' indicates
410 the direction to the Initiator.

7.4.3.3 AU1

The AU1 is sent by the Initiator to initialize the communication.

[Req_SL_003]

AU1 SaPDU format shall be as described in Table 2.

Name	Description
MessageTypeIdentifier	Indicates the AU1, it is 7 bits and equal to 0000001.
DirectionFlag	It is equal to '0' because the AU1 is to Responder.
SA	3 unused bytes fixed to 0.
SaF	Equal to 00000010
Rb	Rb random number of the first authentication message generated by the Initiator.

Table 2 – AU1 SaPDU structure

7.4.3.4 AU3

The AU3 is sent by the Initiator after the reception of AU2 from the Responder.

[Req_SL_004]

AU3 SaPDU format shall be as described in Table 3 – AU3 SaPDU structure.

Name	Description
MessageTypeIdentifier	Indicates the AU3, it is 7 bits and equal to 0000011
DirectionFlag	It is equal to '0' because the AU3 is to Responder.
RaRc	RaRc field contains random number Ra XOR Rc; it is generated by the Initiator.

Table 3 – AU3 SaPDU structure

7.4.3.5 SaPDU specific Responder frames

The SaPDU specific Responder frames are the frames sent only by the Responder (see Figure 7-7).

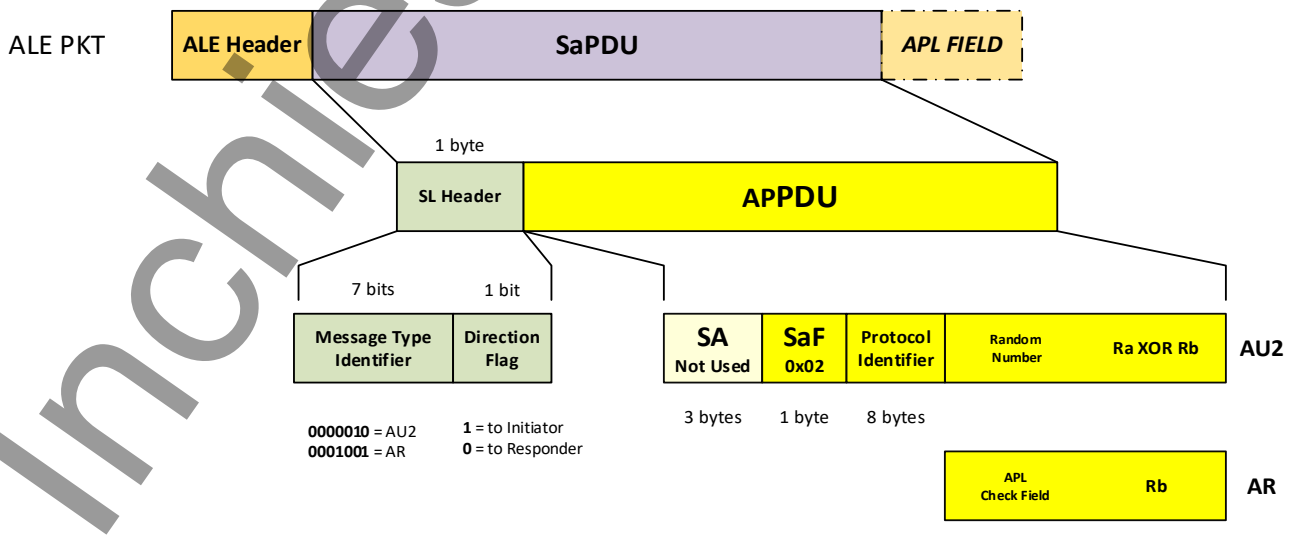


Figure 7-7 - SaPDU Responder frames

7.4.3.6 AU2

The AU2 is sent by the Responder to answer to AU1.

[Req_SL_005]

AU2 SaPDU format shall be as described in Table 4 – AU2 SaPDU structure.

Name	Description
MessageTypeIdentifier	Indicates the AU2 it is 7 bits and equal to 0000010.
DirectionFlag	It is equal to '1' because the AU2 is to Initiator.
SA	3 unused bytes fixed to 0.
SaF	Equal to 00000010
ProtID	Fixed value that identifies the protocol 0x2247524154495322. The receiver (Initiator) must not check this value
Ra	Ra random number Ra XOR Rb; it is generated by the Responder.

Table 4 – AU2 SaPDU structure

7.4.3.7 AR

The AR is sent by the Responder after the reception of AU3.

This frame ends the initial session handshake.

[Req_SL_006]

AR SaPDU format shall be as described in Table 5 – AR SaPDU structure.

Name	Description
MessageTypeIdentifier	Indicates the AR, it is 7bits and equal to 0001001.
DirectionFlag	It is equal to '1' because the AR is to Initiator.
AplCheckField	The AplCheckField contains the random number Rb; this field is used to diagnose the APL behaviour.

Table 5 – AR SaPDU structure

7.4.3.8 SaPDU_Common_Frames

The SaPDU_Common_Frames are the frames that can be sent by the Initiator or by the Responder in accordance with their internal state (see Figure 7-8).

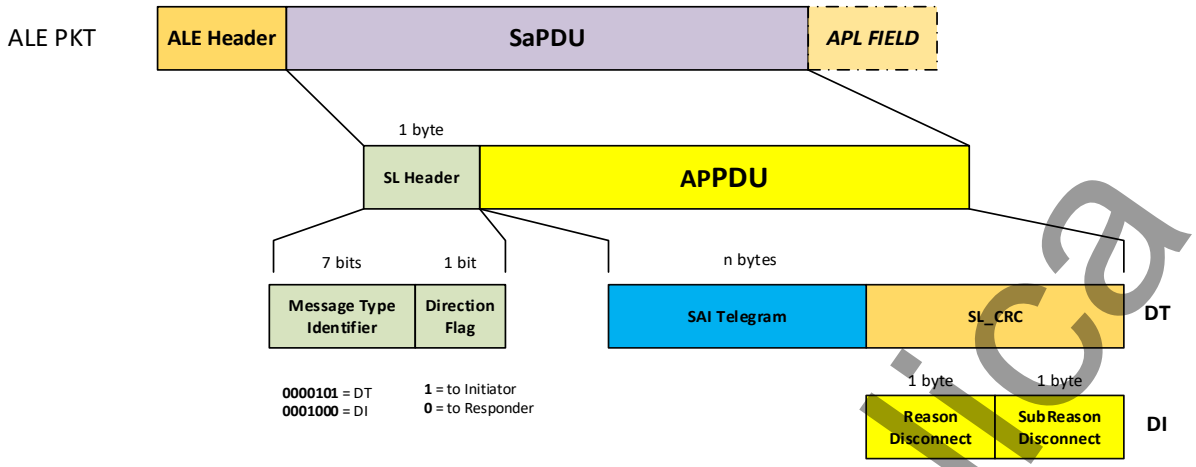


Figure 7-8 – SaPDU Common frames

7.4.3.9 DI - Disconnect frame

The DI SaPDU is sent to close the session for example after errors detection.

[Req_SL_007]

SaPDU Disconnect (DI SaPDU) format shall be as described in Table 6 – SaPDU Disconnect structure.

Name	Description
MessageTypeIdentifier	Indicates the DI, it is 7 bits and equal to 0001000.
DirectionFlag	When '0' indicates the direction to the Responder, when '1' indicates the direction to the Initiator.
ReasonField	The ReasonField indicates the reason of disconnection.
SUB-ReasonField	The SUB-ReasonField is the sub-reason of the disconnect.

Table 6 – SaPDU Disconnect structure

7.4.3.10 DT - Data frame

[Req_SL_008]

Data frame SaPDU (DT SaPDU) format shall be as described in Figure 7-8.

The SaPDU Header format shall be as described in Table 7.

Name	Description
MessageTypeIdentifier	Indicates the Data frame, it is 7 bits and equal to 0000101.
DirectionFlag	When '0' indicates the direction to the Responder, when '1' indicates the direction to the Initiator.

Table 7 – SaPDU_Header structure

The SAI telegram is the frame that contains the information generated by SAI layer.

The SL_CRC contains the check-field computed using to verify the integrity of the frame.

[Req_SL_009]

In the SL_CRC (Safety Code) computations the following polynomials shall be used:

- Channel 1: x^32+x^28+x^19+x^18+x^16+x^14+x^11+x^10+x^9+x^6+x^5+x+1
- Channel 2: x^32+x^31+x^27+x^26+x^23+x^22+x^21+x^18+x^16+x^14+x^13+x^4+1.

The result of CRC computation shall be XOR-ed with Ra or Rc depending on the PDU generator: Responder or Initiator.

[Req_SL_010]

Indicating with CRC1 the CRC calculated with the polynomial of channel 1 and with CRC2 the CRC calculated with the polynomial of channel 2, Safety code shall be composed of CRC1 (byte 0..3) and CRC2 (byte 4..7).

For further details, see §8.3.2 and 8.3.2.2.

7.5 SAI Layer

7.5.1 Functionalities

The SAI layer has two different and not compatible implementations:

- the Monotonic Increasing Sequences (MIS) option, or
- the Pseudo Random Sequences (PRS) option.

In order o represent variables associated to the sequence numbers and the execution cycles, the MIS option makes use of cardinal sequence of integers (integer representation) while the PRS option adds a second representation by means of pseudo-random sequences (pseudo-random representation).

It must be said that both the options implement the same defences with different modalities. However, the Pseudo Random Sequences (PRS) is the standard solution whilst the Monotonic Increasing Sequences (MIS) will remain as an option. This is because the PRS option has the advantage to have a not too complex implementation both in composite and coded safety architectures, therefore being a more general option.

Nevertheless, the Monotonic Increasing Sequences option can be used for a specific project, if agreed by the relevant parties to the contract. Please note that before using the MIS option the constraints listed in §Annex E must be satisfied.

[Req_SAI_001]

SAI layer shall implement the following mechanisms of defence:

- A Sequence Number (SN) that ensures the protection against resequencing, repetition, insertion and deletion.
- An Execution Cycle (EC) that ensures the freshness and protection against delay.

[Req_SAI_002]

The SAI layer defence mechanisms shall be based on

- EC, SN integer values (MIS option), or
- EC, SN integer values and PR-EC, PR-SN pseudo-random values (PRS option)

7.5.2 EC and SN (integer representation)

[Req_SAI_003]

The SN counter shall be coded on 2 bytes (big Endian).

[Req_SAI_004]

The sequence number shall assume an integer value from 0 to 2^{16}-1 (65535).

SN shall be initialized with any value belonging to this range.

[Req_SAI_005]

The sequence numbers of the two communicating nodes shall be independent.

In the same node, two or more instances of PVS shall have different sequence numbers.

[Req_SAI_006]

The EC counter shall be coded on 4 bytes (big Endian).

[Req_SAI_007]

The EC counter shall assume an integer value from 0 to 2^{32}-1 (4294967295).

EC shall be initialized with any value belonging to this range.

[Req_SAI_008]

The EC counters of the two communicating nodes shall be independent.

In the same node, two or more instances of PVS may share the same EC counter.

[Req_SAI_009]

After the first initialisation, the values of Sequence Number and Execution Cycle shall evolve as follows:

- If the Sequence Number is different of the maximum value, the Sequence Number shall be incremented by one at each new message transmission to the other node.
- Once the Sequence Number reaches the maximum value, the value of the Sequence Number at the next message transmission shall be set to "0"zero.
- If the EC counter is different from the maximum value, the EC counter shall be incremented by one at each new execution cycle of the sender.
- Once the EC counter reaches the maximum value, the value of the EC counter at the next cycle shall be set to "0" zero.

7.5.3 PR-EC and PR-SN (pseudo random representation)

The two pseudo random numbers: Pseudo Random Execution Cycle (PR-EC) and Pseudo Random Sequence Number (PR-SN), are based on pseudo-random maximum length sequences.

[Req_SAI_010]

Each one of the pseudo random numbers (PR-SN and PR-EC) shall be constituted by a couple of 32-bit values:

- PR-SN = (PR-SN$_1$, PR-SN$_2$)
- PR-EC = (PR-EC$_1$, PR-EC$_2$)

[Req_SAI_011]

Each 32-bit value, composing one half of the pseudo random numbers (PR-SN and PR-EC), shall be the element of a pseudo-random maximum length sequence encoded on 32 bits.

As such, they shall be initialized with any non-zero 32-bit values.

A pseudo-random maximum length sequence can be generated with a 32-bit maximum-length Linear Feed-back Shift Register (LFSR, see Figure 7-9) that implements the following functions:

- Store the value in a 32-bit register.
- Shift of 32 positions with a proper feedback mask.

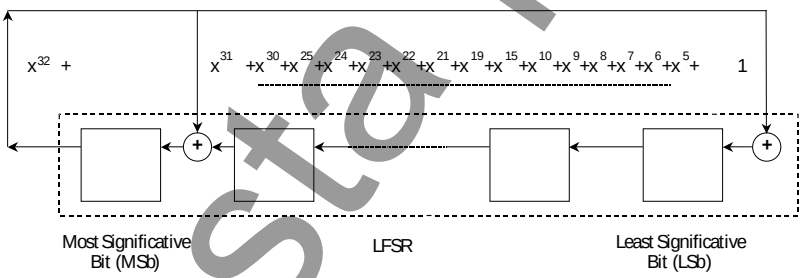


Figure 7-9 - Example of generation of ML-sequence using 32-bit LFSR

The feedback mask is typically identified with a polynomial, as shown in Figure 7-9.

The software implementation is similar to CRC calculation, using a tabular representation of intermediate feedbacks. This operation, indicated as LFSR(X), consists in loading the value X in the shift register with feedback and shifting of 32 positions with feedback.

Since two different polynomials will be used (see Req_SAI_012), the following notation is given: "LFSRy (X)", where y assumes the values 1 or 2.

A maximum-length LFSR produces a ML-sequence (i.e., it cycles through all possible $2^m - 1$ states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change; in this case m is equal to 32, then 4294967295 (2^{32}-1) states.

[Req_SAI_012]

Each LFSR processing operation shall be executed separately on each 32-bit-long element of the couple constituting the pseudo random number.

Two different polynomials shall be used:

$LFSR_1 : x^{32}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{17}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^7+x^2+x+1$

$LFSR_2 : x^{32}+x^{31}+x^{30}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{19}+x^{15}+x^{10}+x^9+x^8+x^7+x^6+x^5+1$

where $LFSR_1$ operates on the first element of the couple (see Req_SAI_010), and $LFSR_2$ operates on the second element of the couple.

NOTE Therefore, the numerical evolution of PR-SNy and PR-ECy (with y being 1 or 2) belong to the same pseudo random sequence (see Figure 7-10).

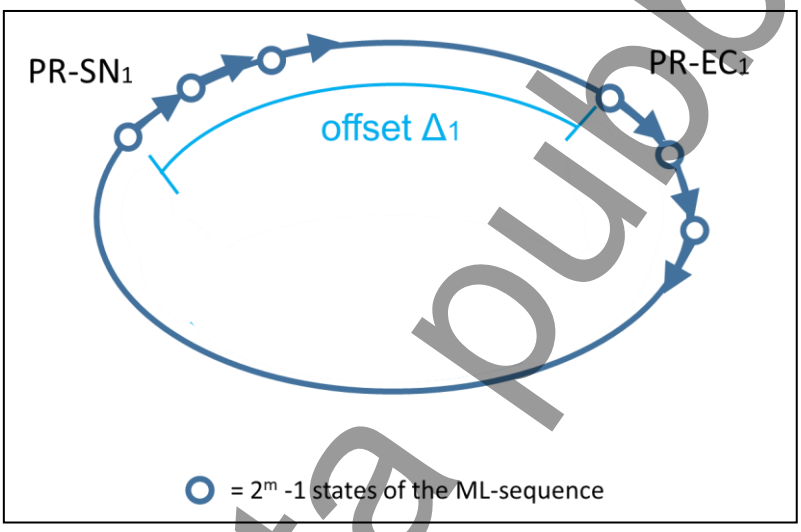


Figure 7-10 – ML-sequences

[Req_SAI_013]

Each field (fields size equal to 8 bytes) of the SAI ApPDUs containing pseudo random numbers shall be composed as follows:

Low part (byte 0..3): element of the couple on which $LFSR_1$ is used

High part (byte 4..7): element of the couple on which $LFSR_2$ is used

[Req_SAI_014]

PR-SN and PR-EC shall be initialized with different values.

The offset Δ (the ordinal distance in the pseudo-random sequence, see Figure 7-10) between the values ($PR\text{-}EC_1$ and $PR\text{-}SN_1$) shall be different from that one of the values ($PR\text{-}EC_2$ and $PR\text{-}SN_2$):

$\Delta_1(PR\text{-}EC_1, PR\text{-}SN_1) \neq \Delta_2(PR\text{-}EC_2, PR\text{-}SN_2)$

NOTE The aim of the above requirement is to avoid that, at a certain cycle, $\Delta_1 = \Delta_2 = 0$.

[Req_SAI_015]

PR-EC shall be initialized with any couple of non-zero 32-bit values, with the following constraint: the initialisation values shall be univocally assigned among all the devices connected to the network.

NOTE For example, PR-EC can be initialized to nSaCEPID-LOC. In case of subsystem with more than one communication flows (and so different nSaCEPID) it is sufficient to use a couple of values.

[Req_SAI_016]

After the first initialisation, the pseudo random values of PR-SN and PR-EC shall evolve as follows and shall not be initialised again (the counters assume all the possible 2^{32}-1 values).

Given "s" a generic element of the ML-sequence,

- PR-SNx (s+1) = LFSRx [PR-SNx (s)] at each new message transmission to the orther node
- PR-ECx (s+1) = LFSRx [PR-ECx (s)] at each new execution cycle of the sender.

NOTE The integer representations of SN and EC are incremented at the same time of their pseudo-random versions, see Req_SAI_009.

7.5.4 SAI frames

SAI frames are depicted in Figure 7-11 (PRS option) and Figure 7-12 (MIS option).

[Req_SAI_017]

All fields of SAI ApPDUs shall be coded using big endian data representation.

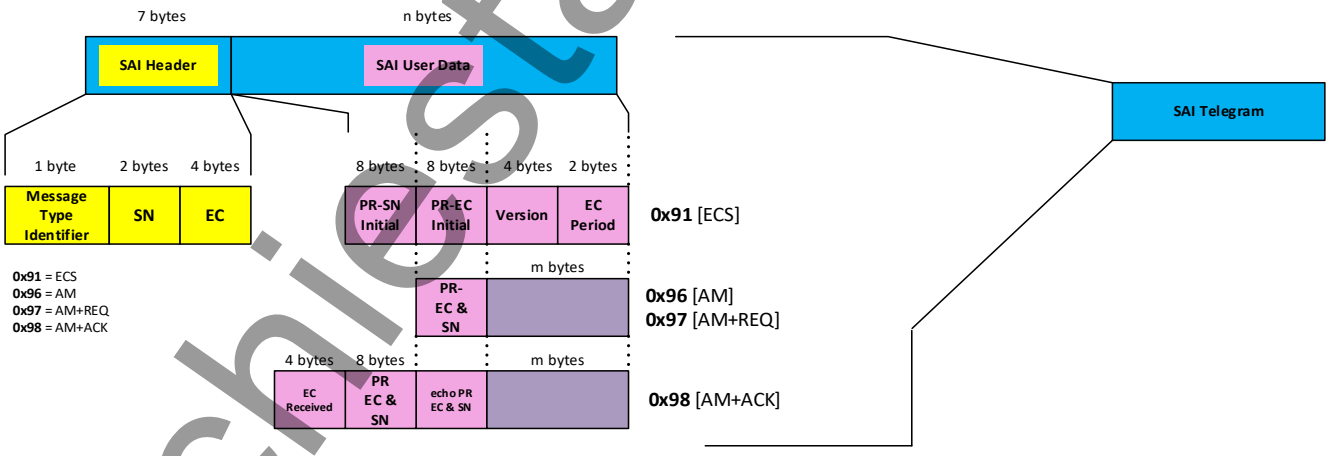


Figure 7-11 - SAI frames – PRS option

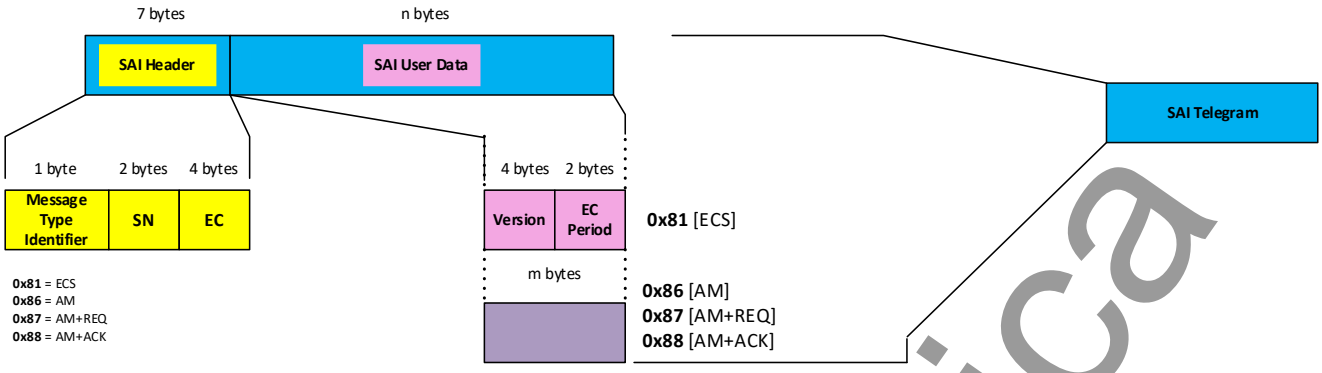


Figure 7-12 – SAI frames – MIS option

7.5.5 SAI Header

The SAI Header contains the field common to all SAI frames.

[Req_SAI_018]

SAI header shall be as described in Table 8.

Name	Description	Size (bytes)
MessageTypeField	The MessageType field identifies the type of SAI frame. ExecutionCycleStart (ECStart): 0x91 (PRS) / 0x81 (MIS) Application Message (AM): 0x96 (PRS) / 0x86 (MIS) Application Message with Request of ACK (AM+REQ): 0x97 (PRS) / 0x87 (MIS) Application Message with ACK (AM+ACK): 0x98 (PRS) / 0x88 (MIS)	1
SequenceNumber	The SequenceNumber field is 2 bytes. It contains the sender Sequence Number Counter.	2
ECCounter	The ECCounter field contains the sender Execution Cycle Number .	4

Table 8 – SAI header

7.5.6 SAI ApPDUs format – PRS option

7.5.6.1 ECStart – ApPDU 0x91 (PRS)

The ECStart frame is sent at connection establishment.

In the SAI_Header the field MessageTypeField is equal to 91h.

[Req_SAI_019]

The ECStart PDU shall be composed of the SAI header plus the fields described in Table 9.

Name	Description	Size (bytes)
PR-SN Initial	The PR-SN Initial field contains the PR-SN 1 XOR nSaCEPID1 and the PR-SN 2 XOR nSaCEPID2.	8
PR-EC Initial	The PR-EC Initial field contains the PR-EC 1 XOR nSaCEPID1 and the PR-EC-2 XOR nSaCEPID2.	8
Version	The Version field is equal to 2. It will be modified only after significant modifications of protocol specification.	4
ExecutionCyclePeriod	The ExecutionCyclePeriod is the sender transmission period (in ms).	2

Table 9 – ECStart fields - PRS option

[Req_SAI_020]

In the ECStart message from initiator:

- PR-SN initial shall be: $PR\text{-}SN_{initiator}$ XOR $nSaCEPID_{initiator}$
- PR-EC initial shall be: $PR\text{-}EC_{initiator}$ XOR $nSaCEPID_{initiator}$

[Req_SAI_021]

In the ECStart message from responder:

- PR-SN initial shall be: $PR\text{-}SN_{responder}$ XOR $nSaCEPID_{responder}$
- PR-EC initial shall be: $PR\text{-}EC_{responder}$ XOR $nSaCEPID_{responder}$

With the reception of Execution Cycle Start, the node is able to know the remote integer and pseudo random values of the Sequence Number and the Execution Cycle.

7.5.6.2 AM ApPDU 0x96 (PRS)

The AM frame is used to send UserData.

In the SAI_Header the field MessageTypeField is equal to 96h.

[Req_SAI_022]

The AM ApPDU shall be composed of the SAI header plus the fields described in Table 10 and depicted in

Figure 7-13.

Name	Description	Size (bytes)
PR-EC&SN	The PR-EC&SN field contains: - PR-EC 1 XOR PR-SN 1 XOR nSaCEPID1 - PR-EC 2 XOR PR-SN 2 XOR nSaCEPID2	8
UserData	The Application Data	n

Table 10 – AM specific fields – PRS option

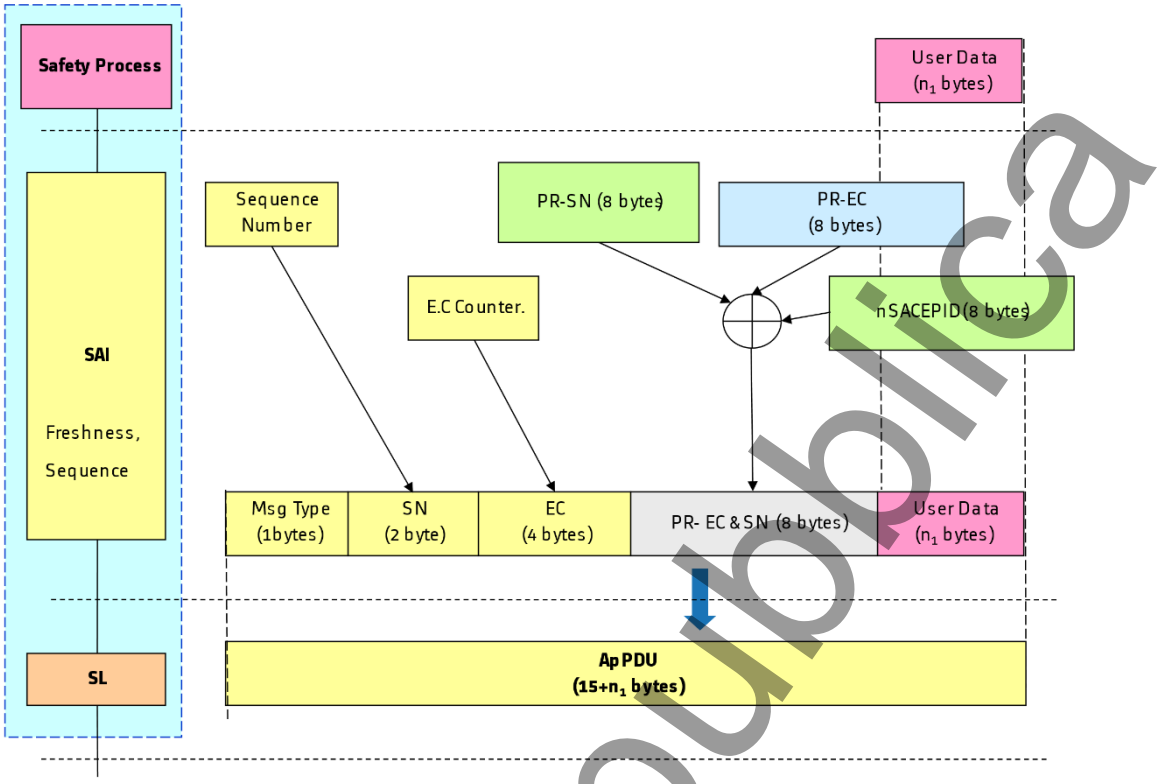


Figure 7-13 – AM, AM+REQ ApPDU (PRS option)

7.5.6.3 AM+REQ ApPDU 0x97 (PRS)

The AM+REQ frame is used to send UserData and to ask a re-alignment.

In the SAI Header the field MessageTypeField is equal to 97h.

[Req_SAI_023]

The AM+REQ PDU shall be composed of the SAI header plus the fields described in Table 11:

Name	Description	Size (bytes)
PR-EC&SN	The PR-EC&SN field contains: - PR-EC 1 XOR PR-SN 1 XOR nSaCEPID1 - PR-EC 2 XOR PR-SN 2 XOR nSaCEPID2	8
UserData	The Application Data (n bytes)	n

Table 11 – AM+REQ specific fields – PRS option

7.5.6.4 AM+ACK ApPDU 0x98

The AM+ACK frame is used to send UserData and to answer to AM+REQ frame.

In the SAI Header the field MessageTypeField is equal to 98h.

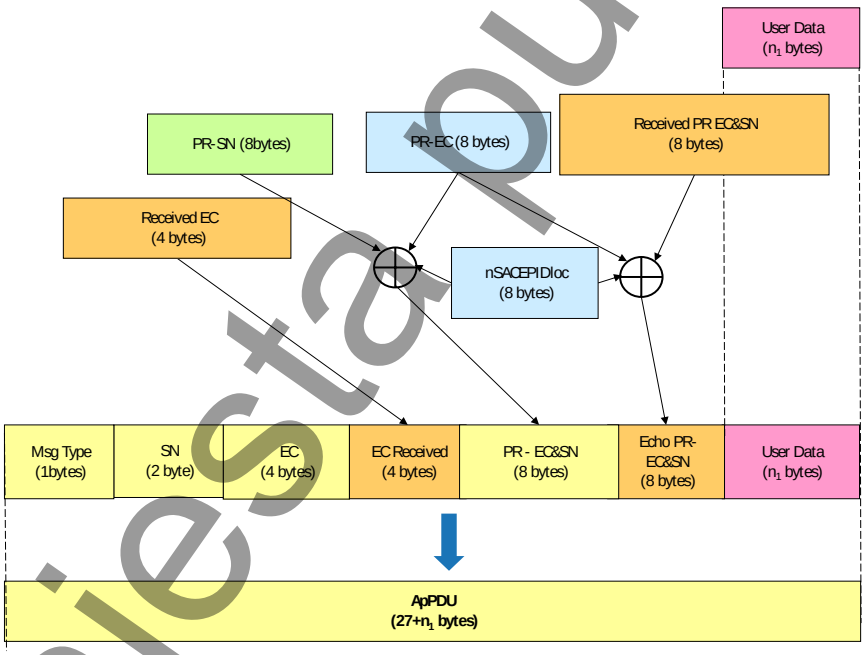
637 **[Req_SAI_024]**

638 The AM+ACK ApPDU shall be composed of the SAI header plus the fields described in Table 12 and
639 depicted in Figure 7-14.

640

Name	Description	Size (bytes)
EC received	The EC received field contains the copy of the EC field (SAI Header) received in the frame AM+REQ.	4
PR-EC&SN	The PR-EC&SN field contains: - PR-EC 1 XOR PR-SN 1 XOR nSaCEPID1 - PR-EC 2 XOR PR-SN 2 XOR nSaCEPID2	8
Echo PR EC&SN	The Echo PR EC&SN field contains: - received PR EC&SN 1 field of the AM+REQ XOR PR-EC 1 XOR nSaCEPID1 - received PR EC&SN 2 field of the AM+REQ XOR PR-EC 2 XOR nSaCEPID2	8
UserData	The Application Data (n bytes)	n

641 **Table 12 – AM+ACK specific fields – PRS option**



642

643 **Figure 7-14 – AM+ACK**

644 **7.5.7 SAI ApPDUs format – MIS option**

645 **7.5.7.1 ECStart – ApPDU 0x81 (MIS)**

646 The ECStart frame is sent at connection establishment.

647 In the SAI_Header the field MessageTypeField is equal to 81h.

[Req_SAI_025]

The ECStart PDU shall be composed of the SAI header plus the fields described in Table 9.

Name	Description	Size (bytes)
Version	The Version field is equal to 2. It will be modified only after significant modifications of protocol specification-	4
ExecutionCyclePeriod	The ExecutionCyclePeriod is the sender transmission period (in ms).	2

Table 13 – ECStart fields – MIS option

7.5.7.2 AM ApPDU 0x86 (MIS)

The AM frame is used to send UserData.

In the SAI_Header the field MessageTypeField is equal to 86h.

[Req_SAI_026]

The AM PDU shall be composed of the SAI header plus the fields described in Table 10:

Name	Description	Size (bytes)
UserData	The Application Data	n

Table 14 - AM specific fields – MIS option

7.5.7.3 AM+REQ ApPDU 0x87 (MIS)

The AM+REQ frame is used to send UserData and to ask a re-alignment.

In the SAI Header the field MessageTypeField is equal to 87h.

[Req_SAI_027]

The AM+REQ PDU shall be composed of the SAI header plus the fields described in Table 11:

Name	Description	Size (bytes)
UserData	The Application Data (n bytes)	n

Table 15 – AM+REQ specific fields – MIS option

7.5.7.4 AM+ACK ApPDU 0x88 (MIS)

The AM+ACK frame is used to send UserData and to answer to AM+REQ frame.

In the SAI Header the field MessageTypeField is equal to 88h.

[Req_SAI_028]

The AM+ACK PDU shall be composed of the SAI header plus the fields described in Table 12:

Name	Description	Size (bytes)
UserData	The Application Data (n bytes)	n

Table 16 – AM+ACK specific fields – MIS option

8 Protocol level –Dynamic description

8.1 Redundancy Management

Every connection between two subsystems is realised through two connections.

The Redundancy Management is managed by the ALE layer in accordance with the following policies.

[Req_ALE_015]

All ALEPKTs are sent on all available connections.

The same TSequence Number shall be transmitted on both connections.

[Req_ALE_016]

Connection Establishment packets AU1 ALEPKT and AU2 ALEPKT complete with User Data shall be sent in an identical way on both the transport connections using the TSequence Number value of "0"

Note: the above requirement means that the TSequence Number has to be initialized to "0" before the connection procedure starts, because AU1 ALEPKT and AU2 ALEPKT are the first packets used to connection establishment.

[Req_ALE_017]

Each subsequent transmission of an ALEPKT shall increase the value of the TSequence Number field by one.

[Req_ALE_018]

At the receiver, a policy to discard duplicated ALEPKTs shall be implemented; the implemented policy, in any case, shall never lead to deadlock conditions of the protocol.

[Req_ALE_019]

When a transmitter is unable to send data on a connection or a connection is deemed in error, the sending of DT ALEPKTs through the other link shall continue. The Adaptation Layer entity that initiated the original connection (the initiator) shall attempt to re-establish any failed connection.

8.2 Creation of a Connection

8.2.1 Connection establishment

The creation of a connection involves all layers (ALE, SL, SAI).

The exchange between entities during the connection establishment phase is as depicted in Figure 8-1.

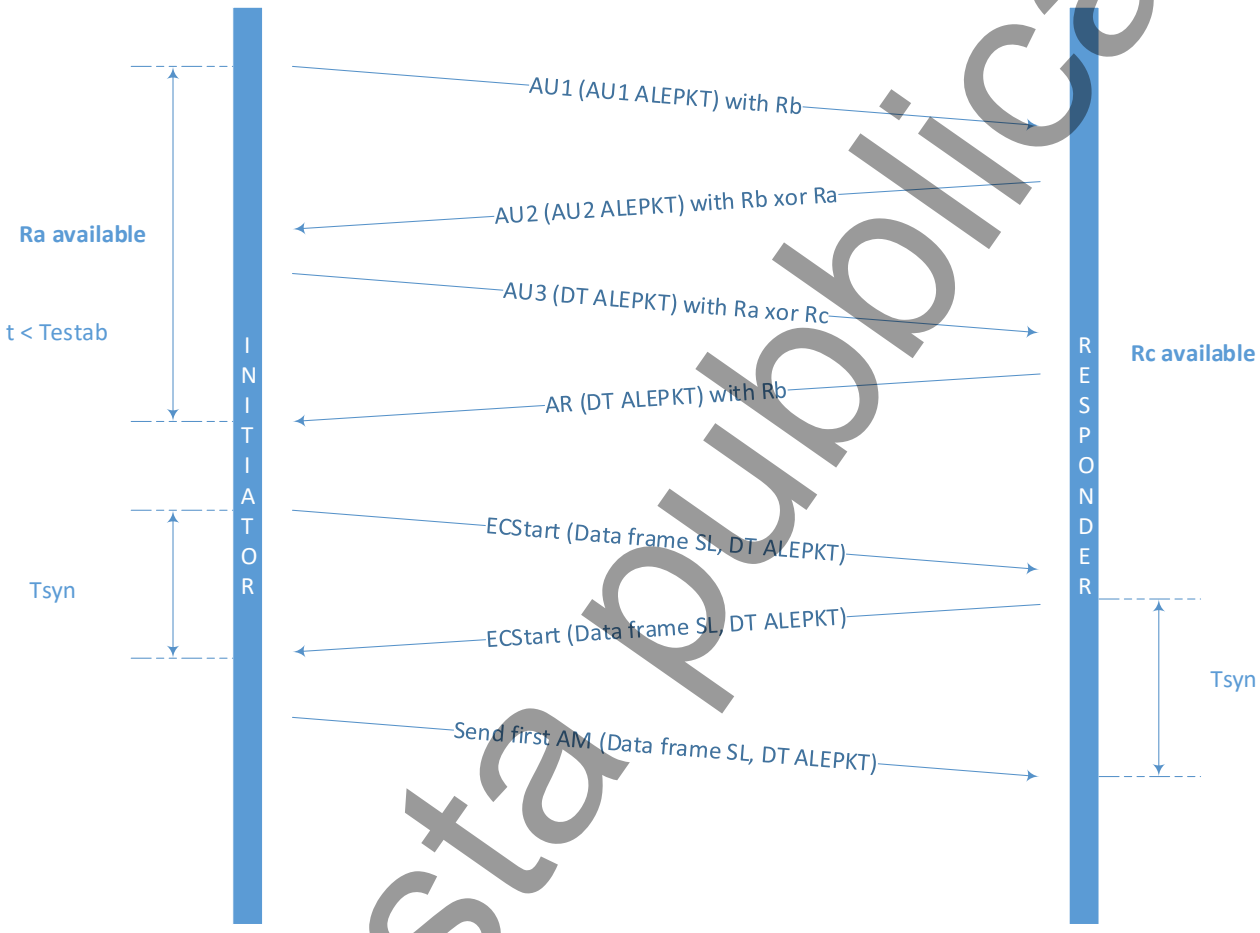


Figure 8-1 - Safe Connection establishment: Detailed protocol sequence

Hereafter an example of protocol State machine to use at connection time has been indicated.

List of states:

Depending on being Initiator or Responder, the list of states is as follows.

1. Initiator:
   - Wait for Request of Connection: it waits for request of connection
   - WaitAU2: it waits for AU2 (second authentication message)
   - WaitAR: it waits for AR (authentication response)
   - WaitECstart: it waits for ECStart
   - Aligned
2. Responder
   - WaitAU1: it waits for AU1 (first authentication message)
   - WaitAU3: it waits for AU3 (third authentication message)

- WaitECstart: WaitECstart: it waits for ECStart
- WaitFirstAM: it waits for First Application Message
- Aligned

List of input events:

Depending on being Initiator or Responder, the list of input events is as follows.

1. Initiator:
   - Received DI: Received disconnection message.
   - Testab>timeout: timeout, for Connection Establishment procedure has expired.
   - Req of Connection: Request of connection
   - Received AU2: Received AU2 (second authentication message)
   - Received AR: Received AR (authentication response)
   - Tsyn expiration: timeout
   - Received ECStart: Received Execution Cycles Start
2. Responder:
   - Received DI: Received disconnection message
   - Rx not expected frame
   - Received AU1: Received AU1 (first authentication message)
   - Received AU3: Received AU3 (third authentication message)
   - Received ECStart: Received Execution Cycles Start
   - Received first AM: Received First Application Message
   - Tsyn expiration: timeout

List of output events:

Depending on being Initiator or Responder, the list of output events is as follows.

1. Initiator:
   - Send DI: it sends disconnection message
   - Send AU1 – run Testab timer: it sends AU1 (first authentication message) and start Testab timer
   - Send AU3: it sends AU3 (third authentication message)
   - Send ECStart and start Tsyn timer: it sends ECStart message and starts Tsyn timer
   - Stop Tsyn and Compute Expected Execution Cycle: it stops Tsyn and computes Expected Execution Cycle
2. Responder:
   - Send DI: it sends disconnection message
   - Send AU2: it sends AU2 (second authentication message)
   - Send AR: it sends AR (authentication response)
   - Start Tsyn and Compute Expected Execution Cycle: it starts Tsyn, computes Expected Execution Cycle and sends ECStart
   - Stop Tsyn: it stops Tsyn

754 Table of transitions:

755 Depending on being Initiator or Responder, the table of transitions is as follows.

756 Initiator:

Current State	Input/Event	Action/Output	Next State
Wait Request of Connection	None	Send AU1- run Testab timer	WaitAU2
	Received DI	None	Wait Request of Connection
	Testab > timeout	Send DI	Wait Request of Connection
WaitAU2	Received AU2	Send AU3	WaitAR
	Received DI	None	Wait Request of Connection
	Testab > timeout	Send DI	Wait Request of Connection
WaitAR	Received AR	Send ECStart, stop Testab and start Tsyn timer	WaitECstart
	Received DI	None	Wait Request of Connection
	Testab > timeout	Send DI	Wait Request of Connection
WaitECstart	Tsyn expiration	Send DI	Wait Request of Connection
	Received ECStart	Stop Tsyn and compute Expected Execution Cycle	Aligned
	Received DI	None	Wait Request of Connection
Aligned	Received DI	None	Wait Request of Connection

757 **Table 17 – Initiator – Finite State Machine - Creation of a Connection**

758 Responder:

Current State	Input/Event	Action/Output	Next State
WaitAU1	Received AU1	Send AU2	Wait AU3
	Received DI	None	WaitAU1
	Rx not expected frame	Send DI	WaitAU1
WaitAU3	Received AU3	Send AR	WaitECstart
	Received DI	None	WaitAU1
	Rx not expected frame	Send DI	WaitAU1
WaitECStart	Received ECstart	Start Tsyn, compute Expected Execution Cycle and Send ECStart	WaitFirstAM
	Received DI	None	WaitAU1
	Rx not expected frame	Send DI	WaitAU1
WaitFirstAM	Tsyn expiration	Send DI	WaitAU1
	Received first AM	Stop Tsyn	Aligned
	Received DI	None	WaitAU1
	Rx not expected frame	Send DI	WaitAU1
Aligned	Received DI	None	WaitAU1

759 **Table 18 – Responder – Finite State Machine - Creation of a Connection**

Note: it is important to remark a difference between Initiator and Responder Finite State Machines, the difference is related to the states in Creation of a Connections:

- for Initiator (Figure 8-2) – a time-out is associated to each state, then if the expected frame is not received the related time-out prevents dead-lock conditions resetting the state machine;
- for Responder (Figure 8-3) – there are states without an associated time-out, then, to prevent dead-lock conditions, the "Rx not expected frame" event is used to reset the state machine.

Initialisation State Machine

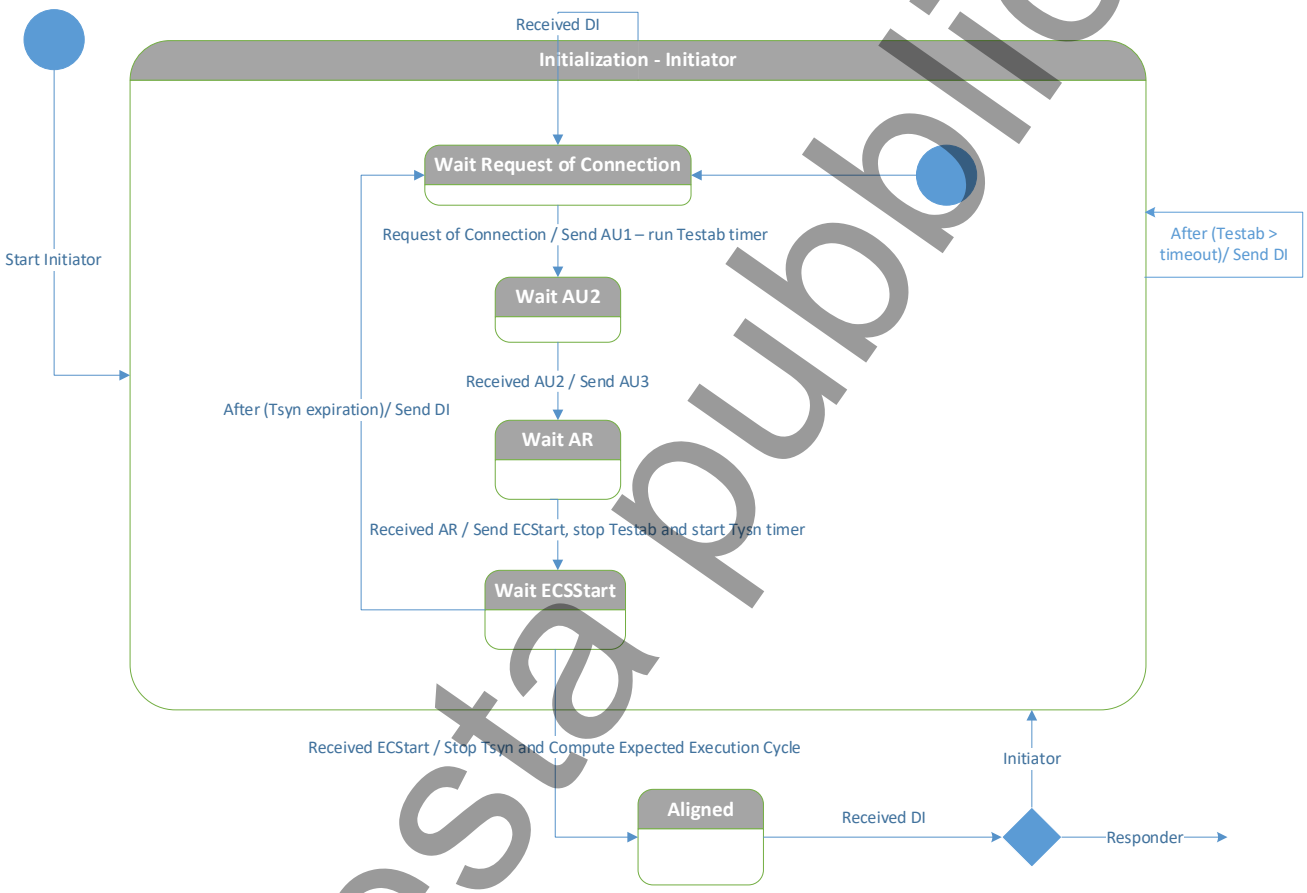


Figure 8-2 - Initialization State Machine – Initiator

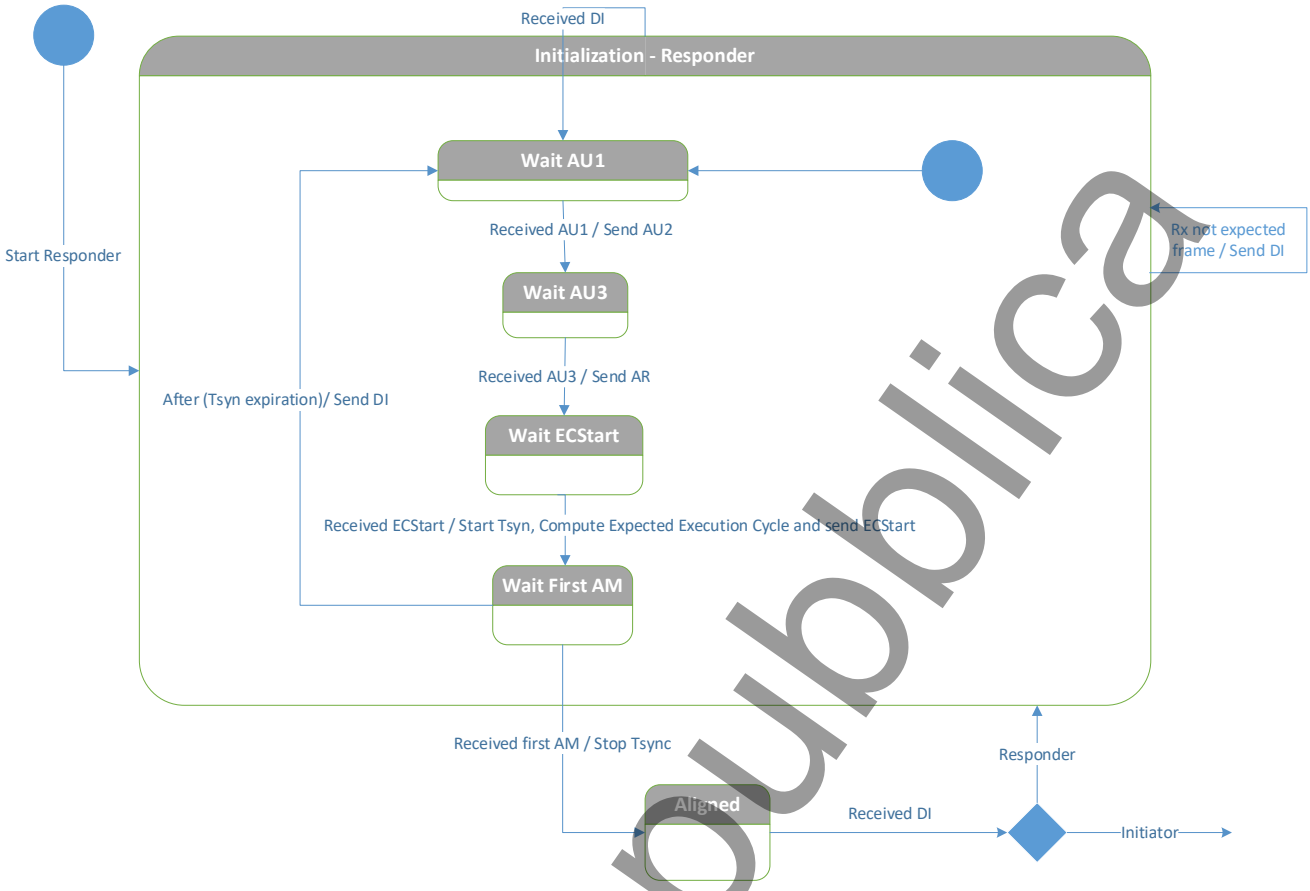


Figure 8-3 - Initialization State Machine – Responder

8.2.2 ALE Layer

[Req_ALE_020]

The Adaptation layer shall initialize a connection based on messages received from upper layers in accordance with Figure 8-1.

[Req_ALE_021]

The packet AU1 ALEPKT shall be sent only by the Initiator.

[Req_ALE_022]

When Responder receives AU1 ALEPKT, it shall verify that the content is equal to the expected one: if the check fails, the Responder shall discard the packet and send DI ALEPKT.

[Req_ALE_023]

The Responder shall be able to refuse or accept the request for connection establishment. If a connection is accepted then the AU2 ALEPKT is sent, otherwise a DI ALEPKT is sent.

[Req_ALE_024]

The subsystem shall carry out a further ALEPKT exchange to complete the related safe connetcion set-up:

Initiator to Responder: DT ALEPKT with User Data containing AU3 SaPDU conforming to the format specified in Table 3 – AU3 SaPDU structure.

Responder to Initiator: DT ALEPKT with User Data containing AR SaPDU conforming to the format specified in Table 5 – AR SaPDU structure.

From the point of view of the Adaptation Layer this last sequence is a normal exchange of data between two application users of the service so it is considered beyond the scope of the transport connection establishment specification.

8.2.3 SL Layer

[Req_SL_011]

The procedure for connection establishment shall be constituted by the following steps:

- (from initiator to responder) PDU AU_1: generated by Initiator;
  - field containing the Random number R_B of 8 bytes
    - unique constraint: R_B shall be different from "Pseudo-random Execution Cycle" (see §7.5.3) of Initiator
- (from responder to initiator) PDU AU_2:
  - the responder shall generate a Random number R_A of 8 bytes
    - unique constraint: R_A shall be different from "Pseudo-random Execution Cycle" (see §7.5.3) of responder;
  - R_A XOR R_B shall be inserted in the PDU AU_2
- (from initiator to responder) PDU AU_3:
  - initiator shall calculate R_A at the reception of AU_2
  - it shall generate a Random number R_C of 8 bytes
    - unique constraint: R_C shall be different from "Pseudo-random Execution Cycle" (see §7.5.3) of Initiator;
  - R_A XOR R_C shall be inserted in PDU AU_3
- (from responder to initiator) PDU AR:
  - Responder shall calculate R_C at the reception of AU_3
  - responder shall send R_B received in AU_1

After those steps, each node has acquired its identifier: the initiator has acquired R_A, while the responder has acquired R_C.

[Req_SL_012]

The maximum connection establishment delay timer T_{estab} is used for detecting unacceptable delay during the connection establishment. The timer is set after transmission of AU1 SaPDU and is stopped after reception of AR SaPDU.

In case of time-out of the timer T_{estab}:

- if waiting AU2: a disconnection request shall be trasmitted to ALE Layer;
- if waiting AR: a DI SaPDU is generated including a proper reason (see 8.3.4).

All SaPDUs shall be ignored if received after the timer elapses.

Note: the behaviour "*a disconnection request shall be trasmitted to ALE Layer* " is the same of Subset-037 Ref.2. WFTC state, time-out Testab "The DI SaPDU is not contained"; then in this case the DI ALEPKT send will have the field SaPDU empty.

[Req_SL_013]

In each DT packet (Data SAPDU), the initiator shall insert in XOR the value R_C on the Safety Code.

[Req_SL_014]

In each DT packet (Data SAPDU), the responder shall insert in XOR the value R_A on the Safety Code.

The proposed mechanism is based on a feedback message to avoid that undue storage leads to accept obsolete information for the reception of R_A and R_C.

There is no correctness check on the value of R_A and R_C because an erroneous value of one of these parameters leads to an error in the verification of received CRCs.

The AR message is not used for authentication purpose, but it used to detect malfunctioning or configuration errors of cryptographic layer (APL, see §7.3), before connection establishment.

[Req_SL_015]

When the responder transmits AR message, it shall inserts in the APL_CHECK_FIELD (see Table 5 – AR SaPDU structure) R_B previously extracted from AU1 PDU received from initiator. Initiator, receiving AR, shall verify that the content of this field is equal to the value of R_B transmitted with AU1.

In fact, as described in §8.2.5.1 and 8.2.5.2, if the APL layer is enabled this field has been:

- encrypted by initiator when transmitting AU1
- decrypted by responder when receiving AU1
- encrypted by responder when transmitting AR
- decrypted by initiator when receiving AR

If the verification made by the initiator on APL_CHECK_FIELD fails, it is possible to detect malfunctioning or configuration errors of cryptographic layer APL.

[Req_SL_016]

If initiator receives an AR message with an erroneous APL_CHECK_FIELD, it shall discard this message and start the actions for disconnection and reset procedure.

8.2.4 SAI Layer

The SAI layer is involved in the creation of a session only to initialize its EC expected counters (see §8.3.3.5).

Session connection is depicted in Figure 8-4.

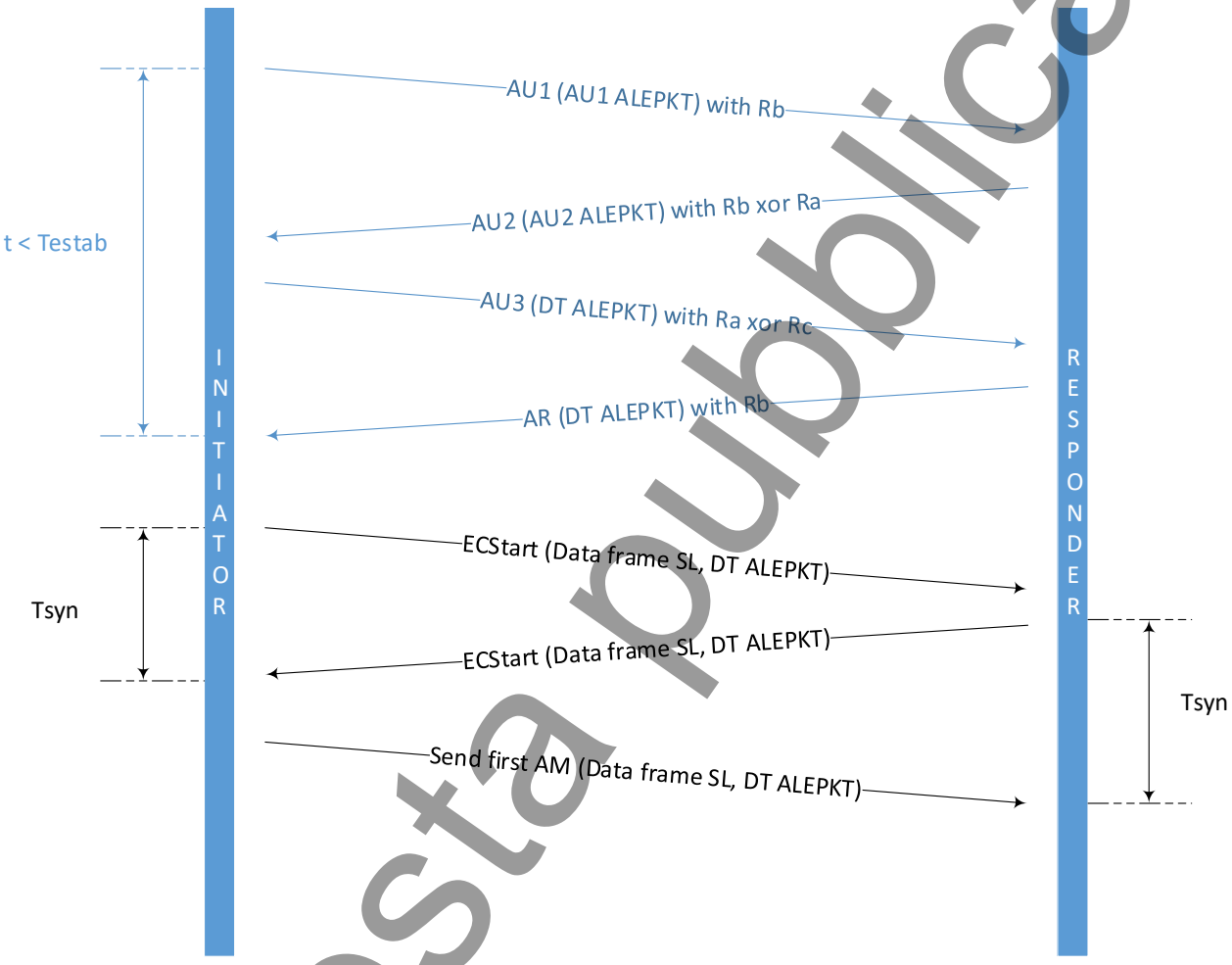


Figure 8-4 - Safe Connection establishment: Detailed protocol sequence - SAI

[Req_SAI_029]

Once the connection has been established between the two entities using the AU1, AU2, AU3 and AR SaPDU's, the initiator shall send an ECStart message with its initial EC counter and its EC period.

[Req_SAI_030]

The responder shall answer with an ECStart message containing its own information.

[Req_SAI_031]

After the reception of the responder's ECStart, the initiator shall send its first AM.

The responder shall answer with its first AM.

[Req_SAI_032]

A timer (Tsyn) shall be implemented in each peer entity in order to detect unacceptable initial delays:

In the SAI of the connection initiator entity: the timer shall be set at the sending of the ECStart message and shall be stopped at the reception of the ECStart message from the responder;

In the SAI of the connection responder entity: the timer shall be set at the sending of the ECStart message and shall be stopped at the reception of the first DT SaPDU by the counterpart SAI, containing the first AM.

If the timer expires before the reception of the expected message, the error shall be managed sending a DI SaPDU and releasing the connection.

[Req_SAI_033]

At the reception of an unexpected ECStart message, the receiver shall close the connection.

It means, in PRS option an ECSStart message different from ox91 and in MIS option an ECSStart message different from ox81.

[Req_SAI_034]

At reception of a correct ECStart both Initiator and Responder shall store the Sequence Number and the Execution Cycle received (integer and pseudo-random values).

To get the pseudo-random values operations equivalent the the following shall be done:

- Received (PR-SN$_y$ XOR nSaCEPID-REM$_{-y}$) XOR nSaCEPID-REM$_{-y}$
- Received (PR-EC$_y$ XOR nSaCEPID-REM$_{-y}$) XOR nSaCEPID-REM$_{-y}$

Note that in requirement Req_SAI_029 the XOR with nSaCEPID-REM$_{-y}$ has the consequence that the correct PR-EC and PR-SN are extracted only in case of message authenticity.

8.2.5 APL Layer

8.2.5.1 APL TX

The Access Protection Layer is between SL and ALE.

To define which messages and which fields have to be encrypted, the following requirements are given.

[Req_APL_004]

Cryptography shall be applied according to the following rules:

- First level of cryptography: to encrypt last 8 bytes of AU1, AU2, AU3, AR and DT SaPDU messages (Safey code field in case of DT messages), the AES algorithm shall be used Ref.4;
- Second level of cryptography: a Cipher-based Message Authentication Code (CMAC) shall be computed on the previous messages except on the last 8 bytes, the AES-CMAC algorithm shall be used (Ref.10).

The results of these two algorithms shall be combined in XOR.

In this way even if the data are transmitted in clear, it is impossible for a hacker to transmit authentic packets.

So each DTSaPDU message contains plain information and its encrypted representation (in addition to other implicit information but not secret such as nSaCEPID and the polynomial to calculate CRC); this could facilitate an attack to detect cryptographic keys. The protection to this menace is ensured by the only information that is hidden: the value R_C/R_A in XOR with the Safety Code.

[Req_APL_005]

The First level of cryptography shall be applied to the following fields (last 8 bytes):

- AU1 frame: the field containing the Rb random number (see Table 2 – AU1 SaPDU structure);
- AU2 frame: the field containing the XOR between the random numbers Ra and Rb (see Table 4 – AU2 SaPDU structure);
- AU3 frame: the field containing the XOR between the random numbers Ra and Rc (see Table 3 – AU3 SaPDU structure);
- AR frame: the field containing the Rb random number (see Table 5 – AR SaPDU structure);
- DT SaPDU frame: the field containing the Safety Code (see Table 7 – SaPDU_Header structure).

The field to be encrypted is 64-bit-long, but AES has a fixed block size of 128 bits, so a padding constituted by the duplication of these 8 bytes shall be added.

[Req_APL_006]

The Second level of cryptography shall be applied to the fields (full message except the last 8 bytes) indicated in Table 19.

SaPDU types	Fields to be used in the CMAC-AES computation	Fields excluded (last 8 bytes) from CMAC-AES computation
AU1 (Table 2 – AU1 SaPDU structure)	02 00 00 00 02	Rb
AU2 (Table 4 – AU2 SaPDU structure)	05 00 00 00 02 22 47 52 41 54 49 53 22	Ra XOR Rb
AU3 (Table 3 – AU3 SaPDU structure)	06	Ra XOR Rc
AR (Table 5 – AR SaPDU structure)	13	Rb
DT (Table 7 – SaPDU_Header structure)	0A/0B User data (SAI_telegram field)	SL_CRC

Table 19 – CMAC-AES vs SL frames

In Figure 8-5 the steps to compute, for DTSaPDU, the two levels of cryptography are depicted.

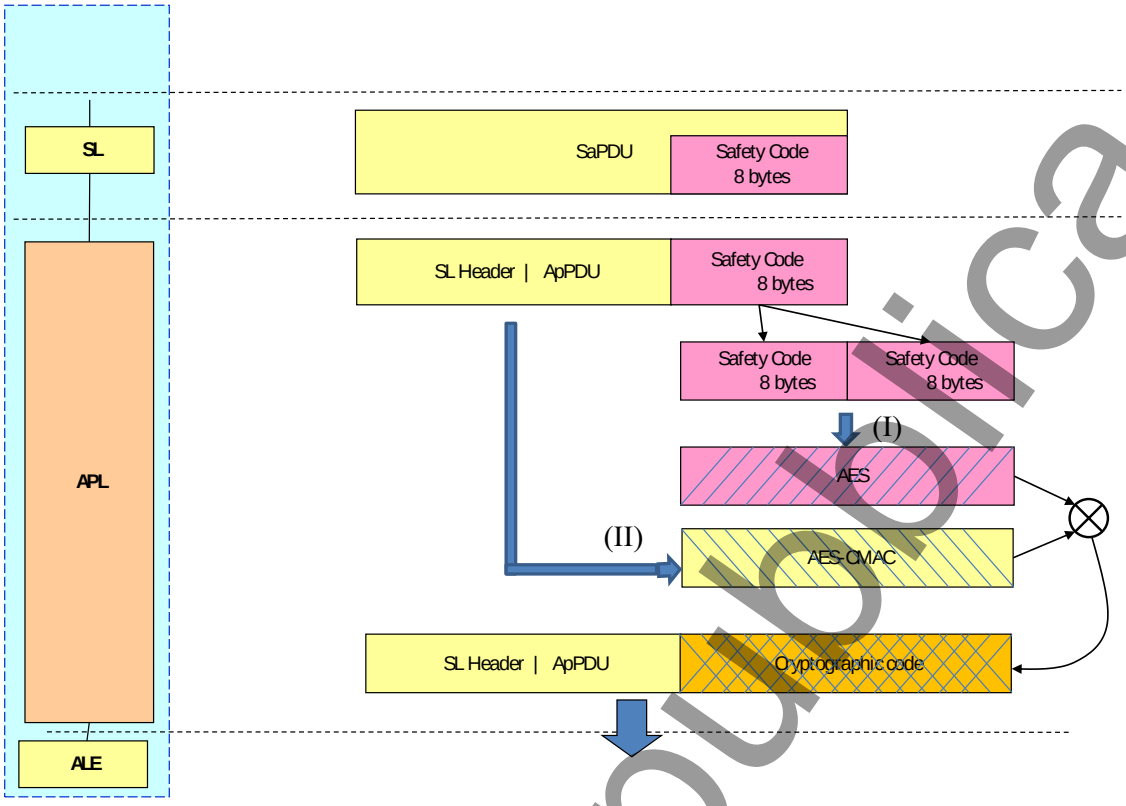


Figure 8-5 – APL Tx

8.2.5.2 APL RX

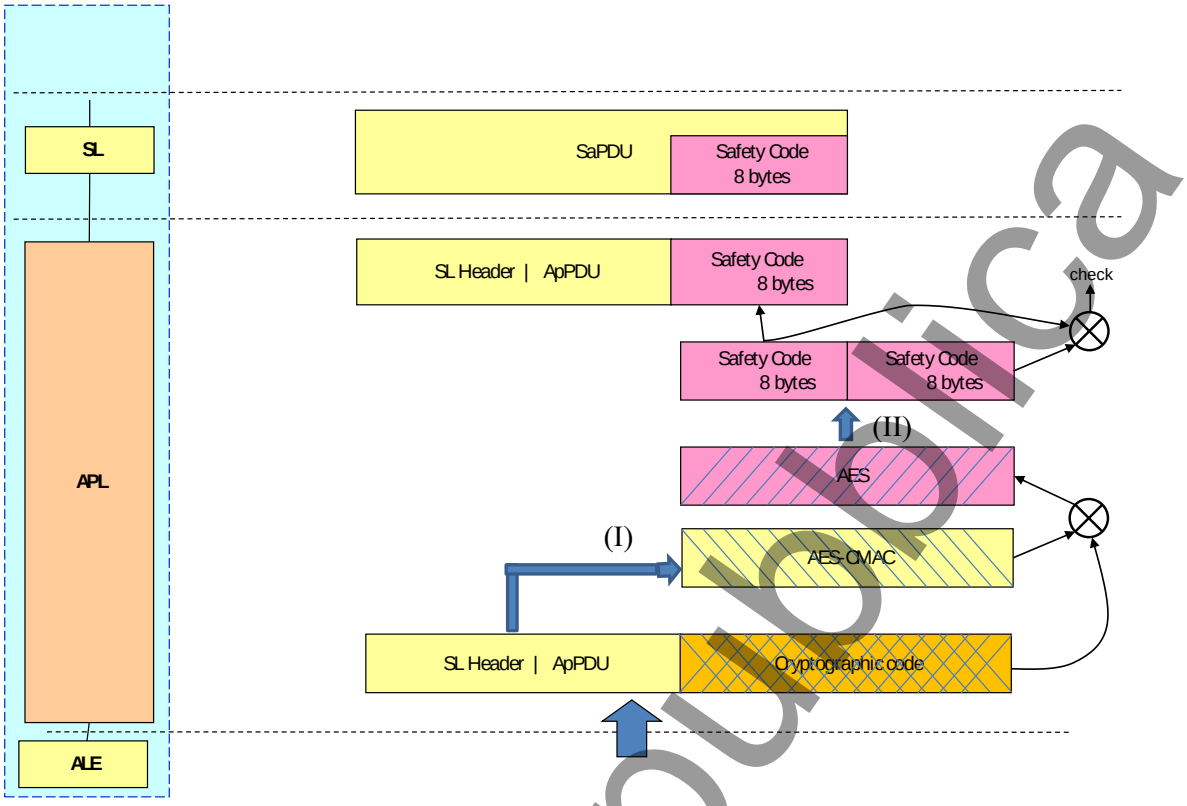
[Req_APL_007]

In case of reception, APL shall:

I) compute the AES-CMAC on the whole frame received from the ALE layer (AU1, AU2 and DT - ALEPKT) excluding last 16 bytes; the result shall be XOR'd with the last 16 bytes of the received frame (second level of cryptography);

II) decrypt (first level of cryptography) the result of previous point (16 bytes), to extract a couple of 8-byte-long values and perform the following check: if the two values are equal (in transmission the duplication of the 8 bytes has been encrypted to reach a length of 16 bytes, i.e. 128 bits is the block size of AES, see 8.2.5), the message containing only one of the two fields of 8 bytes can pass to SL.

The steps to compute, for received DTSaPDU, the two levels of cryptography are depicted in the Figure 8-6.



964

965 **Figure 8-6 - APL Rx**

966

967

968

969

8.3 Run time management

8.3.1 Run-time state machine

In Figure 8-7 an example of protocol State machine to use at run-time is given.

Run-time State Machine

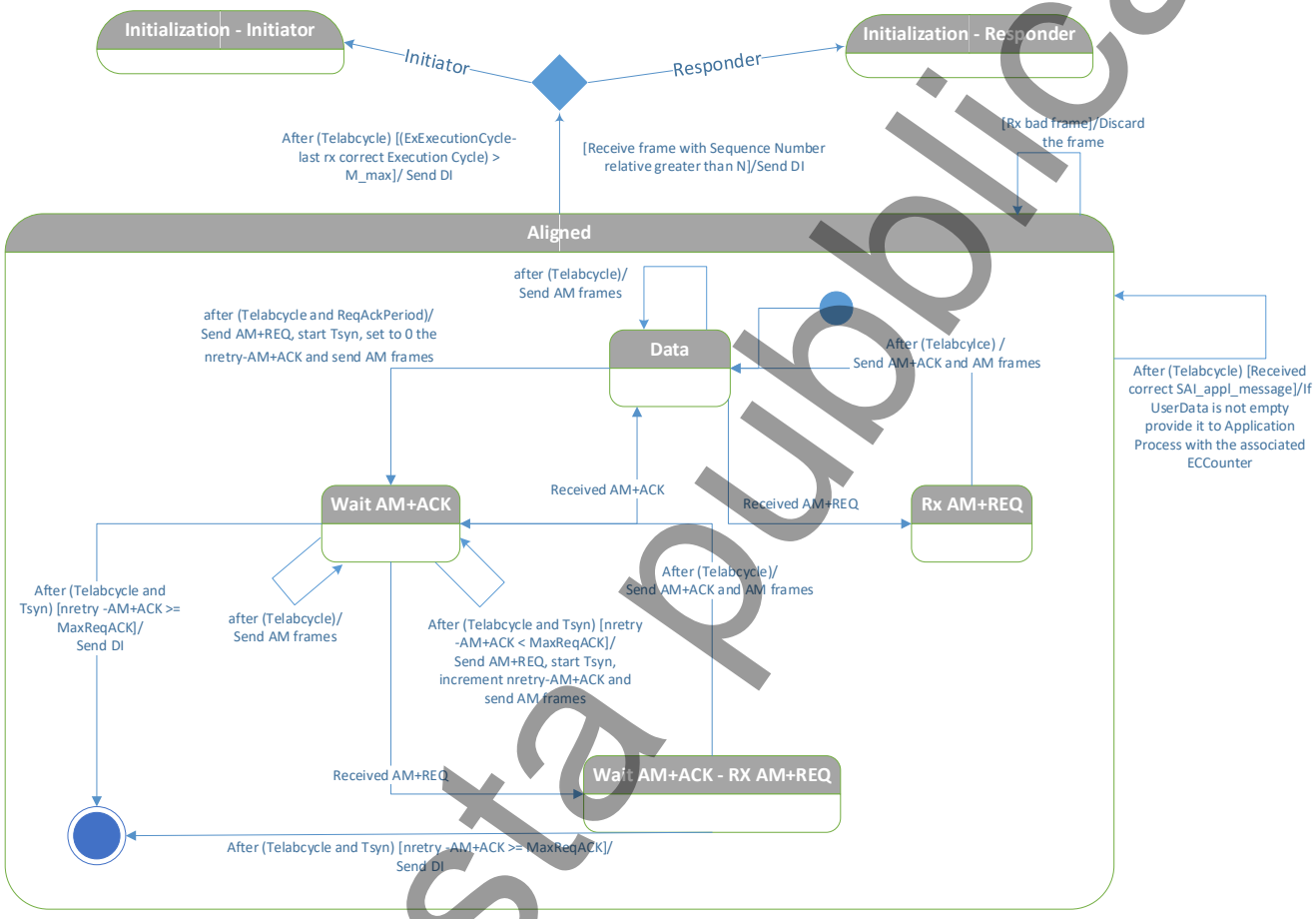


Figure 8-7 – Run-time State Machine

List of states:

When the Initiator (or the Responder) is aligned, the list of possible state is the following:

- Macro state: Aligned: node is aligned
  - Data: it sends AM frames
  - Rx_AM+REQ: AM+REQ has been received
  - Wait_AM+ACK-RX_AM+REQ: it waits for AM+ACK and AM+REQ has been received
  - Wait_AM+ACK: it waits for AM+ACK
  - Aligned: node is aligned

List of input events:

- Rx bad frame: invalid frame received

- After(Telabcycle), (ExExecutionCycle – last rx correct ExecutionCycle)>M_max: at the end of the Execution cycle period, the difference between the expected value and the last correct received value of Execution Cycle is greater than M_min
- After(Telabcycle): end of Execution cycle reached
- Received frame with Sequence Number relative greater than N: a frame with relative SN greater than N has been received
- After(Telabcycle), Received correct SAI_appl_messages: at the end of the Execution cycle period, a correct SAI ApPDU has been received
- After(Telabcycle and ReqAckPeriod): end of the Execution cycle and ReqAckPeriod reached
- Received AM+ACK: it receives AM+ACK
- Received AM+REQ: it receives AM+REQ
- After(Telabcycle and Tsyn) nretry-AM+ACK>=MaxReqACK: at the end of the Execution cycle with Tsyn expired, the number of retries is greater than MaxReqACK
- After(Telabcycle and Tsyn) nretry-AM+ACK<MaxReqACK: at the end of the Execution cycle with Tsyn expired, the number of retries is less than MaxReqACK

List of output events:

- Send DI: it sends DI SaPDU message
- Discard the frame: it discards the frame
- If UserData is not empty provide it to Application Process with the associated ECCounter: if user data is not empty, it must be transferred to application interface with the related EC counters
- Send AM frames: it sends AM frames
- Send AM+ACK and AM frames: it sends AM+ACK and AM frames
- Send AM+REQ, start Tsyn, set to 0 the nretry-AM+ACK and send AM frames: it sends AM+REQ, starts Tsyn, sets to 0 the number of retries and sends AM frames
- Send AM+REQ, start Tsyn, increment nretry-AM+ACK and send AM frames: it sends AM+REQ, starts Tsyn, increments the number of retries and sends AM frames

Table of transitions:

When the Initiator (or the Responder) is aligned the table of transitions is as follows:

Current State	Input/Event	Action/Output	Next State
Aligned - Data	After(Telabcycle)	Send AM frames	Aligned – Data
	Received AM+REQ	None	Aligned - Rx_AM+REQ
	After(Telabcycle and ReqAckPeriod)	Send AM+REQ, start Tsyn, set to 0 the nretry-AM+ACK and send AM frames	Aligned – Wait_AM+ACK
	Received frame with Sequence Number relative greater than N	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	After(Telabcycle), Received correct SAI_appl_messages	If UserData is not empty provide it to 2Appl interface with the associated ECCounter	Aligned – Data
	After(Telabcycle), (ExExecutionCycle – last rx	Send DI	If Initiator -> Initialization – Initiator

Current State	Input/Event	Action/Output	Next State
	correct ExecutionCycle)>M_min		If Responder -> initialization – Responder
	Rx bad frame	Discard the frame	Aligned - Data
Aligned - Rx_AM+REQ	After(Telabcycle)	Send AM+ACK and AM frames	Aligned – Data
	Received frame with Sequence Number relative greater than N	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	After(Telabcycle), Received correct SAI_appl_messages	If UserData is not empty provide it to 2Appl interface with the associated ECCounter	Aligned - Rx_AM+REQ
	After(Telabcycle), (ExExecutionCycle – last rx correct ExecutionCycle)>M_min	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	Rx bad frame	Discard the frame	Aligned - Rx_AM+REQ
Aligned - Wait_AM+ACK-RX_AM+REQ	After(Telabcycle)	Send AM+ACK and AM frames	Aligned - Wait_AM+ACK
	After(Telabcycle and Tsyn) nretry-AM+ACK>=MaxReqACK	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	Received frame with Sequence Number relative greater than N	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	After(Telabcycle), Received correct SAI_appl_messages	If UserData is not empty provide it to 2Appl interface with the associated ECCounter	Aligned - Wait_AM+ACK-RX_AM+REQ
	After(Telabcycle), (ExExecutionCycle – last rx correct ExecutionCycle)>M_min	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	Rx bad frame	Discard the frame	Aligned - Wait_AM+ACK-RX_AM+REQ
Aligned - Wait_AM+ACK	After(Telabcycle)	Send AM frames	Aligned - Wait_AM+ACK
	Received AM+REQ	None	Aligned - Wait_AM+ACK-RX_AM+REQ
	After(Telabcycle and Tsyn) nretry-AM+ACK<MaxReqACK	Send AM+REQ, start Tsyn, increment nretry-AM+ACK and send AM frames	Aligned - Wait_AM+ACK
	After(Telabcycle and Tsyn) nretry-AM+ACK>=MaxReqACK	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	Received AM+ACK	None	Aligned - Data
	Received frame with Sequence Number relative greater than N	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder

Current State	Input/Event	Action/Output	Next State
	After(Telabcycle), Received correct SAI_appl_messages	If UserData is not empty provide it to 2Appl interface with the associated ECCounter	Aligned - Wait_AM+ACK
	After(Telabcycle), (ExExecutionCycle – last rx correct ExecutionCycle)>M_min	Send DI	If Initiator -> Initialization – Initiator If Responder -> initialization – Responder
	Rx bad frame	Discard the frame	Aligned - Wait_AM+ACK

Table 20 – Finite State Machine – Run time management

8.3.2 SL layer

8.3.2.1 Safety code construction procedure in transmission

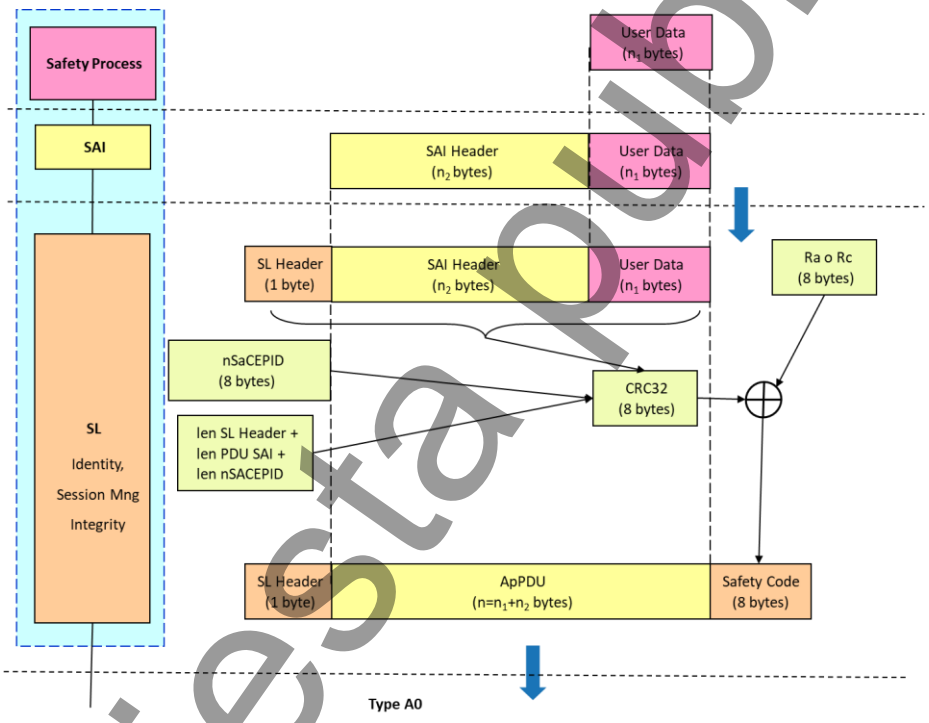


Figure 8-8- Safety code construction procedure in transmission

The SL layer manages the Safety code.

Safety code construction is depicted in Figure 8-8.

[Req_SL_017]

For the Safety code construction in transmission, an algorithm equivalent to the following steps shall be executed ("|" denotes concatenation):

1. Set direction flag of message m (value '0' for initiator, value '1' for responder).
2. Append the nSaCEPID-REM (8 bytes) in front of the message m: " nSaCEPID-REM | m".
3. Compute length of string " nSaCEPID-REM | m" in bytes and append length L (2 bytes) in front of the string for CRC computation, i.e. L | nSaCEPID-REM | m

4. If the length of the message (L | nSaCEPID-REM | m) in bits is not a multiple of 64 then perform padding with zeros and append padding data p: (L | nSaCEPID-REM | m | p)
5. Calculate the CRC on (L | nSaCEPID-REM | m | p), specifications are:
    - The p padding of the sequence on which the CRC is calculated is made with value 0;
    - The initialisation value for CRC calculation is 0;
    - The CRC (8 bytes) is composed of the couple (CRC1, CRC2)
        - byte 0..3 : CRC1
        - byte 4..7 : CRC2
6. XOR the result with Ra or Rc (to ensure the correct identification of the session) depending on the PDU generator: Responder or Initiator.
7. Store the result in Safety Code field.

8.3.2.2 Safety code verification procedure in reception

Safety code verification in reception is depicted in Figure 8-9.

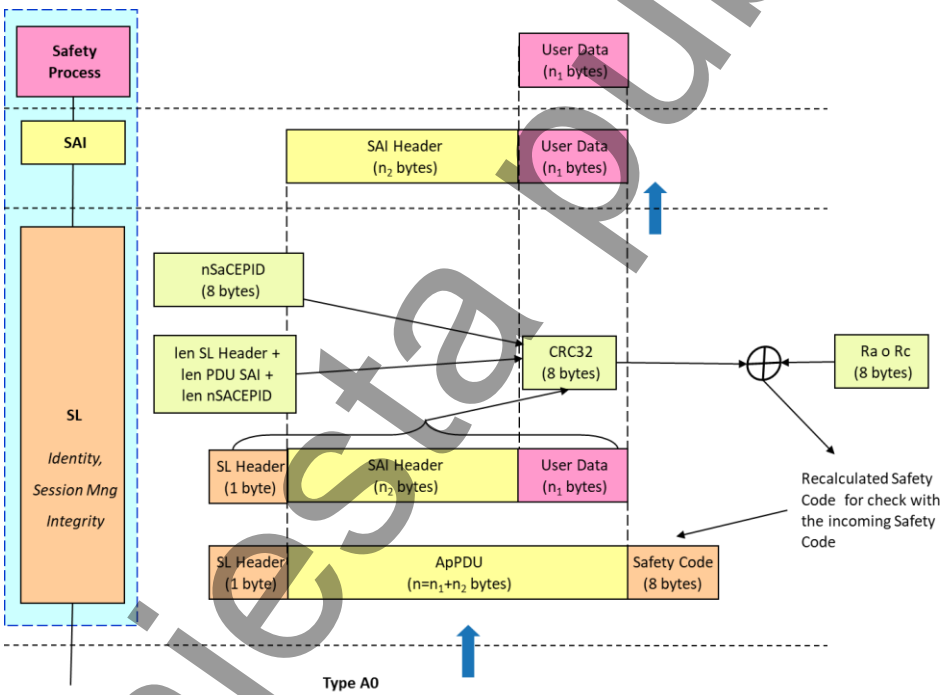


Figure 8-9 - Safety code verification procedure in reception

[Req_SL_018]

For the Safety code verification in reception, an algorithm equivalent to the following steps shall be executed ("|" denotes concatenation):

1. Append the nSaCEPID-LOC (8 bytes) in front of the message m: " nSaCEPID-LOC | m".
2. Compute length of string " nSaCEPID-LOC | m" in bytes and append length L (2 bytes) in front of the string for CRC computation, i.e. L | nSaCEPID-LOC | m
3. If the length of the message (L | nSaCEPID-LOC | m) in bits is not a multiple of 64 then perform padding with zeros and append padding data p: (L | nSaCEPID-LOC | m | p)
4. Calculate the CRC on (L | nSaCEPID-LOC | m | p) ; specifications are:

a. The p padding of the sequence on which the CRC is calculated is made with value 0;

b. The initialisation value for CRC calculation is 0;

c. The CRC (8 bytes) is composed of the couple (CRC1,CRC2)

i. byte 0..3 : CRC1

ii. byte 4..7 : CRC2

5. XOR the result with R_A or R_C (to ensure the correct identification of the session) depending on the sender: Responder or Initiator.

6. Compare the two values of CRC: the one computed and the one received. In case of wrong CRC, the message is discarded.

7. Verify the value of the direction flag

8.3.3 SAI layer

8.3.3.1 Procedure for detection of transmission delay

This procedure is described in Figure 8-10 (where one requesting entity only is shown).

Figure 8-10 - Procedure for detection of network delay

[Req_SAI_035]

The Procedure for detection of transmission delay shall be executed periodically, being the period a configurable parameter (ReqACKPeriod).

[Req_SAI_036]

The Procedure for detection of transmission delay shall begin by sending a AM+REQ message to the remote node.

[Req_SAI_037]

At the reception of the AM+REQ message, the Subsystem shall answer to the peer entity sending in the next cycle an AM+ACK message.

[Req_SAI_038]

In PRS option:

A timer (Tsyn) shall be implemented in the node that sends the AM+REQ (0x97) in order to provide a further detection of unacceptable transmission delays: the timer shall be set at the sending of the AM+REQ (0x97) and shall be stopped at the reception of the related AM+ACK (0x98) from the remote node.

If the timer expires before the reception of the expected message the error shall be managed re-sending an AM+REQ message.

If the number of consecutive timer expiration is greater than the parameter MaxReqACK the connection shall be released.

The AM+ACK relative to AM+REQ is those that contains in the fields EC Received and Echo PR EC and SN the values derived from the received AM+REQ (EC and PR EC and SN fields), see Figure 8-11.

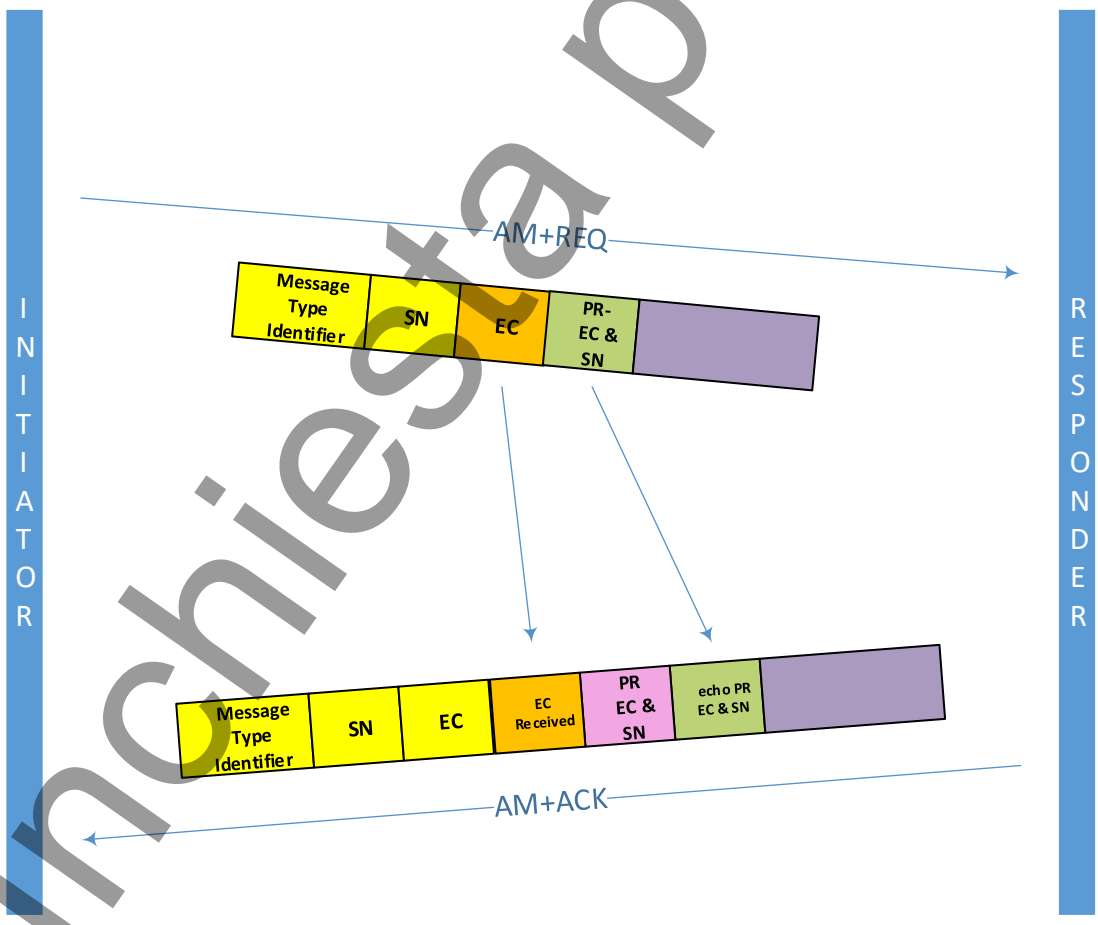


Figure 8-11 – Procedure for detection of network delay – PRS

[Req_SAI_039]

In MIS option:

A timer (Tsyn) shall be implemented in the node that sends the AM+REQ (0x87) in order to provide a further detection of unacceptable transmission delays: the timer shall be set at the sending of the AM+REQ (0x87) and shall be stopped at the reception of the first AM+ACK (0x88) from the remote node.

If the timer expires before the reception of the expected message the connection shall be released.

8.3.3.2 Transmission algorithm

The transmission is in charge to send the upper layers data and to keep alive the connection with the remote node.

[Req_SAI_040]

At each Execution cycle (Telabcycle expiration) the first frame to send shall be one of the possible frames:

- AM+ACK in response to a received AM+REQ;
- AM+REQ if requested by the procedure for detecting a transmission delay;
- AM otherwise

The UserData field shall contain the first packet received, during last Execution cycle period, from the Application layer if available; otherwise it shall be empty (NOP).

If more than one packets are received from the Application layer, they shall be sent each one in a separated AM frame.

The frames with UserData field empty are also called NOP frames.

8.3.3.3 Reception algorithms

The reception algorithms are in charge to verify the correctness (in terms of sequence and freshness) of the received frames.

In the case of an error being detected, the SAI is not able to take any action to recover the missing data, then it will consider the frame as not received and in some cases the connection has to be released.

[Req_SAI_041]

A received frame (AM, AM+ACK or AM+REQ) shall be provided to the Application layer only if:

- all the verifications related to Sequence and Execution cycle numbers (integer and pseudo-random) are positive;
- the UserData is not empty.

The procedure to use for verification of Sequence and Freshness are detailed in the following sections.

8.3.3.4 Verification of Sequence Number (integer value)

A parameter N is defined for the message sequence check.

N-1 is the number of missing messages allowed, with N≥1 parameter to be configured.

[Req_SAI_042]

The receiver shall accept any sequence number (integer value) in the first sequence numbered message received from the remote transmitter.

[Req_SAI_043]

If the received SN (integer value) is less than or equal to the SN of the last accepted message, the message shall be discarded without releasing the safe connection.

[Req_SAI_044]

If the received SN (integer value) is greater than the SN of the last accepted message + N, the message shall be discarded and optionally the safe connection shall be released.

[Req_SAI_045]

If the received SN is greater or equal to the SN of the last accepted message + 1 and less than or equal to the SN of the last accepted message + N, the message is not discarded.

8.3.3.5 Verification of Execution Cycle (integer value)

The EC defence technique is used to protect the application data message against the delay threat, as required by EN 50159), and to ensure the time validity of the application data.

The protection against the delay threat is achieved by using the EC counter contained in the Header of each message received from the peer entity. The EC must have a fixed period.

The detection of the delay threat is achieved by making a "comparison" between the EC counter contained in the message received from the peer entity and the expected EC counter.

This defence technique consists in guaranteeing in transmission the availability of a message (without application data, if not required by the application) within a certain time period starting from last transmitted message, therefore turning the transmission mode into pseudo-cyclic. So at the other side of the safe connection, it is possible to manage a timeout on the data receiving.

The first step of the EC defence technique consists in defining the expected EC Counter. For the computation of the expected EC value, the ratio R between the local and the remote EC period is computed by each SAI entity at the end of the initialisation of the EC defence technique using the following formula:

R = receiver EC period / sender EC period. Note: R is a real number.

Once the initialisation procedure of the EC defence technique has been successfully achieved, each entity checks if the application data transmitted by the peer entity is obsolete or not.

This defence technique is based on the evaluation of the value of the EC Counter contained in the messages received by the remote subsystem, which is compared with the expected one (Ex).

[Req_SAI_046]

When the initiator receives the EC Start message from the responder, the initiator shall initialise the Ex using the EC counter value contained in that message.

Equivalent activities shall be performed for the pseudo-random values (Ex Cod) (PRS option).

[Req_SAI_047]

When the responder receives the EC Start message from the initiator, the responder shall initialise the Ex using the EC counter value contained in that message received from the initiator. Equivalent activities shall be performed for the pseudo-random values (Ex Cod) (PRS option).

[Req_SAI_048]

At each Execution Cycle (EC), the Expected Execution Cycle (Ex) shall be updated with an algorithm equivalent to:

Next Ex=Ex+R

where $R = EC_{receiver\_period} / EC_{sender\_period}$ (R is a real number)

[Req_SAI_049]

At each Execution cycle of the receiver, the variable M is computed as follows:

1) M = Lower integer value(Expected Execution Cycle - EC contained in the last accepted message in this Execution cycle).
2) If no message arrives to the receiver in an Execution Cycle, the receiver shall compute M using the EC counters of the last accepted message.

[Req_SAI_050]

EC counter check procedure.

Let M be Lower integer value(Expected Execution Cycle- EC contained in the last received message).

Given M_min and M_max, configurable parameters indicating respectively the minimum and maximum admissible values for M, the following cases shall be considered:

1) [M < M_min < 0] two options are possible:
   a) the message shall be accepted, the "Procedure for detection of transmission delay" (§8.3.3.1) shall be executed, the following assignment shall be done: Next Ex=EC received +R (overwriting the previous value)
   b) the message shall be discarded and the the connection shall be released
2) [M_min ≤ M < 0] the message shall be accepted
3) [0 ≤ M ≤ M_max] the message shall be accepted
4) [0 < M_max < M] the message shall be discarded and the the connection shall be released

[Req_SAI_051]

The connection status shall be made available to the Application Process.

8.3.3.6 Verification of pseudo-random Sequence Number and Execution Cycle

This paragraph is applicable only to the PRS option.

The verification procedure is depicted in Figure 8-12.

The algorithms here described make use of the following pseudo-random variables:

- PR-Ex$_y$
- PR-EC-REM-RX$_y$
- PR-EC-CHECK$_y$
- PR-SN-REM$_y$
- PR-SN-CHECK$_y$

[Req_SAI_052]

For what concerns pseudo-random values, when Ex is incremented, the PR-Expected Execution Cycle$_y$ (PR-Ex) values shall be incremented with an algorithm equivalent to the following steps:

1. Incr = Lower integer value (Next Ex) - Lower integer value (Current Cycle Ex).
2. Next PR-Ex$_y$ = LFSRy(PR-Ex$_y$) repeated Incr times.

Examples related to Ex management versus Incr value are given in §Annex C.

[Req_SAI_053]

The pseudo-random values PR-EC and PR-SN shall be checked only if the checks on integer values are succesful.

[Req_SAI_054]

Before accepting the frame, PR-SN and PR-EC of the received frame shall be verified.

The algorithm to follow, implementation dependent, shall be equivalent to the following one:

Let D (0<D≤N, see §8.3.3.4) be the difference between the integer SN of the received message and that of the last accepted message

Compute:

- PR-SN-CHECK$_y$ = LFSR$_y$ (PR-SN$_y$ of last accepted message) repeated D times
- PR-EC-REM-RX$_y$= received [PR-SN-REM$_y$ XOR PR-EC-REM$_y$ XOR nSaCEPID-REM$_y$]
  XOR nSaCEPID-REM$_y$
  XOR PR-SN-CHECK$_y$

Let M be Lower integer value(Expected Execution cycle - Received Execution Cycle), the following cases shall be considered:

- [M<0] compute PR-EC-CHECK$_y$ =LFSR$_y$(PR-Ex$_y$) repeated |M| times
  The result PR-EC-CHECK$_y$ = PR-EC-REM-RX$_y$ leads to the successful verification.
- [0≤M≤M_max] compute PR-EC-CHECK$_y$ =LFSR$_y$(PR-EC-REM-RX$_y$) repeated M times.
  The result PR-EC-CHECK$_y$ = PR-Ex$_{-y}$ leads to the successful verification.

Note that in the PR-EC and PR-SN verification (Req_SAI_052) the XOR with nSaCEPID-REM$_y$ has the consequence that a successful verification is a verification also of message authenticity.

[Req_SAI_055]

If all tests related to sequence and freshness are successful (see §8.3.3.4, §8.3.3.5 and §8.3.3.6), then the received packet is correct: received SN and EC shall be updated both in integer and pseudo-random formats.

Otherwise the packet shall be discarded.

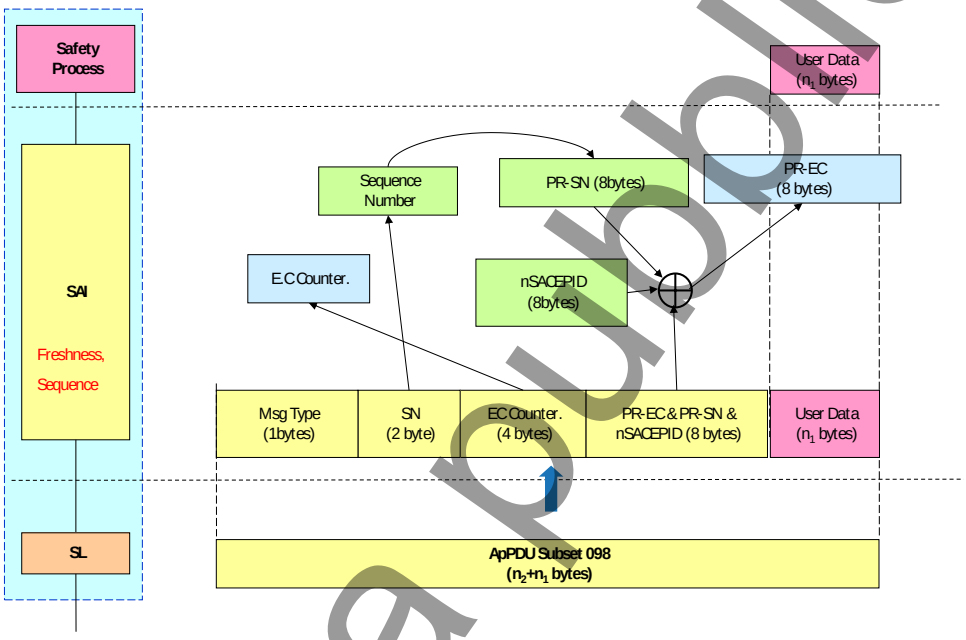


Figure 8-12 – Sequence and Freshness tests (PRS option)

8.3.3.7 Reception of AM/AM+REQ/ACM+ACK – PRS option

At the reception of AM the sequence and the freshness have to be verified as described in §8.3.3.4, §8.3.3.5 and §8.3.3.6.

[Req_SAI_056]

At Reception of AM+REQ with PRS option: in addition to the verifications performed for AM frame (§8.3.3.4, §8.3.3.5 and §8.3.3.6):

If the received packet is correct the following values shall be stored, to be used for the transmission of AM+ACK:

- Received (PR-SN$_y$ XOR PR-EC$_y$ XOR nSaCEPID-REM$_y$)
- Received Execution Cycle

[Req_SAI_057]

At Reception of AM+ACK with PRS option: it shall be verified that the field "Execution Cycle Number Received" extracted from received AM+ACK (0x98) is equal to the EC value transmitted in the last sent message AM+REQ (0x97):

- if not, the message shall be discarded.
- If yes, the message shall be managed in accordance with requirements Req_SAI_058, Req_SAI_059 and Req_SAI_060.

NOTE In this case the received AM+ACK is discharged because it is not that related to the sent AM+REQ being the Execution Cycle Number Received not equal.

[Req_SAI_058]

At Reception of AM+ACK with PRS option: the field "Echo of Received pseudo-random Execution Cycle and Sequence Number" of the received PDU shall be put in XOR with:

- The value ($PR\text{-}SN_y$ XOR $PR\text{-}EC_y$ XOR $nSaCEPID\text{-}LOC_y$) written in the frame Request of ACK (0x97) previously sent;
- $nSaCEPID\text{-}REM_y$

in order to obtain the value of $PR\text{-}EC_y$ of the remote node.

[Req_SAI_059]

At Reception of AM+ACK with PRS option: both the representation (integer and pseudo-random) of Ex shall be updated by using the values EC and PR-EC extracted from the received AM+ACK (0x98) message.

[Req_SAI_060]

At Reception of AM+ACK with PRS option: after updating Ex, all the verification executed for AM (0x96) (§8.3.3.7) shall be performed.

8.3.3.8 Reception of AM/AM+REQ/ACM+ACK – MIS option

At the reception of AM/AM+REQ/AM+ACK the sequence and the freshness have to be verified as described in §8.3.3.4 and §8.3.3.5.

[Req_SAI_061]

At Reception of AM+ACK with MIS option: the Ex shall be updated by using the EC extracted from the received AM+ACK (0x88) message.

8.3.4 Error handling procedure

8.3.4.1 General

If an error occurs in the SAI_SL the error management has to indicate, in the SaPDU Disconnect (DI SaPDU) §7.4.3.9 , the reason and sub-reason of this error. One indicated reason may be caused by different sub-reasons which may be detected by symptoms requiring different error handling actions.

[Req_ERR_001]

The tuples (reason code, sub-reason code) shall be used to indicate in the DI SaPDU frame (fields ReasonField and SUB-ReasonField) the type of the error to the user of the service.

8.3.4.2 Sub-reasons for the reason 'Application layer request'

[Req_ERR_002]

For 'Application layer request', Reason Code and Subreason code shall be as defined in Table 21.

Reason Code	Sub-reason Code	Description
0	-	Disconnection requested by the Application Layer

Table 21 - Sub-reasons for the reason 'Application layer request'

8.3.4.3 Sub-reasons for the reason 'Invalid APL_CHECK_FIELD'

[Req_ERR_003]

For 'Invalid APL_CHECK_FIELD', Reason Code and Subreason code shall be as defined in Table 22.

Reason Code	Sub-reason Code	Description
4	4	APL_CHECK_FIELD error in AR SaPDU: failure or configuration errors of APL layer

Table 22 - Sub-reasons for the reason 'Invalid APL_CHECK_FIELD '

8.3.4.4 Sub-reasons for the error type 'failure in sequence integrity'

[Req_ERR_004]

For 'failure in sequence integrity ', Reason Code and Subreason code shall be as defined in Table 23.

Reason Code	Sub-reason Code	Description
5	1	Replay of authentication message (AU1 SaPDU, AU2 SaPDU, AU3 SaPDU, AR SaPDU) after connection establishment. Error code is used, if the error is not covered by reason code 9.

Table 23 - Sub-reasons for the error type 'failure in sequence integrity'

8.3.4.5 Sub-reasons for the reason 'Failure in the direction flag'

[Req_ERR_005]

'Failure in the direction flag' - if the flag is not correct, there shall be a SA-DISCONNECT.indication.

[Req_ERR_006]

For 'Failure in the direction flag', Reason Code and Subreason code shall be as defined in Table 24.

Reason Code	Sub-reason Code	Description
6	1	Value of direction flag '0' instead of '1'
6	2	Value of direction flag '1' instead of '0'

Table 24 - Sub-reasons for the reason 'Failure in the direction flag'

8.3.4.6 Sub-reasons for the reason 'Time out at connection establishment'

[Req_ERR_007]

For 'Time out at connection establishment', Reason Code and Subreason code shall be as defined in Table 25.

Reason Code	Sub-reason Code	Description
7	3	Time out of Testab without receiving the AR SaPDU

Table 25 - Sub-reasons for the reason 'Time out at connection establishment'

8.3.4.7 Sub-reasons for the reason 'Invalid SaPDU field'

[Req_ERR_008]

For 'Invalid SaPDU field', Reason Code and Subreason code shall be as defined in Table 26.

Reason Code	Sub-reason Code	Description
8	1	Invalid information field
8	5	Invalid AU1 SaPDU : the header indicates a AU1 SaPDU, but the rest of the Sa PDU does not match with the structure of an AU1 SaPDU.

Table 26 - Sub-reasons for the reason 'Invalid SaPDU field'

1.1.1.1 Sub-reasons for the reason 'Failure in sequence of the SaPDUs during connection set-up'

[Req_ERR_009]

For 'Failure in sequence of the SaPDUs during connection set-up', Reason Code and Subreason code shall be as defined in Table 27.

Reason Code	Sub-reason Code	Description
9	1	Transmission of AU1 SaPDU but a message different from AU2 SaPDU is obtained.
9	2	Transmission of AU2 SaPDU but a message different from AU3 SaPDU is obtained.
9	3	Transmission of AU3 SaPDU but a message different from AR SAPDU is obtained.

Table 27 - Sub-reasons for the reason 'Failure in sequence of the SaPDUs during connection set-up'

8.3.4.8 Sub-reasons for the reason 'SaPDU length error'

[Req_ERR_010]

For 'SaPDU length error', Reason Code and Subreason code shall be as defined in Table 28.

Reason Code	Sub-reason Code	Description
10	1	AU1 SaPDU length error
10	2	AU2 SaPDU length error
10	3	AU3 SaPDU length error
10	5	DT SaPDU length error
10	8	AR SaPDU length error

Table 28 - Sub-reasons for the reason 'SaPDU length error'

8.3.4.9 Other errors

[Req_ERR_011]

The code 127 (unknown) for Reason Code shall be used, when

- no proper reason code or subreason code can be selected
- the reason code or subreason code is undefined

[Req_ERR_012]

The reason codes 11-126 shall reserved for future use. The reason codes 128 and 129 shall be reserved for PVS implementation specific use as described in Table 29. For these reason codes the subreason codes (0…126, 128…255) shall also reserved for implementation specific use.

Reason Code	Sub-reason Code	Description
128	1	Reception of an unexpected SAI ECStart message (0x81) when PRS option configured.
128	2	Reception of an unexpected SAI ECStart message (0x91) when MIS option configured.
128	3	Version field of ECStart different from the expected one
128	4	Timeout Tsyn expired during "Initialization procedure of the EC defence technique" phase (see §8.3.3.5)
128	5	Timeout Tsyn expired during "Procedure for detection of transmission delay" (see §8.3.3.1). The disconnection occurs after the periodical retry of realignment has failed a configurable number of times. (If the timer expires before the reception of the expected message, the safe connection is released after a configurable number of successive errors (Tsucc_er))
129	1	Freshness algorithm error (see §8.3.3.5). Error detected in the field containing integer numbers
129	2	Freshness algorithm error (see §8.3.3.5). Error detected in the field containing pseudo-random numbers
129	3	SAI message received with Sequence Number over the threshold (The message is discarded and the safe connection is released if the received SN is greater than the SN of the last accepted message + N)

Table 29 - Sub-reasons for the 128 and 129 reasons

9 Configurable parameters list

[Req_CONF_001]

Table 30 contains a list of all protocol parameters (Parameter and Value columns) that shall be defined and justified in the system configuration phase and shall be parameters for PVS implementation.

In a computer, for each defined instance of PVS protocol a list of these parameters shall be available.

Note: if 2 or more PVS instances are defined within the same vital computer, an equivalent number of the following parameters list will be defined.

Parameter	Description	Refer to	Value
nSaCEPID	nSaCEPID of the PVS flow Where: • nSaCEPIDCh1: nSaCEPID Channel 1 • nSaCEPIDCh2: nSaCEPID Channel 2	See §7.4.2	Couple of 32-bit-long values (one for each channel) in hexadecimal format.
Remote nSaCEPID	nSaCEPIDCh1 Hexadecimal characters for the nSaCEPID 1 value of the remote unit nSaCEPIDCh2 Hexadecimal characters for the nSaCEPID 2 of the remote unit		
Telabcycle	It indicates the cyclical transmission time (expressed in milliseconds)	See §8.3.3.2	Cycle time expressed in milliseconds.
ReqACKPeriod	ReqACKPeriod is the period used for sending AM+REQ messages.	See §8.3.3.1	Number of local cycles
M_max	M_max defines the maximum acceptable value of the difference between the expected Execution Cycle (EC) and the received one.	See § 8.3.3, 8.3.3.5.	It is a positive value, with minimum value 0
M_min	M_min defines the minimum acceptable value of the difference between the expected Execution Cycle (EC) and the received one.	See § 8.3.3, 8.3.3.5.	It is a negative value.
Testab	Testab defines the timeout for Connection Establishment procedure.	See §8.3.4.6	It is a positive value expressed in ms. Minimum value 1.
Tsyn	Tsyn defines the timeout used during the "Procedure for detection of transmission delay" (AM+REQ / AM+ACK).	See §8.3.3.1	It is a positive value expressed in ms. Minimum value 1.
N	N-1 defines the maximum tolerated number of missing messages in the sequence.	See § 8.3.3.7	It is a positive value. Minimum value 1.
MaxReqACK	MaxReqACK defines the maximum number of retries used during the "Procedure for detection of transmission delay" (AM+REQ / AM + ACK).		It is a positive value. Minimum value 0.
Initiator	Initiator defines the role of the instance	See §5.2.	Where the value can be • "0" for Responder • "1" for Initiator
CryptKey	Key to use for the cryptography of APL layer (AES first level of cryptography).	See §7.3.	
CryptKeyE	Key to use for the cryptography of APL layer (AES-CMAC - second level of cryptography)	See §7.3.	

Table 30 - Configurable parameters list

10 Summary of PVS messages

As a recap, Figure 10-1 and Figure 10-2 depict all the fields part of PVS protocol frames.

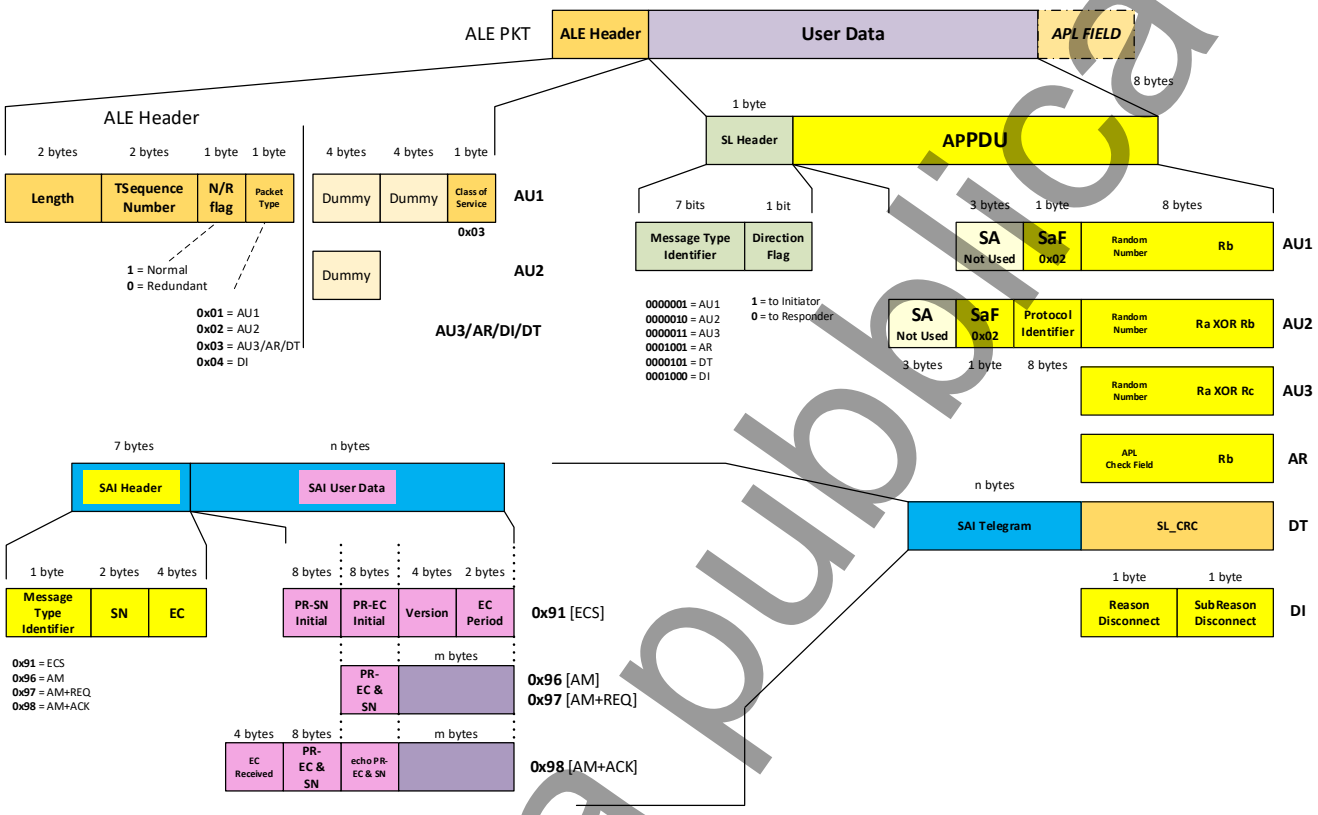


Figure 10-1 – summary of PVS messages PRS-OPTION

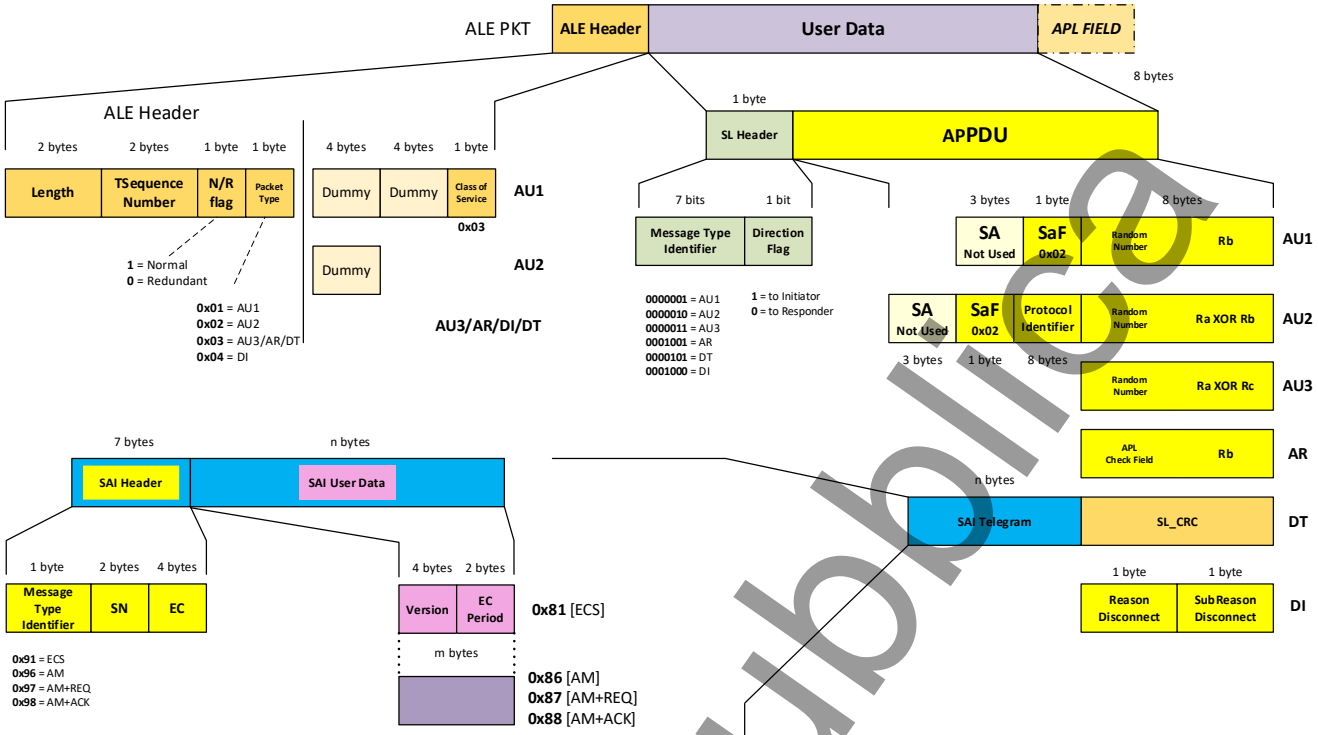


Figure 10-2 - summary of PVS messages MIS-OPTION

Annex A
(normative)

Application conditions

[Req_AC_001]

For each PVS instance the peer entities shall be configured coherently with the Initiator/Responder configuration.

[PVS_AC_002]

In case UDP protocol is used, unicast or multicast communication shall be selected at configuration level with the same value used on both peers, to reduce the combinations.

[Req_AC_003]

A procedure to handle the cripto keys used by the APL shall be defined in order to keep security protection.

[Req_AC_004]

nSaCEPIDs shall fulfill the following requirements:

- The size of nSaCEPID shall be 8 bytes (4+4 in analogy with CRC).
- The Hamming distance among all the high parts, as well as among all the low parts constituting nSaCEPID (e.g. high part) shall be at least 6 and maximum 26.
- For each instance (connection) of a subsystem, a nSaCEPID-LOC shall be assigned univocally and this values shall be known only to the instance itself and to the remote instance to communicate.

[Req_AC_005]

The peer entities shall be configured coherently for the expected maximum size of APPDU field of Data frames SaPDU (DT SaPDU) exchanged.

Each peer shall have the possibility to be configured at least with a maximum size of APPDU field of 1023 bytes.

[Req_AC_006]

In case of MIS option, the sequence 87-88 must be respected without repetitions (87...87 – 88) and within the foreseen times (Tsyn). In the event of a time-out after request 87, the requesting node must restart the connection, without forwarding a new request 87.

In other words, with the MIS option the parameter MaxReqACK shall be set to 0.

Annex B
(informative)

Examples of protocol analysis

B.1 APL disabled

B.1.1 Configuration parameters

	Initiator	Responder
nSaCEPID	0xA0A0A0A0A0A0A0A0	0x281C21046A5B0106

B.1.2 Initiator transmits AU1

00 1a 00 00 01 01 00 00 00 00 00 00 00 00 03 02 00 00 00 02 6b 64 1e 14 8b 21 fb 89

ALE Header is composed of the following fields and is highlighted in green:

- 00 1a: Packet Length
- 00 00: Sequence Number
- 01: N/R flag
- 01: Packet Type (Connect Request)
- 00 00 00 00: Ale Connect Header – (not used)
- 00 00 00 00: Ale Connect Header – (not used)
- 03: Ale Connect Header – Class of Service

AU1 SaPDU is composed of the following fields and is highlighted in yellow:

- 02: SL specific header for AU1
- 00 00 00: fields not used
- 02: Requested Safety Feature
- 6b 64 1e 14 8b 21 fb 89: Random Number R_B

B.1.3 Responder transmits AU2

00 1d 00 00 01 02 cc cc cc cc 05 00 00 00 02 22 47 52 41 54 49 53 22 e9 c3 0f e5 d8 8e 4b 82

ALE Header is composed of the following fields and is highlighted in green:

- 00 1d: Packet Length
- 00 00 : Sequence Number
- 01: N/R flag

- 02: Packet Type (Connect Confirmation)
- cc cc cc cc: (not used)

AU2 SaPDU is composed of the following fields and is highlighted in yellow:

- 05: SL specific header for AU2
- 00 00 00: fields not used
- 02: Requested Safety Feature
- 22 47 52 41 54 49 53 22 : fixed value, protocol identifier
- e9 c3 0f e5 d8 8e 4b 82: $R_A \oplus R_B$

B.1.4 Initiator transmits AU3

00 0d 00 01 01 03 06 2c 8e 17 ed be 86 00 94

ALE Header is composed of the following fields and is highlighted in green:

- 00 0d: Packet Length
- 00 01 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)
-

AU3 SaPDU is composed of the following fields and is highlighted in yellow:

- 06: SL specific header for AU3
- 2c 8e 17 ed be 86 00 94: $R_A \oplus R_C$

B.1.5 Responder transmits AR

00 0d 00 01 01 03 13 6b 64 1e 14 8b 21 fb 89

ALE Header is composed of the following fields and is highlighted in green:

- 00 0d: Packet Length
- 00 01 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

AR SaPDU is composed of the following fields and is highlighted in yellow:

- 13: SL specific for AR
- 6b 64 1e 14 8b 21 fb 89: R_B

B.1.6 Initiator transmits Execution Cycle Start (ECS)

00 2a 00 02 01 03 0a 91 00 01 00 00 00 10 d5 ed c7 12 51 ad 4b 07 31 8e 6a 92 ff d0 d5 db 00 00 00 02 02 58 78 c5 b7 65 60 b0 ef bf

ALE Header is composed of the following fields and is highlighted in green:

- 00 2a: Packet Length
- 00 02: Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

ECS is composed of the following fields and is highlighted in yellow:

- 0a: SL Header (DTSaPDU transmitted by Initiator)
- 91: SAI Message Type (0x91 for ECS pseudo-random sequence)
- 00 01: Sequence Number integer
- 00 00 00 10: Execution Cycle integer
- d5 ed c7 12 51 ad 4b 07: $SN_{pr} \oplus$ nSaCEPID
- 31 8e 6a 92 ff d0 d5 db: $EC_{pr} \oplus$ nSaCEPID
- 00 00 00 02: Version
- 02 58: EC period (600 ms)
- 78 c5 b7 65 60 b0 ef bf: Safety Code

B.1.7 Responder transmits Execution Cycle Start (ECS)

00 2a 00 02 01 03 0b 91 00 00 00 00 02 99 88 bc 81 a4 ca fb a1 a6 f4 23 13 7b 51 b8 5e 5a 00 00 00 02 01 f4 33 7c 46 ce 51 9b f7 51

ALE Header is composed of the following fields and is highlighted in green:

- 00 2a: Packet Length
- 00 02 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

ECS is composed of the following fields and is highlighted in yellow:

- 0b: SL Header (DTSaPDU sent from Responder, it is different from those of Initiator for Direction flag bit)
- 91: SAI Message Type (0x91 - ECS pseudo-random sequence)
- 00 00: Sequence Number intero
- 00 00 02 99: Execution Cycle intero
- 88 bc 81 a4 ca fb a1 a6: $SN_{pr} \oplus$ nSaCEPID
- f4 23 13 7b 51 b8 5e 5a: $EC_{pr} \oplus$ nSaCEPID
- 00 00 00 02: Version
- 01 f4: EC period (500 ms)
- 33 7c 46 ce 51 9b f7 51: Safety Code

B.1.8 Start of transmission of Application Messages protected by EC defence technique

Initiator to Responder

00 20 00 03 01 03 0a 96 00 02 00 00 00 17 64 16 2c a8 90 4b 9d 03 00 00 00 00 15 fe 7e 9d 05 b4 5b 6d

ALE Header is composed of the following fields and is highlighted in green:

- 00 20: Packet Length
- 00 03: Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message is composed of the following fields and is highlighted in yellow:

- 0a: SL Header (DTSaPDU transmitted by Initiator)
- 96: SAI Message Type (0x96 for Application Message pseudo-random sequence)
- 00 02: Sequence Number integer
- 00 00 00 17: Execution Cycle integer
- 64 16 2c a8 90 4b 9d 03: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 00 00 00 00: User Data (Initiator transmits a data field of 4 bytes)
- 15 fe 7e 9d 05 b4 5b 6d: Safety Code

Responder to Initiator

00 1e 00 03 01 03 0b 96 00 01 00 00 02 9a 2a d3 07 12 42 a5 be 72 00 00 7b 69 69 b7 6c 22 d9 0b

ALE Header is composed of the following fields and is highlighted in green:

- 00 1e: Packet Length
- 00 03 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message is composed of the following fields and is highlighted in yellow:

- 0b: SL Header (DTSaPDU transmitted by Responder)
- 96: SAI Message Type (0x96 for Application Message pseudo-random sequence)
- 00 01: Sequence Number integer
- 00 00 02 9a: Execution Cycle integer
- 2a d3 07 12 42 a5 be 72: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 00 00: User Data (Responder transmits a data field of 2 bytes)
- 7b 69 69 b7 6c 22 d9 0b: Safety Code

and so on…

00 20 00 04 01 03 0a 96 00 03 00 00 00 18 23 48 64 f3 c9 94 32 40 00 00 00 00 6d ef eb 3d 0b 56 e0 62

00 1e 00 04 01 03 0b 96 00 02 00 00 02 9b e9 ac 39 68 e2 0c 8c 06 00 00 1f f2 71 83 ee d1 21 fd

B.1.9 Periodic transmission of "Application Message with Req of ACK" and related responses "Application Message with ACK"

From Responder to Initiator:

Responder sends Application Message with Req of ACK

00 1e 00 8a 01 03 0b 97 00 88 00 00 03 21 9b 5e e4 3b b7 49 d3 23 00 00 b1 4e 5f a8 7d a5 b2 65

ALE Header is composed of the following fields and is highlighted in green:

- 00 1e: Packet Length
- 00 8a: Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message with Req of ACK is composed of the following fields and is highlighted in yellow:

- 0b: SL Header (DTSaPDU transmitted Responder)
- 97: SAI Message Type (0x97 for Application Message with Req of ACK pseudo-random sequence)
- 00 88: Sequence Number integer
- 00 00 03 21: Execution Cycle integer
- 9b 5e e4 3b b7 49 d3 23: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 00 00: User Data (Responder transmits a data field of 2 bytes)
- b1 4e 5f a8 7d a5 b2 65: Safety Code

Initiator answers with Application Message with ACK

00 2c 00 72 01 03 0a 98 00 71 00 00 00 86 00 00 03 21 27 d0 d0 46 c1 63 ae 64 f0 98 13 59 ef 01 cb b3 00 00 00 00 79 b8 c9 82 b0 33 47 b3

ALE Header is composed of the following fields and is highlighted in green:

- 00 2c: Packet Length
- 00 72 : Sequence Number
- 01: N/R flag

- 03: Packet Type (Data indication)

Application Message with ACK is composed of the following fields and is highlighted in yellow:

- 0a: SL Header (DTSaPDU transmitted by Initiator)
- 98: SAI Message Type (0x98 for Application Message with ACK pseudo-random sequence)
- 00 71: Sequence Number integer
- 00 00 00 86: Execution Cycle integer
- 00 00 03 21: Echo of Execution Cycle integer (Execution cycle received in AM+REQ to answer to)
- 27 d0 d0 46 c1 63 ae 64: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- f0 98 13 59 ef 01 cb b3: Echo of ($SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID) received in AM+REQ $\oplus EC_{pr}$ $\oplus$ nSaCEPID
- 00 00 00 00: User Data (Initiator transmits a data field of 4 bytes)
- 79 b8 c9 82 b0 33 47 b3: Safety Code

B.2 APL enabled with key length of 256 bits

B.2.1 Configuration Parameters

	Initiator	**Responder**
nSaCEPID	0xA0A0A0A0A0A0A0A0	0x281C21046A5B0106
Cryptographic key AES 256 bits	0x0102030405060708090A0B0C0D0E0F10111213141516171819 1A1B1C1D1E1F20	
Cryptographic key CMAC-AES	0x21222324252627282 92A2B2C2D2E2F30	

Random Number Ra = 0x 82 A7 11 F1 53 AF B0 0B

Random Number Rb = 0x 6b 64 1e 14 8b 21 fb 89

Random Number Rc = 0x AE 29 06 1C ED 29 B0 9F

B.2.2 Initiator trasmits AU1

00 22 00 00 01 01 00 00 00 00 00 00 00 03 02 00 00 00 02 13 FE 61 64 73 99 60 3D 5B 42 22 6C D0 31 FE A6

ALE Header is composed of the following fields and is highlighted in green:

- 00 22: Packet Length
- 00 00: Sequence Number
- 01: N/R flag
- 01: Packet Type (Connect Request)
- 00 00 00 00: Ale Connect Header – (not used)
- 00 00 00 00: Ale Connect Header – (not used)
- 03: Ale Connect Header – Class of Service

AU1 SaPDU is composed of the following fields and is highlighted in yellow:

- 02: SL Header specific for AU1
- 00 00 00: fields not used
- 02: Requested Safety Feature
- 13 FE 61 64 73 99 60 3D 5B 42 22 6C D0 31 FE A6: last 16 bytes encrypted with AES and AES-CMAC composed of:
    - 9a 7e e3 ae 1c a7 c2 dd b4 b8 bc c4 4a a1 35 2c: Random Number R_B encrypted
    - 89 80 82 CA 6F 3E A2 E0 EF FA 9E A8 9A 90 CB 8A: AES-CMAC

B.2.3 Responder transmits AU2

00 25 00 00 01 02 CC CC CC CC 05 00 00 00 02 22 47 52 41 54 49 53 22 09 D4 EA B8 E4 97 56 3B D4 FC B8 B4 27 66 2C 99

ALE Header is composed of the following fields and is highlighted in green:

- 00 25: Packet Length
- 00 00 : Sequence Number
- 01: N/R flag
- 02: Packet Type (Connect Confirmation)
- cc cc cc cc: (not used)

AU2 SaPDU is composed of the following fields and is highlighted in yellow:

- 05: SL Header specifico per AU2
- 00 00 00: campi non utilizzati
- 02: Requested Safety Feature
- 22 47 52 41 54 49 53 22 : fixed value, protocol identifier
- 09 D4 EA B8 E4 97 56 3B D4 FC B8 B4 27 66 2C 99: last 16 bytes AES and AES-CMAC encrypted, composed of:
    - 29 6D AD 33 C5 01 7F 6D C5 D2 95 55 4A 10 A0 63: AES-CMAC
    - 20 B9 47 8B 21 96 29 56 11 2E 2D E1 6D 76 8C FA: RA ⊕ RB ENCRYPTED AES

B.2.4 Initiator transmits AU3

00 15 00 01 01 03 06 77 8A E7 6D B3 14 A6 3C 84 1D 01 7A EF FF E2 FB

ALE Header is composed of the following fields and is highlighted in green:

- 00 15: Packet Length
- 00 01 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)
-

AU3 SaPDU is composed of the following fields and is highlighted in yellow:

- 06: SL Header specific for AU3
- 77 8A E7 6D B3 14 A6 3C 84 1D 01 7A EF FF E2 FB: LAST 16 BYTES ENCRYPTED WITH AES AND AES-CMAC COMPOSED OF:
    - A3 26 24 38 C2 59 68 D7 5E 91 1C B7 80 50 96 96: AES-CMAC
    - D4 AC C3 55 71 4D CE EB DA 8C 1D CD 6F AF 74 6D: RA ⊕ RC AES ENCRYPTED

B.2.5 Responder transmits AR

00 15 00 01 01 03 13 30 D9 A8 08 AC 62 92 6A A5 1D 08 F4 F7 0F F0 B8

ALE Header is composed of the following fields and is highlighted in green:

- 00 15: Packet Length
- 00 01 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

AR SaPDU is composed of the following fields and is highlighted in yellow:

- 13: SL Header specific for AU3
- 30 D9 A8 08 AC 62 92 6A A5 1D 08 F4 F7 0F F0 B8: LAST 16 BYTES ENCRYPTED WITH AES AND AES-CMAC COMPOSED OF:
  - AA A7 4B A6 B0 C5 50 B7 11 A5 B4 30 BD AE C5 94: AES-CMAC
  - 9A 7E E3 AE 1C A7 C2 DD B4 B8 BC C4 4A A1 35 2C: RB ENCRYPTED AES

B.2.6 Initiator transmits Execution Cycle Start (ECS)

00 32 00 02 01 03 0a 91 00 01 00 00 00 10 03 1e 10 65 99 4c 42 8f 31 8e 6a 92 ff d0 d5 db 00 00 00 02 02 58 DD CF 45 D4 6B 94 5A D1 EE 09 E8 73 D3 47 D2 31

ALE Header is composed of the following fields and is highlighted in green:

- 00 32: Packet Length
- 00 02: Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

ECS is composed of the following fields and is highlighted in yellow:

- 0a: SL Header (DTSaPDU transmitted by Initiator)
- 91: SAI Message Type (0x91 for ECS pseudo-random sequence)
- 00 01: Sequence Number integer
- 00 00 00 10: Execution Cycle integer
- 03 1e 10 65 99 4c 42 8f: $SN_{pr} \oplus$ nSaCEPID
- 31 8e 6a 92 ff d0 d5 db: $EC_{pr} \oplus$ nSaCEPID
- 00 00 00 02: Version
- 02 58: EC period (600 ms)
- DD CF 45 D4 6B 94 5A D1 EE 09 E8 73 D3 47 D2 31 : SAFETY CODE AES AND CMAC ENCRYPTED
  - 2C FF A0 1C FE 3D 2D BA C0 1C BE 46 FE 05 9F 83: AES-CMAC
  - 3F A6 EA EF AC 48 00 D5 3F A6 EA EF AC 48 00 D5: SAFETY CODE
  - F1 30 E5 C8 95 A9 77 6B 2E 15 56 35 2D 42 4D B2: SAFETY CODE AES ENCRYPTED

B.2.7 Responder transmits Execution Cycle Start (ECS)

00 32 00 02 01 03 0B 91 00 00 00 00 00 1E 88 BC 81 A4 CA FB A1 A6 36 F3 EC 7F CD DA 7B A4 00 00 00 02 01 F4 B8 33 C4 11 57 B5 80 BB 73 0E 20 22 83 3D 9B D9

ALE Header is composed of the following fields and is highlighted in green:

- 00 32: Packet Length
- 00 02 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

ECS is composed of the following fields and is highlighted in yellow:

- 0b: SL Header (DTSaPDU tranmitted by Responder, different from the one sent by Initiator for theDirection flag bit)
- 91: SAI Message Type (0x91 for ECS pseudo-random sequence)
- 00 00: Sequence Number integer
- 00 00 00 1e: Execution Cycle integer
- 88 bc 81 a4 ca fb a1 a6: $SN_{pr} \oplus$ nSaCEPID
- 36 f3 ec 7f cd da 7b a4: $EC_{pr} \oplus$ nSaCEPID
- 00 00 00 02: Version
- 01 f4: EC period (500 ms)
- B8 33 C4 11 57 B5 80 BB 73 0E 20 22 83 3D 9B D9 : SAFETY CODE AES AND AES-CMAC ENCRYPTED
  - B0 63 BE 40 79 E7 41 21 48 FD 6E A5 F1 AA 52 60: AES-CMAC
  - BA CE 31 65 DB E9 A5 C7 BA CE 31 65 DB E9 A5 C7: SAFETY CODE
  - 08 50 7A 51 2E 52 C1 9A 3B F3 4E 87 72 97 C9 B9: SAFETY CODE AES ENCRYPTED

B.2.8 Start of transmission of Application Messages protected by EC defence technique

Initiator to Responder

00 28 00 03 01 03 0A 96 00 02 00 00 00 16 85 98 58 67 CC F8 41 2B 00 00 00 00 11 22 6D D1 2A 91 57 E8 82 C8 16 53 1A 64 04 86

ALE Header is composed of the following fields and is highlighted in green:

- 00 28: Packet Length
- 00 03: Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message is composed of the following fields and is highlighted in yellow:

- 0a: SL Header (DTSaPDU transmitted by Initiator)
- 96: SAI Message Type (0x96 for Application Message pseudo-random sequence)
- 00 02: Sequence Number integer
- 00 00 00 16: Execution Cycle integer
- 85 98 58 67 cc f8 41 2b: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 00 00 00 00: User Data (Initiator sends a data field of 4 bytes)
- 11 22 6D D1 2A 91 57 E8 82 C8 16 53 1A 64 04 86: SAFETY CODE AES AND AES-CMAC ENCRYPTED

- 47 DE BC D8 4A BF FD C3 D7 6E 97 97 1F 2D 56 E0: AES-CMAC
- A0 E4 68 61 42 8A 19 1C A0 E4 68 61 42 8A 19 1C: SAFETY CODE
- 56 FC D1 09 60 2E AA 2B 55 A6 81 C4 05 49 52 66: SAFETY CODE AES ENCRYPTED

Responder to Initiator

00 26 00 03 01 03 0B 96 00 01 00 00 00 1F 34 48 23 18 95 0E 2F 52 00 00 A0 F9 E3 BC 81 18 74 F3 DA 7A C4 42 E0 2C A0 1F

ALE Header is composed of the following fields and is highlighted in green:

- 00 26: Packet Length
- 00 03 : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message is composed of the following fields and is highlighted in yellow:

- 0b: SL Header (DTSaPDU transmitted by Responder)
- 96: SAI Message Type (0x96 for Application Message pseudo-random sequence)
- 00 01: Sequence Number integer
- 00 00 00 1f: Execution Cycle integer
- 34 48 23 18 95 0e 2f 52: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 00 00: User Data (Responder trasmits a data field of 2 bytes)
- d4 d8 f1 49 aa b0 d9 ce f1 5d b7 fb e2 8d 84 09: Safety Code encrypted
- A0 F9 E3 BC 81 18 74 F3 DA 7A C4 42 E0 2C A0 1F: SAFETY CODE AES AND AES-CMAC ENCRYPTED
    - 16 29 C5 42 FD 93 24 C9 5E 8E 8D B4 9E A1 DF 77: AES-CMAC
    - 6F 97 86 C2 64 5B 8F 66 6F 97 86 C2 64 5B 8F 66: SAFETY CODE
    - B6 D0 26 FE 7C 8B 50 3A 84 F4 49 F6 7E 8D 7F 68: SAFETY CODE AES ENCRYPTED

and so on…

Initiator to Responder

00 28 00 04 01 03 0a 96 00 03 00 00 00 17 0d ed 06 ee 98 e5 cb cb 00 00 00 00 5F FF D0 FB 83 35 6E 73 DB 83 38 12 DF 22 7E CA

Responder to Initiator

00 26 00 04 00 03 0b 96 00 02 00 00 00 20 69 a3 ab 99 92 58 6b 2f 00 00 79 2D BF D4 51 CA C3 D3 5A 04 A6 9B A5 4A 98 7C

B.2.9 Periodic transmission of "Application Message with Req of ACK" and related responses "Application Message with ACK"

Example of message sent from Initiator to Responder:

Initiator sends Application Message with Req of ACK

00 28 01 2A 01 03 0A 97 01 29 00 00 01 3D D1 9F 2A 45 46 D2 98 9C 00 00 00 00 2B F0 EE 3A 99 BF F3 72 A8 0E F4 32 FB 91 EC E9

ALE Header is composed of the following fields and is highlighted in green:

- 00 28: Packet Length
- 01 2a: Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message with Req of ACK is composed of the following fields and is highlighted in yellow:

- 0a: SL Header (DTSaPDU transmitted by Initiator)
- 97: SAI Message Type (0x97 for Application Message with Req of ACK pseudo-random sequence)
- 01 29: Sequence Number integer
- 00 00 01 3d: Execution Cycle integer
- d1 9f 2a 45 46 d2 98 9c: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 00 00 00 00: User Data (Initiator sends a data field of 4 bytes)
- ef 85 f6 c2 29 7e 1a b5 91 8a cc 42 ef cf c6 b6: Safety Code encrypted
- 2B F0 EE 3A 99 BF F3 72 A8 0E F4 32 FB 91 EC E9: Safety Code criptato AES and AES-CMAC encrypted
    - 84 0A 6F 07 47 78 7B 32 AA 8B B9 B3 97 35 D8 33: AES-CMAC
    - 4C 48 34 36 31 33 9D 26 4C 48 34 36 31 33 9D 26: Safety Code
    - AF FA 81 3D DE C7 88 40 02 85 4D 81 6C A4 34 DA: Safety Code AES encrypted

Responder answers with Application Message with ACK

00 32 01 6F 01 03 0B 98 01 6D 00 00 01 8B 00 00 01 3D 18 06 5E 8D 76 0F 3A 73 03 40 D5 47 A9 53 E4 36 00 00 DD DB 41 0C 62 CE F8 2C AA 97 3A B8 0E ED 24 8E

ALE Header is composed of the following fields and is highlighted in green:

- 00 32: Packet Length
- 01 6f : Sequence Number
- 01: N/R flag
- 03: Packet Type (Data indication)

Application Message with ACK is composed of the following fields and is highlighted in yellow:

- 0b: SL Header (DTSaPDU transmitted by Responder)
- 98: SAI Message Type (0x98 for Application Message with ACK pseudo-random sequence)
- 01 6d: Sequence Number integer
- 00 00 01 8b: Execution Cycle integer
- 00 00 01 3d: Echo di Execution Cycle integer (Execution cycle received in AM+REQ to answers to)
- 18 06 5e 8d 76 0f 3a 73: $SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID
- 03 40 d5 47 a9 53 e4 36: Echo di ($SN_{pr} \oplus EC_{pr} \oplus$ nSaCEPID) received in AM+REQ $\oplus EC_{pr} \oplus$ nSaCEPID
- 00 00: User Data (Responder transmits a data field of 2 bytes)
- DD DB 41 0C 62 CE F8 2C AA 97 3A B8 0E ED 24 8E: Safety Code AES and AES-CMAC encrypted
    - E7 BE 17 DB 35 58 94 4C 69 7D 78 E8 29 DC 5E A8: AES-CMAC
    - 21 14 11 3F 05 7B 4C E3 21 14 11 3F 05 7B 4C E3: Safety Code
    - 3A 65 56 D7 57 96 6C 60 C3 EA 42 50 27 31 7A 26: Safety Code AES encrypte

Annex C
(informative)

Examples of Ex management

Examples related to Ex management versus Incr value.

Ex	Next Ex	Incr
1	1,5	0
1,5	2	1
2	2,5	0
2,5	3	1
3	3,5	0
3,5	4	1
4	4,5	0
4,5	5	1
5	5,5	0
5,5	6	1
6	6,5	0
6,5	7	1
7	7,5	0
7,5	8	1
8	8,5	0
8,5	9	1
9	9,5	0

Table C.1- EC receiver = 250 ms, EC sender = 500 ms, R equal to 0,5

Ex	Next Ec	Incr
1	1,6	0
1,6	2,2	1
2,2	2,8	0
2,8	3,4	1
3,4	4	1
4	4,6	0
4,6	5,2	1
5,2	5,8	0
5,8	6,4	1
6,4	7	1
7	7,6	0
7,6	8,2	1
8,2	8,8	0
8,8	9,4	1
9,4	10	1
10	10,6	0
10,6	11,2	1

Table C.2 - EC receiver = 300 ms, EC sender = 500 ms, R equal to 0,6

1947

Ex	Next Ec	Incr
1	2,12	1
2,12	3,24	1
3,24	4,36	1
4,36	5,48	1
5,48	6,6	1
6,6	7,72	1
7,72	8,84	1
8,84	9,96	1
9,96	11,08	2
11,08	12,2	1
12,2	13,32	1
13,32	14,44	1
14,44	15,56	1
15,56	16,68	1
16,68	17,8	1
17,8	18,92	1
18,92	20,04	2

1948 **Table C.3 - EC receiver = 336 ms, EC sender = 300 ms, R equal to 1,12**

1949

Ex	Next Ec	Incr
1	2,2	1
2,2	3,4	1
3,4	4,6	1
4,6	5,8	1
5,8	7	2
7	8,2	1
8,2	9,4	1
9,4	10,6	1
10,6	11,8	1
11,8	13	2
13	14,2	1
14,2	15,4	1
15,4	16,6	1
16,6	17,8	1
17,8	19	2
19	20,2	1
20,2	21,4	1

1950 **Table C.4 - EC receiver = 600 ms, EC sender = 500 ms R equal to 1,2**

Ex	Next Ec	Incr
1	2,428571429	1
2,428571429	3,857142857	1
3,857142857	5,285714286	2
5,285714286	6,714285714	1
6,714285714	8,142857143	2
8,142857143	9,571428571	1
9,571428571	11	2
11	12,42857143	1
12,42857143	13,85714286	1
13,85714286	15,28571429	2
15,28571429	16,71428571	1
16,71428571	18,14285714	2
18,14285714	19,57142857	1
19,57142857	21	2
21	22,42857143	1
22,42857143	23,85714286	1
23,85714286	25,28571429	2

1951 **Table C.5 - EC receiver = 500 ms, EC sender = 350 ms, R equal to 1,428571429**

1952

Ex	Next Ec	Incr
1	1,7	0
1,7	2,4	1
2,4	3,1	1
3,1	3,8	0
3,8	4,5	1
4,5	5,2	1
5,2	5,9	0
5,9	6,6	1
6,6	7,3	1
7,3	8	1
8	8,7	0
8,7	9,4	1
9,4	10,1	1
10,1	10,8	0
10,8	11,5	1
11,5	12,2	1
12,2	12,9	0

1953 **Table C.6 - EC receiver = 350 ms, EC sender = 500 ms, R equal to 0,7**

Ex	Next Ec	Incr
1	1,833333333	0
1,833333333	2,666666667	1
2,666666667	3,5	1
3,5	4,333333333	1
4,333333333	5,166666667	1
5,166666667	6	1
6	6,833333333	0
6,833333333	7,666666667	1
7,666666667	8,5	1
8,5	9,333333333	1
9,333333333	10,16666667	1
10,16666667	11	1
11	11,83333333	0
11,83333333	12,66666667	1
12,66666667	13,5	1
13,5	14,33333333	1
14,33333333	15,16666667	1

1954 **Table C.7 - EC receiver = 500 ms, EC sender = 600 ms, R equal to 0,833333333**

1955

1956

Annex D
(informative)

Example of APL test

If the APL is positioned outside the Safety Related Equipment (see Figure 3-3), a diagnostic mechanism for the APL must be defined.

In the following paragraphs, an example of service primitives used by the safety layer for APL diagnostics and the related PDUs used will be described.

The diagnosability of APL refers to an architecture of the encryption system of the type shown in Figure 5-3 or Figure 5-4, in which the encryption and decryption processes, carried out separately on a non-secure machine or on a non-intrinsically safe part of a safety-related equipment, are tested separately with PDUs containing patterns known a priori (for example installed in the SL during configuration). In this way, the message that the cryptographic module (encryption/decryption) returns to the SL is different from the one sent, but still known.

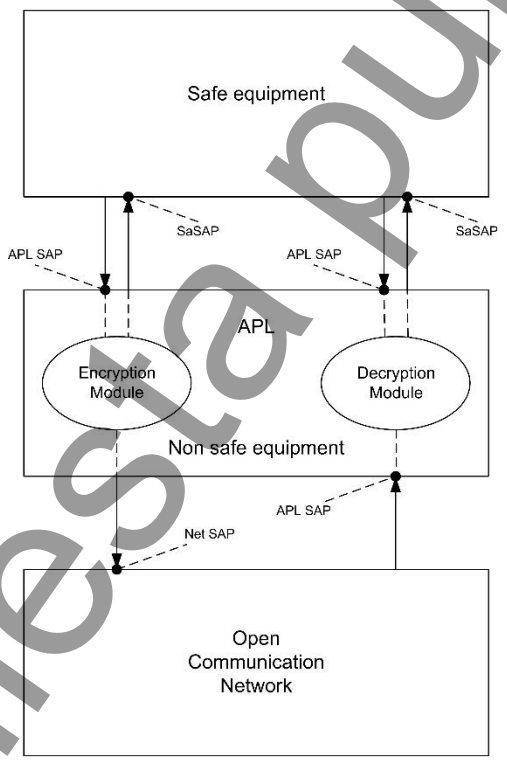


Figure A.1 — Figure title

Figure A.1 — Figure title

Figure 6-7 - Reference Architecture for APL Diagnostics

The number of APL diagnostic messages to be used in the SL is to be defined. The important thing is that it is such as to allow a sufficient cyclicality to avoid diagnostic errors due to undue memorization.

It was decided to use fixed keys for APL diagnostic test messages to avoid having to change the configuration of the SL if one or more keys, used by the APL for the encryption/decryption of messages, change,.

Diagnostic messages must be stored in the SL in both plain and encrypted form, the mechanism of operation is shown in Figure 7-8.

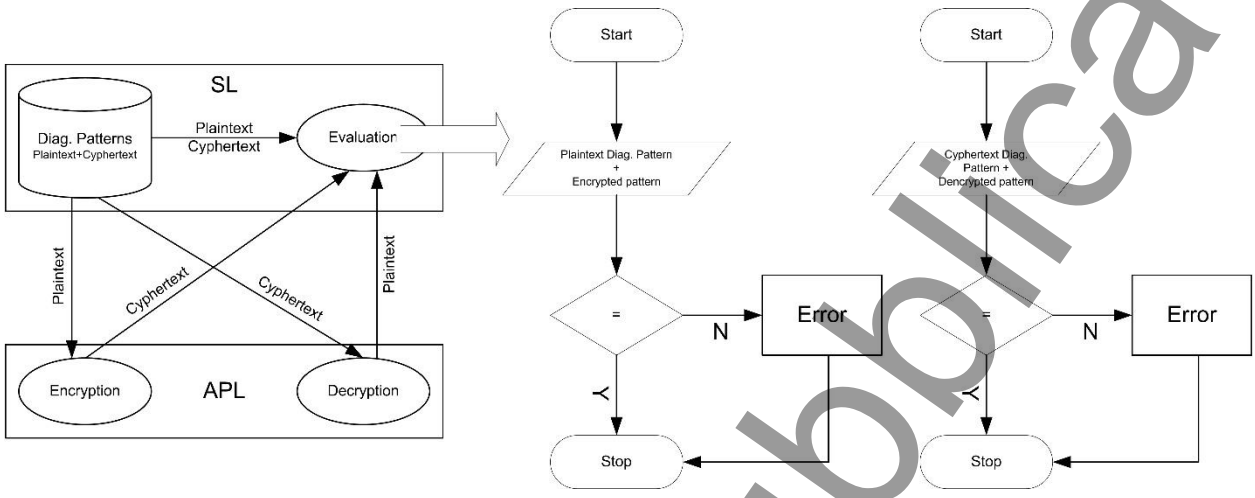


Figure 6-8 - Reference Architecture for APL Diagnostics

D.1 Cryptography module diagnostic test

The SL sends a clear diagnostic message to the encryption module. The module under test encrypts it using the appropriate key fixed and shared with the SL and sends it back to the SL. It compares the encrypted value returned to it with the corresponding value stored during configuration. If the two messages compared are the same, the encryption module has passed the diagnostic test, otherwise it is declared non-functional and appropriate countermeasures must be taken.

D.2 Decryption module diagnostic test

The SL sends an encrypted diagnostic message to the decryption module. The module under test decrypts it using the appropriate key fixed and shared with the SL and sends it back to the SL. It compares the plaintext value returned to it with the corresponding value stored during configuration. If the two messages compared are the same, the decryption module has passed the diagnostic test, otherwise it is declared non-functional and appropriate countermeasures must be taken.

D.3 Service primitives for APL diagnostics

APL diagnostics are carried out using the service primitives shown in Figure 6-9:

1. T-Diag.Request – Request for diagnostics, sending a specific PDU

2. T-Diag.Indication – Receipt of the diagnostic request and related PDU

3. T-Diag.Response – Transmission of the diagnostic response and related PDU

4. T-Diag.Confirm – Receiving diagnostic response and related PDU

The APL diagnostics, with reference to the architecture of case 1, must be a confirmed communication, as can be seen from the previous list, since it involves the bidirectional data exchange between the safe and non-secure part of the system.

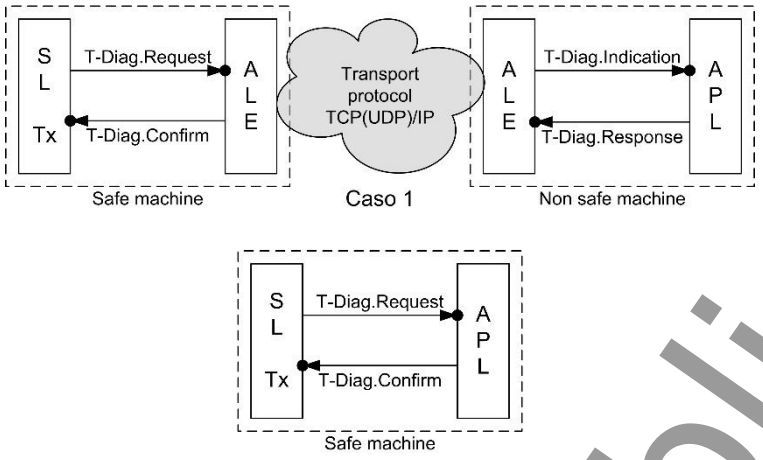


Figure 6-9 – APL Diagnostics

D.4 The exchange of APL diagnostic PDUs

With regard to APL diagnostic PDUs, three different classes of PDUs must be distinguished, which may affect APL:

1. PDUs used in communication (AU1, AU2, AU3, DT, DI, AR)
2. Cryptor Diagnostic PDU (ED)
3. Decrypter Diagnostic PDU (DD)

This distinction can be achieved by using the bits of the MTI field in the header byte of the SL as described in Table D.1.

Bit 5	Bit 4	Bit 3	Bit 2	PDU
0	0	0	0	
0	**0**	**0**	**1**	**AU1**
0	**0**	**1**	**0**	**AU2**
0	**0**	**1**	**1**	**AU3**
0	1	0	0	
0	**1**	**0**	**1**	**DT**
0	1	1	0	
0	1	1	1	
1	**0**	**0**	**0**	**DI**
1	**0**	**0**	**1**	**AR**
1	0	1	0	

1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	**ED**
1	1	1	1	**DD**

Table D.1 – APL Message Classes and Diagnostics

For example, 1110 and 1111 configurations are not used and can be useful for encoding encrypting diagnostic messages (ED) and decrypting diagnostic messages (DD), respectively.

The APL diagnostic PDUs can be composed as depicted in Figure D.10-1.

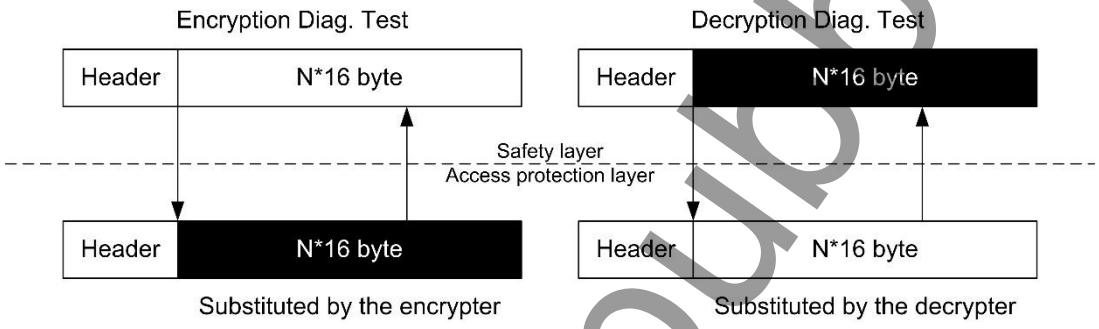


Figure D.10-1 – Diagnostics PDU, type 1

The SL sends plaintext or encrypted bytes to the APL. The module under test replaces the suitable bytes, taking into account its output, and sends the packet back to the SL. In this case, the diagnostic comparison should be performed only on the two distinct parts of the package.

The size of the test package must be equal to the maximum packet size provided in the specific configuration to ensure that encryption is applied to all bytes transmitted/received.

Encrypter and Decrypter must have the same behavior as the nominal case of packet transmission and receipt:

Encrypter:

- Encrypt the last 16 bytes with AES
- Calculate CMAC on the rest of the package (in this case Header)
- Put in XOR the two contributions calculated above

Decrypter:

- Calculate CMAC on the packet (in this case Header) except for the last 16 bytes
- Put in XOR the CMAC with the final 16 bytes
- Decrypt the last 16 bytes with AES

D.5 APL Diagnostic scheduling

The APL diagnostic process is performed as described in Figure D.10-2.

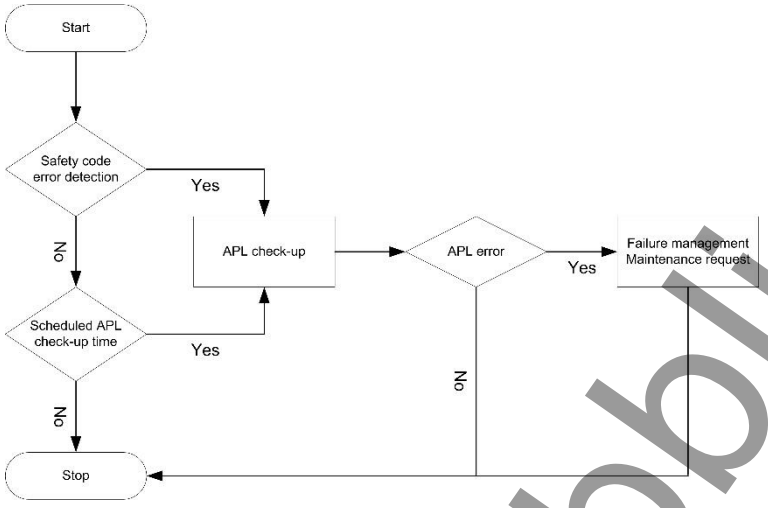


Figure D.10-2 - Algorithm for APL functional verification

In each processing cycle, the APL functional verification procedure may take place in two different cases:

1.) the safety layer, if it detects an error in the safety code, or

2) each TOT cycles (for a period of your choice as long as < 60 minutes), if, in that cycle, an APL survey has been scheduled.

In the event that there is an error of the cryptographic modules, transmission over an open network is suspended and the intervention of an entity that performs maintenance tasks is invoked.

This choice was made because, by encrypting the safety code, it is the safety layer that takes charge of the diagnosability of the APL in every vital input and output message, as cryptographic errors are reflected in safety code errors, detected by the safety layer. In this way, the SL is the entity that also operates the real access protection function, revealing any flaws in the cryptogram.

Table D.2 shows a set of possible risks incurred in the realization of an APL with separate hardware and not in the safety architecture. For each of them, the consequences and countermeasures taken are identified.

Risks	Consequences	Countermeasures
Crash link with APL (unavailability of the link)	Bidirectional data does not pass between SL and APL	Doubling of the link between SL and APL
Crash APL	Unavailability of cryptographic functions	APL diagnostics and maintenance
Malfunction of encryption and / or decryption software not common mode	Incorrect or missing encryption/decryption of vital messages	
Common mode software malfunction of cryptographic modules	Incorrect application of encryption not revealed by diagnostics	Software development criteria (diversification, independence, modularity…)

Table D.2 – Risks for a separate APL

Annex E
(informative)

RISK ANALYSIS and SAFETY requirements

From [EN 50159-2] Table 1 – Threats/Defences matrix

	Defences							
Layers →	*SAI*	*SAI*	*SAI*	*SAI*	*SL*	*SL*	*SL*	*APL*
Threats	**Sequence number**	**Time stamp**	**Time-out**	**Source and destination identifiers**	**Feed-back message**	**Identification procedure**	**Safety code**	**Cryptographic techniques**
Repetition	X (C)	X						
Deletion	X (B)							
Insertion	X (C)			X (D)	X (F)	X (F)		
Re-sequence	X (C)	X						
Corruption							X (E)	X
Delay		X (A)	X (A)					
Masquerade				(D)	X	X		X (G)

In table 1, the "X" indicate the protections suggested by the Standard; the numbers instead show those identified in PVS on the basis of the analysis shown below.

The right column (REF) shows the tracing to the requirements described in [SRS].

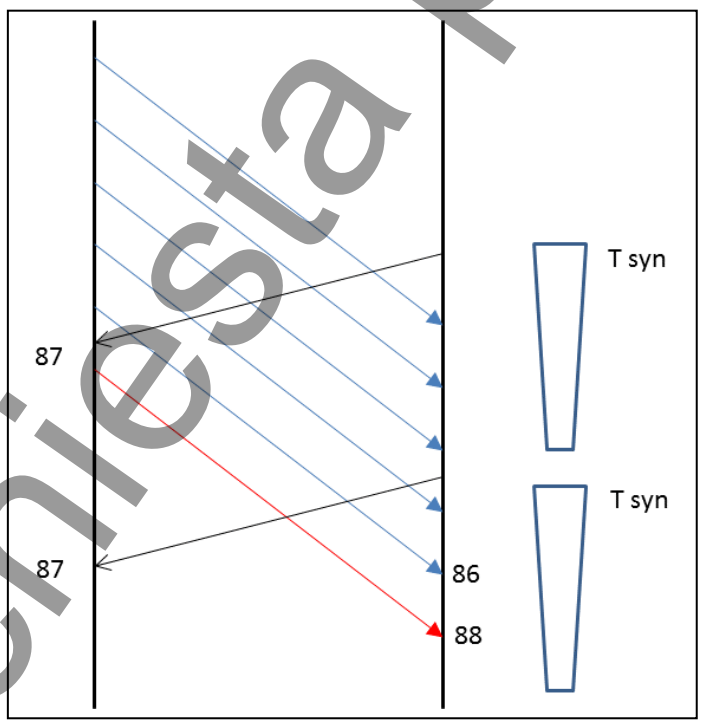
Table A - Threat: Delay / Defence: Time-stamp, Time-out [SAI level]

Risk A1	the Rx application does not receive any data.	Ref to requirements
Assumption	It is assumed that the lack of information does not involve any safety risk at the level of the receiving node. The receiving node, if needed, will apply a restrictive state in that situation. Therefore the risk has only effect on the availability.	
Risk A2	A delayed message is unduly accepted after the maximum time out	
Assumption	Tx operates cyclical transmissions, with a cycle known to Rx	
Requirement A2	Each Tx transmits its own EC. Initial exchange of ECs with msg 81-81 or 91-91. Rx locally reproduces the cycle time of Tx, compares it with the one received and discards the msgs that exceed a maximum tolerance N_MAX.	SAI_029, SAI_030, SAI_031, SAI_032, SAI_033, SAI_034
Note	This protection combines the "time-out" (evaluated on the basis of the locally reproduced transmitter time) with the "relative time stamp" (see [50159], Table 1).	
Risk A3	The message 81/91 is late. Receipt of a 81/91 msg forces the initial values of EC, SN. This entails an intrinsic risk associated with the msg, namely that it itself is out of time.	
Note	This is a "second level risk" because derives from a possible shortcoming of the requirement A2.	

Requirement A3	The initialization of the ECs requires that the sequence Initiator:ECS - Responder:ECS - Initiator:first.appl.msg be strictly respected both in the order and in the foreseen times. A double synchronization is therefore applied: - Initiator : send ExecutionCycleStart (ECS) and verify that Responder responds with its ECS within a time Tsyn ; - Responder: after sending its own ECS, check that the Initiator responds with the first application Msg within a Tsyn time.	SAI_032
Note	Initiator 's ECS message is not time-verified and therefore it does not bring application data.	
Config. Req.	The configuration values Tsyn and N shall be justified.	CONF_001
Special condition	In case of delay or even in nominal conditions (if the cycle time of Rx is lower than that of Tx), it may happen that there are no new data received and yet that the last packet received is still valid (t<N_MAX).	
Suggested solution	In case of lack of updated data and for (t<N_MAX), if PVS-Rx must supply its application with the data received with local cyclicity, it could supply the application with an indication which confirms the validity of the last message received. There is no intention to standardize the PVS-Rx - Application interface, therefore this is not a mandatory requirement: alternative solutions are admissible. For example, PVS-Rx could not supply any information, leaving the receiving application to evaluate the validity of the last message received. In effect, this means moving the (t<N_MAX) check to the application level.	APP_012
Risk	A delayed message is accepted because the sequence has been slowed down during a valid session (slipping delay).	
Note	The " slipping delay" is a progressive temporal spacing of messages, such that the delay of each single message is still admissible if considered in relation to the previous message, but no longer if evaluated with respect to the absolute time in which the message was generated.	
Requirement	Periodic synchronization with msg 87-88 or 97-98 is required to reduce the risk of slipping delay. With these messages, the node sending the request (msg 87/ 97) receives the current values of EC, SN of the other (msg 88/ 98) and can thus resynchronize itself.	SAI_035 SAI_036 SAI_037 SAI_038 SAI_039
Second level risk	msg 88/98 late. The 88/98 msg forces new values of EC,SN (as already happened with the initial 81/91 msg). This entails an intrinsic risk associated with the msg, namely that it itself is out of time. The protection is similar to what happens in the initial synchronization (see 0): whoever sends the 87/97 request activates a timer with time Tsyn within which he waits for the 88/98 reply.	
Note	The re-sync procedure is one-way, unlike the initial sync procedure, which is two-way.	
Risk	The sequence must be strictly respected. For example, scenarios like the following should be avoided: node B sends 87 and throws Tsyn (time-out). When the time-out expires, he sends a new 87. He then receives 88, which could be the answer to the first or second 87. This uncertainty could invalidate the current data and all subsequent ones. See figure 1 at the end of this table. It is noted that in the "PRS option (pseudorandom values)" this risk is excluded by the fact that 98 contains the echo of 97 (EC/SN of the applicant). In this way the applicant can correctly associate the 98 with the 97 that generated it.	
Possible solution	The sequence 87-88 must be respected without repetitions (87…87 – 88) and within the foreseen times (Tsyn). In the event of a time-out after request 87, the requesting node must restart the connection, without forwarding a new request 87. Remark: in case of TCP communication (since it manages retransmissions etc) it is legitimate to think that setting no retry on this time-out has no	Req_AC_006

	impact on availability. In case of use of UDP communication (where it is possible to lose packets) it is instead possible that the absence of retry could have an impact on the availability of the connection. An alternative solution (not implemented) would have been to echo in 87-88 the time of the requester, as it is done in 97-98.	
Configuration requirement	the re -synchronization period (ReqACKPeriod) must be defined and justified on the basis of the Tx and Rx cycle times and the maximum possible delay in the transmission system.	CONF_001
Risk	PVS-Rx incorrectly delivers outdated data to the application.	
No requirement	Ensuring this does not happen is an implementation requirement of PVS-Rx.	
Risk	PVS-Rx incorrectly delivers a received packet that is different from the last one.	
Application requirement	When Tx has a smaller cycle than Rx, Rx could receive more messages in the same cycle. In this case, a possible solution is to deliver either the last one received (usual solution for indications) or all of them (usual solution for controls). The solution is application-dependent.	APP_011, APP_014
Risk	The Rx application incorrectly uses outdated data.	
Application requirement	Requirement: this error is independent of the presence of PVS and must be protected at the application level.	
Risk	In the absence of updated data , the Rx application receives and uses obsolete data.	
Application requirement	In case of missing valid data (t>N_MAX), PVS-RX does not pass any information to the application level. In the absence of data, the Rx Application activates restrictive conditions.	

2077



2078

2079 Figure 1

2080

2081 **Table B - Threat: Deletion / Defense : Sequence Number [SAI level]**

2082

Assumption	The loss of a certain number of messages, up to a maximum of configurable N-1, is allowed. N will be equal to 1 when no loss is admittable.	
Remark	There is no safe retransmission request mechanism in case of loss of a msg. However, at the transmission level retransmission features (e.g. TCP) might be present.	
Risk	The Rx application uses a msg after having lost more than N-1 consecutive ones.	
Requirement	Each Tx transmits its SN. Initial exchange of SNs with msg 81/81 or 91/91. Rx locally verifies the increase of SN, calculating the delta (difference) with respect to the last valid message. Messages with SN exceeding the tolerance window (delta > N, with N defined in the configuration phase) are rejected and lead to disconnection.	SAI_044, SAI_045, SAI_054
Requirement	the law of progress of the PR-SN, being linear, imposes that PR-SN is not initialized to 0.	SAI_011

2083

2084

2085 **Table C - Threats: Repetition, Insertion, Resequence / Defense : Sequence Number [SAI**
2086 **level]**

2087

	Threat: Repetition, Insertion, Resequence / Defense : Sequence Number	
	Risk: the Rx application uses a message in the wrong sequence.	
	Prerequisite: Tx transmits its SN. Initial exchange with msg 81/81 or 91/91. Rx locally verifies the increase of SN, calculating the delta (difference) with respect to the last valid message. Messages with decreased SN (delta = negative) are rejected.	SAI_043
	Note: the protections offered by SN could also be attributed to EC, which is likewise an ordered sequence of numbers. The only difference is that SN can accelerate if more messages are sent per cycle. If only EC applied, these messages would all have the same EC so their sequential ordering would no longer be possible. In conclusion, the use of the EC alone would be possible if the risk associated with repetition, insertion, resequencing of messages belonging to the same cycle is tolerated.	
	Constraint. In the PRS option (pseudorandom values), the values of PR-EC and PR-SN are two pairs of pseudo-random counters. These values are superimposed in XOR: • PR-SN = ($PR\text{-}SN_1$, $PR\text{-}SN_2$) • PR-EC = ($PR\text{-}EC_1$, $PR\text{-}EC_2$) Transmitted value: - $PR\text{-}EC_1$ XOR $PR\text{-}SN_1$ XOR $nSaCEPID_1$ - $PR\text{-}EC_2$ XOR $PR\text{-}SN_2$ XOR $nSaCEPID_2$ Since the progress algorithm of the two pseudo-random counters is the same on each channel, and since the two counters do not have the same step, it	SAI_022 SAI_014

	may happen that, even if starting with different values, PR-ECx and PR-SNx converge on the same value (consequently xor = 0). To avoid this happening on both channels at the same time, the offsets between the channels must be different: Δ_1 (PR-EC$_1$, PR-SN$_1$) ≠ Δ_2 (PR-EC$_2$, PR-SN$_2$) where Δ is the distance in the pseudorandom sequence.	

2088

2089

2090 **Table D - Threat: Masquerade by error / Defense : Source and Dest . Identifiers [SAI level]**

2091

	Threat: Masquerade by error / Defense : Source and Dest . Identifiers	
	Risk: the Rx application attributes the message to a different sender.	
	Requirement: Node ID (nSACEPID), specific 64-bit value for each node. For communication between A and B, the IDs of both nodes are used.	SL_001
	Remark: The node identifier is known to the sender and receiver at the configuration level. Therefore, it is transmitted implicitly (contributes to the safety code but is not transmitted in its own field). Advantages: bandwidth reduction.	
	Constraint. Management of the nSACEPIDs should be envisaged to guarantee their uniqueness on all specific applications. Note: the technique protects against error masquerade (can also be seen as insertion protection). It is not intended to protect against malicious masquerade (for which encryption is used) and therefore no secret handling of nSACEPID values is needed.	AC_004

2092

2093

2094 **Table E - Threat: Corruption / Defense: Safety Code [SL level]**

2095

	SL-LEVEL PROTECTIONS	
	Threat: Corruption / Defense: Safety Code	
	Risk: the Rx application uses a message with corrupted data.	
	Requirement: Safety Code with appropriate sizing.	SL_009 and subclause F.4

2096

2097 **Table F Threat: Insertion/ Defense : Feedback msg & Identification Procedure [SL level]**

2098

Risk:	the Rx application uses an obsolete message because it was generated in a previous session.	
Solution	: You must ensure that the messages belong to the current session.	
Requirement:	During the connection initialization procedure, each pair of nodes (here A and C for simplicity) generates a pair of random numbers Ra and Rc when the session is opened which are uniquely assigned to the session itself. During initialization, the two random numbers are exchanged between the two nodes.	SL_011
Requirement	: to prevent Ra and Rc from being known by other nodes, or from being reused in other sessions, the two numbers are exchanged in encrypted form via XOR with a third number Rb .	SL_011
Requirement:	In the subsequent 81/91 message, node A includes the random number Rc (received from C) and vice versa. Ra and Rc ensure that the messages belong to the current session. They are added in XOR on the safety code. Ra and Rc are equally applied to all subsequent msgs, whose belonging to the current session is also guaranteed by the EC/SN counters.	SL_011
	constraint. Ra, Rb , Rc must be 8 bytes in length (superimposable to the Safety Code), different from each other and from the value 0. As the sessions vary, the selection of these values must be pseudo-random.	SL_011
Limitations:	This solution implies point-to-point communication	
Note:	Being random numbers, having the same size as the safety code, it is not necessary that Ra and Rc are protected in transmission by the safety code.	

2099

2100 **Table G Threat: Intentional masquerade / Defense: Crypto [APL level]**

2101

2102

Risk	the Rx application uses an intentionally altered msg	
Requirement:	Encryption (only for open transmission systems).	APL_004
Constraint:	APL level, optional and separate from SL. APL can be outside of safety-related equipment. In this case, "the continued correct functioning of the access protection processes shall be checked with adequate safety-related techniques for the application" [50159, C2]	
Note:	This constraint characterizes PVS with respect to [Subset 98].	
Configuration requirement:	Selection of key length (192 or 256 bit).	AC_003

2103

2104

2105

2106

Annex F
(informative)

Safety demonstration

F.1 Source identifiers (nSaCEPID)

The PVS uses 64-bit identifiers of the flows from source to destination, called nSaCEPID.

Each instance (connection) of an host has assigned an unique nSaCEPID-LOC, known only to the instance itself and to the remote instance communicating with it.

The identification of the source through nSACEPID is obtained by making the Safety Code of a packet to be transmitted a function of nSaCEPID-REM, i.e. of nSaCEPID-LOC of the instance to which the message is to be sent. In this way only the instance which has been assigned nSaCEPID-LOC equal to nSaCEPID-REM used by the transmitter to calculate the Safety Code will be able to correctly verify the integrity of the packet received and then process it.

The nSaCEPIDs shall satisfy the following requirements:

1) the size of the nSaCEPID must be 8 bytes, a couple of 4 bytes values;
2) Each 4-byte part of the nSaCEPID (e.g. high part) must be at a distance of Hamming 6 from the corresponding parts (e.g. high) of the nSaCEPID of the other communication streams.

Through the requisites listed above it is possible to ensure an upper limit to the probability that an initial value ($nSaCEPID_a$) can be transformed into another one ($nSaCEPID_b$). This is achieved by imposing a minimum distance d_{min} between the possible values.

The probability of nSaCEPID transformation, i.e. the d-bits transformation, can be evaluated as:

$$P(nSaCEPID_a \rightarrow nSaCEPID_b) = \frac{p^d \cdot (1-p)^{n-d}}{\binom{n}{d}} = p^d \cdot (1-p)^{n-d} \frac{(n-d)!}{n!} \cdot d!$$

To identify the maximum value of the probability as a function of p, keeping d as a parameter, we calculate the derivative of P(A→B) with respect to p:

$$\frac{\partial P(A \rightarrow B)}{\partial p} = p^{(d-1)}(1-p)^{(n-d-1)}(d-np)\frac{(n-d)!d!}{n!}$$

Then the maximum value of P(A→B) is obtained when:

$$p = \frac{d}{n}$$

$$P(nSaCEPID_a \rightarrow nSaCEPID_b)_{MAX} = \frac{d^d \cdot (n-d)^{n-d}}{n^n} \cdot \frac{(n-d)!}{n!} \cdot d!$$

By selecting the nSaCEPID as 32-bit (n) values with the property of having a Hamming distance (d) of at least 6 and a maximum of 26 bits, from the previous relation we obtain (n=32 and d=6):

$P(\text{nSaCEPID}_a \rightarrow \text{nSaCEPID}_b)$= 2.169E-13.

Since nSaCEPID is composed of a pair of 32-bit values, we obtain that the transformation probability is given by

$P(\text{nSaCEPID}_a \rightarrow \text{nSaCEPID}_b)$= 4.704561E-26.

This value is less than 2^{-64} (5.42E-20).

F.2 Linear Feedback Shift Register proprieties

F.2.1 Linear Feedback Shift Register proprieties

The PR-SN, PR-EC and the SL_CRC are based on LFSR; an LFSR is a shift register whose inputs bit are driven by the XOR of some bits of the overall shift register value.

LFSR operator, performs the polynomial division operation between:

- the polynomial whose coefficients are given by the input bit sequence, which is the dividend polynomial;

- the characteristic polynomial of the operator, which is the divisor polynomial.

The Linear Feedback Shift Register (LFSR), can be implemented in hardware or simulated in software.

As a result of an operation on LFSR the contents of the register are considered, which mathematically coincides with the remainder of the polynomial division operation.

The contents of LFSR can be interpreted as a data compression operation: the exact output value certifies the exactness and correct sequence of the input data.

In general, data compression techniques generate a kind of error called Aliasing, i.e. different input sequences can produce the same output and therefore be indistinguishable. This phenomenon is quantified by the AEP (Aliasing Error Probability).

The following paragraphs provide general information on LFSR, Aliasing and AEP and then introduce the specific properties of the LFSR used in the calculation of the SL_CRC and the progress of the PR-SN and PR-EC values.

F.2.2 LFSR Structure

Figure 10-1 depicts the general structure of an internal XOR-LFSR: the coefficients c_i and the contents y_i of the register are defined on the two elements Galois field GF(2); they can therefore assume values 0 or 1. In the register they are combined with each other using the XOR operator, linear on GF(2). The LFSR is therefore linear.

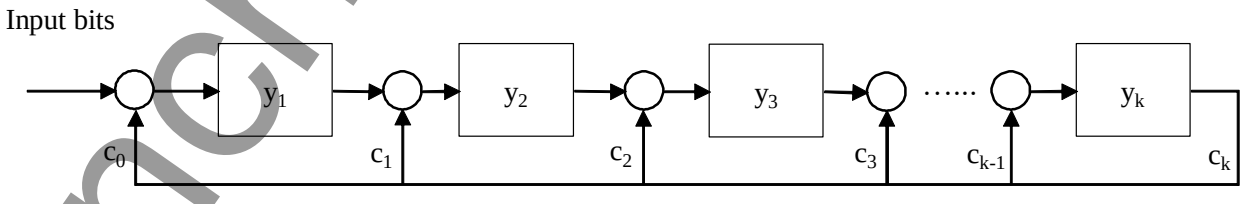


Figure 10-1 – Internal XOR-LFSR

At each clock, a new bit is introduced into the register and the contents of all stages are shifted, n clocks are required to process a sequence of n bits.

The state of the LFSR can be evolved autonomously, i.e. with an input sequence consisting of all 0s (this is the way in which they are used to advance the sequences relating to PR-SNs and PR-ECs).

The implementation of the feedback connections in an LFSR can be represented by a polynomial, called characteristic polynomial of the register, of degree equal to the number of stages of the register and whose coefficients are equal to the weight of the connections themselves:

$$f(x) = \sum_{i=o}^{k} c_i x^i$$

where $c_0 = 1$ e $c_k = 1$ (i.e. the feedback from output to input is always present).

The feedback of the output on the intermediate stages is instead linked to the value of the coefficients c_i.

The dual-LFSR of internal XOR-LFSR depicted in Figure 10-1 is the external XOR-LFSR depicted in Figure 10-2.

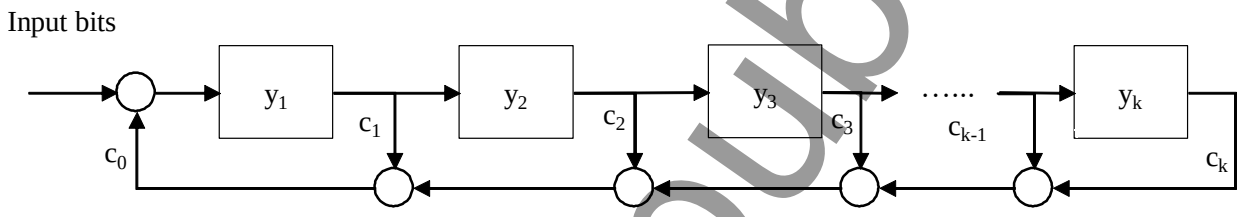


Figure 10-2 – External XOR-LFSR

The dual LFSR (external XOR-LFSR), is described by the same characteristic polynomial

$$f(x) = \sum_{i=o}^{k} c_i x^i$$

where $c_0 = 1$ e $c_k = 1$.

With this LFSR structure the intermediates bit-shift are not impacted by the feed-back, but they modify the inputs according to c_i values.

F.2.3 Aliasing general concepts

Compression techniques generate a kind of error called aliasing. Aliasing refers to the case where an error in data, under compression, maps into the same result without any error.

Suppose thar the inputs bits are affected by errors, introduced by noise or due to malfunctions. The aliasing phenomenon could cause the production of a final output equal to the expected value with correct inputs.

A wrong input bit can be expressed as the correct value xor-red with an error.

$b_{je} = b_j \oplus e_j$

Considering an incorrect input bit sequence, since the LFSR is a linear system (F.2.1), it is possible to apply the superposition effects; for the LFSR state at a cycle n we find the following expression in vectorial form:

$$Y_{ne} = Y_n \oplus Y_{ne}^*$$

dove

Y_n it is the output at cycle n without input errors.

Y_{ne}^* it is the output at cycle n related only to errors

According to this representation, the aliasing take place when:

$Y_{ne}^* = 0$

The AEP (Aliasing Error Probability) indicates the probability of occurrence depending on the single bit error probability.

Assuming independent errors on the single bits entering the register, the state later of the LFSR will depend statistically only on the present state and the input current. The behavior of the registry can therefore be described with a Markov chain.

The evaluation of Analytical expressions for AEP has been widely analyzed in literature (for example [DAOLFARI89] and [ELSHSE98]), the conclusion, reported also in [EN50159] §C.4, is

$$AEP = 2^{-k}$$

Where k is the number of bits in the LFSR, that is 32 for each PVS LFSR.

F.2.4 LFSR sequences – pseudorandom proprieties

Sequences generated via LFSR are widely analyzed on literature.

A summary of their properties (autocorrelation, intercorrelation, etc...) is described in [SARPUR80] and [WILSLO76]; in [DKNUTH69] they are examined as random bit generators.

To test the randomness of the sequences PR-SN and PR-EC, the test suite provided by NIST was used.

The suite is a battery of 15 tests, independent of each other, they analyze M sequences of N bits provided as input.

The result of each test on each sequence is represented by the "P-value", defined as the probability that a real random numbers generator can produce a sequence with poorer random properties than those identified in the sequence under analysis.

A "significance level" is chosen with which to compare the obtained P-value: if the P-value is greater than the "significance level" then the test is considered passed and the sequence is considered random.

Further details on the tests and on the methods of interpretation of the results are described in the documentation attached to the suite [NIST10].

F.2.5 LFSR sequences – pseudorandom test - results

As indicated in §7.5.3 (Req_SAI_012) for the evolution law of PR-SN and PR-EC two maximum length LFSR are used, they have the following characteristics polynomials:

$LFSR_1$: 10FC22F87 Hex $(x^{32}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{17}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^7+x^2+x+1)$

2247 $LFSR_2$: 1C3E887E1 Hex $(x^{32}+x^{31}+x^{30}+x^{25}+x^{24}+x^{23}+x^{22}+x^{21}+x^{19}+x^{15}+x^{10}+x^9+x^8+x^7+x^6+x^5+1)$

2248

2249 These polynomials are irreducible, primitive and reciprocal.

2250 Starting from the current value of the PR-SN or PR-EN, the new value is obtained by performing
2251 32 shifts on the LFSR loaded with the current value.

2252 For each of the two LFSRs, 1000 sequences formed by 10^6 bits each were produced, initializing
2253 the LFSRs with the following values

2254 $LFSR_1$: 0x3F4D897A

2255 $LFSR_2$: 0x54871ABD

2256 The LFSR evolution results are concatenated to be analyzed by the suite provided by NIST.

2257 The results, hereafter listed, have positive results for both LFSR; then the sequences generated
2258 by above LFSR can be classified as random.

F.2.6 LFSR sequences – pseudorandom test - reports

The test suite was executed, for each LFSR, on 1000 sequences of length 10^6 bits each one.

The results produced by the suite are collected in a report which is attached below

Results for $LFSR_1$

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
115	89	100	93	98	113	99	99	92	102	0.701366	0.9930	Frequency
99	105	114	91	109	97	105	79	115	86	0.171867	0.9900	BlockFrequency
115	96	88	119	85	102	109	95	88	103	0.194813	0.9900	CumulativeSums
117	112	87	88	91	85	99	103	105	113	0.183547	0.9940	CumulativeSums
95	103	94	110	105	100	82	113	93	105	0.572847	0.9870	Runs
86	111	108	95	91	101	124	110	87	87	0.090388	0.9920	LongestRun
119	107	116	99	88	81	100	104	103	83	0.094854	0.9890	FFT
104	120	89	90	111	108	88	89	90	111	0.159020	0.9900	NonOverlappingTemplate(1)
106	104	105	108	89	137	89	99	83	80	0.003684	0.9890	OverlappingTemplate
90	108	89	102	88	116	80	104	100	123	0.060112	0.9920	Universal
101	93	97	101	82	111	90	120	107	98	0.305599	0.9860	ApproximateEntropy
62	59	46	58	53	66	66	48	78	54	0.125989	0.9864	RandomExcursions(2)
56	61	60	56	58	56	66	60	58	59	0.997674	0.9915	RandomExcursionsVariant(3)
82	103	113	99	107	94	100	103	88	111	0.473064	0.9880	Serial
91	120	117	97	87	95	82	100	103	108	0.133404	0.9880	Serial
107	122	107	92	102	82	88	91	85	124	0.017912	0.9850	LinearComplexity

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 0.980561 for a sample size = 1000 binary sequences. The minimum pass rate for the random excursion (variant) test is approximately 0.977711 for a sample size = 590 binary sequences.

Results for $LFSR_2$

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
121	76	91	123	79	109	93	106	90	112	0.003224	0.9900	Frequency
85	98	101	104	112	110	99	89	107	95	0.651693	0.9910	BlockFrequency
112	95	87	112	98	99	114	84	96	103	0.380407	0.9900	CumulativeSums
102	107	97	83	107	112	101	102	99	90	0.689019	0.9890	CumulativeSums
94	101	76	120	107	107	101	94	108	92	0.173770	0.9890	Runs
97	101	96	81	111	105	114	91	97	107	0.467322	0.9890	LongestRun
106	113	88	88	92	104	100	98	107	104	0.697257	0.9830	FFT
92	106	91	102	95	96	107	105	93	113	0.818343	0.9890	NonOverlappingTemplate(1)
116	102	104	92	100	94	101	93	102	96	0.878618	0.9840	OverlappingTemplate
93	107	116	86	92	101	93	102	93	117	0.345650	0.9900	Universal
102	94	98	113	93	110	105	93	90	102	0.798139	0.9930	ApproximateEntropy
54	64	68	61	68	56	48	67	57	58	0.666591	0.9917	RandomExcursions(2)
67	56	65	59	58	60	52	60	69	55	0.881915	0.9867	RandomExcursionsVariant(3)
96	95	113	107	110	95	87	100	91	106	0.668321	0.9870	Serial
97	102	106	93	97	109	111	95	99	91	0.900569	0.9890	Serial
90	84	105	109	103	116	110	98	92	93	0.397688	0.9880	LinearComplexity

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 0.980561 for a sample size = 1000 binary sequences. The minimum pass rate for the random excursion (variant) test is approximately 0.977824 for a sample size = 601 binary sequences.

(1) only the result relating to the pattern 0000000001 is reported

(2) only the result of the first of the 8 tests of this type performed is reported

(3) only the result of the first of the 18 tests of this type performed is reported

Short description of the meaning of the columns in the test reports

- The first 10 columns (C1.. C10) show the frequency of P-values whose value are within the range relating to the n-th column (C1 range from 0 to 0.1, C2 range from 0.1 to 0.2 and so on up to C10 range from 0.9 to 1).

- The "P-VALUE" column is the P-value of the P-values obtained on the various sequences; it provides an evaluation of the uniformity of the distribution of P-values. Values greater than 10^{-4} indicate that the distribution can be considered uniform.
- The column "PROPORTION" reports the ratio between the number of sequences that have passed the test and the total number of sequences tested (1000);
- As written in the test report values higher than 0.980561 indicate that the used sequence can be considered as random (the threshold is different for the "Random excursion variant" test and is equal to 0.977711 for $LFSR_1$ and 0.977824 for $LFSR_2$).
- The "STATISTICAL TEST" column indicates the type of test performed.

F.3 Sequence number and time stamp represented by pseudo-random sequences

With PRS option, in addition to the "integer" representation of the Sequence Number (SN) and Execution Cycle (EC) the related representations are also transmitted, obtained through the use of LFSR (§7.5.3).

Taking into account that PR-SN and PR-EC are combined in XOR, the goal is to obtain uncorrelatedness between two successive values of the PR-SN and PR-EC sequences related to the same communication flow.

Greater uncorrelation between two successive values of the sequences, the lower will be the probability that an incorrectly PR-SN or PR-EC values can be mapped to another valid PR-SN or PR-EC following an error.

It has been demonstrated that using LFSR Maximum Length it is possible to obtain pseudo-random sequences (see §F.2.5 for further details): then the generated values are all equiprobable and therefore the probability of mapping, following an error, an incorrect value onto a valid value is equal to 2^{-k}, with k length of the LFSR.

Considering a receiver peer temporally aligned with the transmitter, the receiver is therefore able to check if the received packets are in sequence (Verification of Sequence Number) and fresh (Verification of Execution Cycle). The hypothetical hazard is that the transmitter fails in its task of generating the PR-SN or the PR-EC and, during the transmission and/or reception phase in the related peers, the incorrect value is compensated in corruption so that it can be considered valid by the receiver. Being PR-SN and PR-EC represented by two independent pseudo-random sequence generators each one 32 bit long, the probability of occurrence of the accidental generation of the correct pattern is equal to $2^{-64} = 5.4*10^{-20}$.

F.4 Length of safety code

The safety code used in PVS (SL_CRC field) is defined in §7.4.3.10 (requirement Req_SL_009)

The used polynomials are:

$LFSR_1$: 1100D4E63 Hex ($x^{32}+x^{28}+x^{19}+x^{18}+x^{16}+x^{14}+x^{11}+x^{10}+x^{9}+x^{6}+x^{5}+x+1$)

$LFSR_2$: 18CE56011 Hex ($x^{32}+x^{31}+x^{27}+x^{26}+x^{23}+x^{22}+x^{21}+x^{18}+x^{16}+x^{14}+x^{13}+x^{4}+1.$)

The chosen characteristic polynomials are primitives.

A polynomial of degree k is said to be primitive if:

- it is irreducible, i.e. cannot be factored
- 2^k-1 is his order

As already written in F.2.3 the AEP is 2^{-k}, that in case of PVS is equal to $2^{-64} = 5.4*10^{-20}$.

F.5 Safety demonstration – conclusions

The standard [EN50159] in §C.4 provides a model to compute, in specific cases, the target hazardous failure rate of the complete transmission system; at protocol definition most of parameters related to a complete transmission system are unknown, only the hazardous failure rate of EMI can be defined.

This parameter is defined as: p_{UT} x p_{US} x f_W

p_{US}: probability of undetected failure due to the performance of the safety code

p_{UT}: probability of undetected failure due to the performance of the transmission code, when the non-trusted transmission systems contain no transmission coding mechanisms then pUT = 1 has to be assumed;

f_W : frequency of wrong (corrupted) messages, to be conservative it is indicated as 1 (all messages corrupted)

Then, in the worst case scenario, hazardous failure rate of EMI is less than 10^{-18} (sum of F.1, F.3 and F.4); then negligible compared to the THR requested for a SIL4 application (10^{-9}) [EN50129].

Annex G
(informative)

Example of Application level

G.1 General description

In this annex an example of a possible PVS protocol Application level is described.

This layer, name INDICATIONS_CONTROLS, is in charge to convert the received Indication and Controls in the internal vital computer data representation.

At this level there is no difference between an Initiator and a Responder; there is only data received and data to send.

G.2 Static description

G.2.1 Indications

For the encoding of the indications, the following requirements must be fulfilled for the related user data (also called packets).

[Req_APP_001]

At INDICATIONS_CONTROLS level each packet containing the indications shall have two bytes at the beginning indicating:

- 1st byte: format version (structure and content) of transmitted data
- 2nd byte: configuration version (order of transmission of the indications).

Note: the configuration version has to be intended as a subfield of the format version.

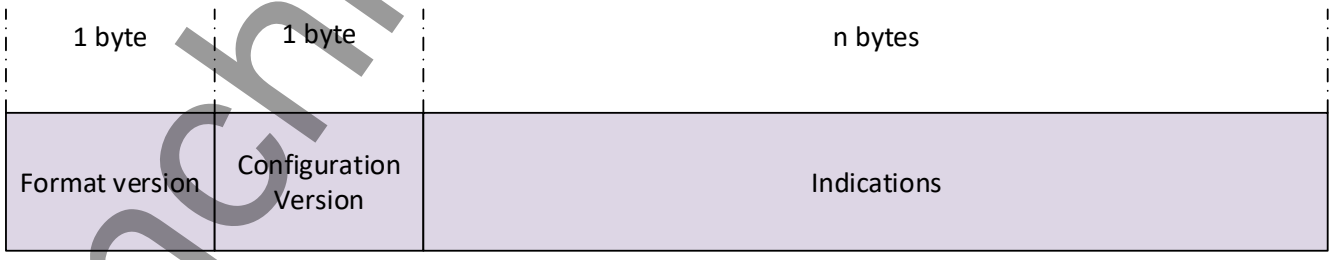


Figure G.10-1 – Packet containing Indications

[Req_APP_002]

Data format shall be in big endian format.

[Req_APP_003]

Each indication shall be encoded with a number of bits sufficient to represent, for each variable, all the possible values. For each indication, the size shall not be greater than 8 bits (1 byte). So for each indication, 256 different values can be transmitted at maximum.

[Req_APP_004]

In a single byte the indications shall be transmitted without unused bits, as represented in Figure 7-14.

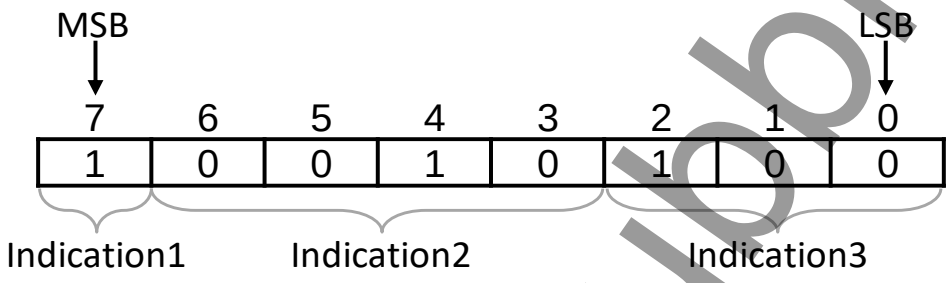


Figure G.10-2 - Encoding of indications

[Req_APP_005]

Each indication shall be completely included in one byte. For indications that bridge the addressing of two consecutive bytes, spare bits with value 0 shall be used for alignment.

Note: the number of spare bits must be optimized at configuration level (optimization of the allocation of the indications in different bytes).

G.2.2 Controls

For the encoding of the controls, the following requirements must be fulfilled for the related user data (also called packets).

[Req_APP_006]

For controls at INDICATIONS_CONTROLS level, each packet shall contain a unique control; if more than one control shall be sent in a cycle, more than one packet shall be transmitted.

At SAI_SL layer that means one frame for each control; if more controls in the same cycle: more frames with the same EC and SN incremented.

[Req_APP_007]

Each control shall be sent in a packet with data field of fixed length of 8 bytes: 4 bytes for the header and 4 bytes for the payload. The header shall contain the following fields:

– Subsystem Id (1 byte)
– Station Id (1 byte)
– Source (1 byte)
– Source attribute (1 byte)

The payload shall contain the following fields:

– Object Type (1 byte)
– Control Number (1 byte)
– Object Number (2 bytes)

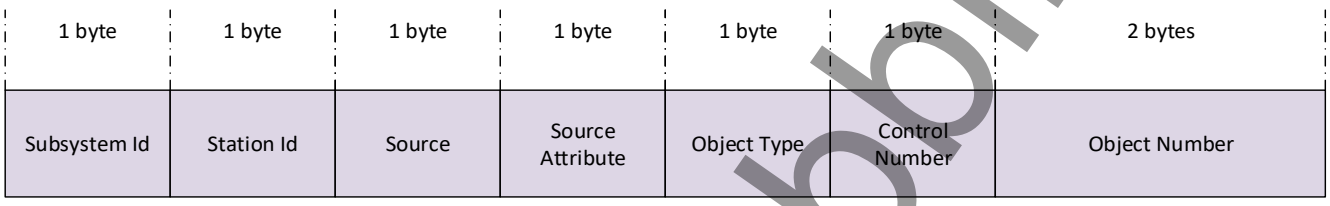


Figure G.10-3 - Packet containing Control

Note: an example of the header parameters meaning is the following:

- Subsystem Id: it identifies a SML400GP Project.
- Station Id: it identifies a station part of a multi-station Project (for example 0 the first one , 1 the second one etc.).
- Source: it identifies the source type of the control, for example the MMI or the TCS, an alternative possibility is to use this field to identify if the control is safety related or not;
- Source attribute: it identifies, within the type set by Source, the progressive number that identifies the physical device (0, 1, 2 etc).

G.3 Dynamic description

G.3.1 General

[Req_APP_008]

The INDICATION_CONTROLS layer shall be cyclic and activated at each Execution cycle.

[Req_APP_009]

A received data by the INDICATIONS_CONTROLS layer shall be declared as valid only if:

- at SAI_SL layer the connection is still established and
- its received Execution Cycle values are, in the current Execution Cycle, valid in agreement with the EC defence technique (see 8.3.3.5 and 8.3.3.6).

This requirement means that the Req_SAI_050 and Req_SAI_054 have to be applied to last received frame by the INDICATIONS_CONTROLS layer.

G.3.2 Indications

For the indications, the following requirements must be fulfilled for the related user data (also called packets).

[Req_APP_010]

At each Execution cycle, all the indications shall be transmitted and encapsulated in the same frame, then only one frame is provided to SAI_SL for transmission.

The frame provided to SAI_SL shall be complete, that means only one frame with all indications and the two initial bytes (see Figure G.10-1) shall be provided to SAI_SL; if INDICATION_CONTROLS layer is not able to create a complete frame, nothing shall be provided to SAI_SL layer.

Note:

- at SAI_SL level, in this case, is not possible to send frames with SN incremented, but with the same EC;
- in case of INDICATION_CONTROLS layer does not provide a frame to SAI_SL layer, it will send a NOP frame as indicated in the Req_SAI_043.

[Req_APP_011]

If the transmitter is faster than the receiver (i.e. TX cycle time is lower than RX cycle time), more than one packet is received in the same Execution cycle; only the last valid received message shall be provided to the Application process (ApplProcess).

Note that at SAI_SL level, in this case, are received more frames with EC and SN values incremented progressively.

[Req_APP_012]

If the transmitter is slower than the receiver (i.e. TX cycle time is greater than RX cycle time), the receiver may not receive an update of the indications before its cycle time expires.

In this case it shall send to the Application process (ApplProcess) the last received data still valid.

G.3.3 Controls

[Req_APP_013]

INDICATIONS_CONTROLS packets containing controls shall be transmitted when one or more controls shall be activated; one packet shall contain only one control.

Note:

- at SAI_SL level, in this case, is possible to send frames with SN incremented, but with the same EC;
- when no control is activated nothing is provided to SAI_SL layer then it will send a NOP frame as indicated in the Req_SAI_040.

[Req_APP_014]

All the valid controls received shall be provided to the Application Process (ApplProcess).

The number of controls that can be managed in the same Execution Cycle is implementation dependent

G.3.4 Application conditions and configuration parameters

[Req_APP_AC_001]

For each instance, a unique typology of messages (indications or control) shall be used for each direction of communication.

For example, if A transmits to B both indications and controls, two sessions shall be opened; vice versa if A sends controls and B answers with indications, only one session may be used.

[Req_APP_AC_002]

Table G.1 contains a list of all protocol parameters (Parameter and Value columns) that shall be defined in the system configuration phase and shall be parameters for INDICATION_CONTROLS implementation.

In a vital computer, for each defined instance a list of these parameters shall be available.

Note: if 2 or more PVS instances are defined within the same vital computer, an equivalent number of the following parameters list will be defined.

Parameter	Description	Refer to	Value
TxFmtVersion	It is the format version of transmitted data in case of Indications.	See Figure G.10-1.	Minimum value 0 (8 bits value).
TxCfgVersion	It is the configuration version of transmitted data in case of indications.	See Figure G.10-1.	Minimum value 0 (8 bits value).
RxFmtVersion	It is the format version of received data in case of Indications.	See Figure G.10-1.	Minimum value 0 (8 bits value).
RxCfgVersion	It is the configuration version of received data in case of indications.	See Figure G.10-1.	Minimum value 0 (8 bits value).
Subsystem Id	Used only in case of controls: it identifies the subsystem that generates the control	See Figure G.10-3.	It is an unsigned value of 1 byte
Station Id	**Used only in case of controls: it identifies the station that generates the control**	**See Figure G.10-3.**	**It is an unsigned value of 1 byte**
Source	**Used only in case of controls: it identifies the Source that generates the control**	**See Figure G.10-3.**	**It is an unsigned value of 1 byte**
Source attribute	**Used only in case of controls: it identifies the Source attribute that generates the control**	**See Figure G.10-3.**	**It is an unsigned value of 1 byte**
Object Type	**Used only in case of controls, a parameter for each Control: it identifies the Object type**	**See Figure G.10-3.**	**It is an unsigned value of 1 byte**
Control Number	**Used only in case of controls, a parameter for each Control: it identifies the number of Control**	**See Figure G.10-3.**	**It is an unsigned value of 1 byte**
Object Number	**Used only in case of controls, a parameter for each Control: it identifies the Object number**	**See Figure G.10-3.**	**It is an unsigned value of 2 bytes**

2559 **Table G.1 – INDICATION_CONTROLS Configurable parameters list**

2560

2561

2562

Bibliography

Ref.	Title	Issue	Date
Ref.1	Railway Applications – Communication, signalling and processing systems. Safety-related communication in transmission systems	CENELEC EN 50159	September 2010
Ref.2	ERTMS/ETCS – Euroradio FIS	SUBSET-037	3.2.0
Ref.3	ERTMS/ETCS – RBC-RBC Safe Communication Interface	SUBSET-098	3.0.0
Ref.4	Specification for the Advanced Encryption Standard (AES)	FIPS Publication 197	Nov 26, 2001
Ref.5	PROTOCOLLO VITALE STANDARD	RFI DTC DNS SS RT IS05 021 F	12/06/2017
Ref.6	User Datagram Protocol (UDP)	RFC768	-
Ref.7	Transmission Control Protocol (TCP): Sept 81- TCP Extensions for High Performance	RFC793, RFC1323	-
Ref.8	Tabella Comandi/Controlli Verticali PVS (PVS PFV) PP/ACC e PP/ACEI" RFI-DIN.DIPT.PCA0011P20160000618_1;	RFI-DIN.DIPT.PCA0011P20160000618_1	-
Ref.9	Protocollo Vitale Standard RFI – Analisi dei Rischi	RFI DTC DNS SS RT IS08 028 A	22/02/0212
Ref.10	The AES-CMAC Algorithm - JH.Song et al.	RFC 4493	June 2006

2563
2564

2565

Comitato Tecnico Elaboratore
CT 9-Sistemi e componeneti elettrici ed elettronici per trazione
Altre norme di possibile interesse sull'argomento

Progetto
C. 1336:2024-01 – SCAD. 18-03-2024
Totale Pagine 116

Via Saccardo, 9
20134 Milano
Tel. 02.21006.1
Fax 02.21006.210
cei@ceinorme.it
www.ceinorme.it