



SURFACE VEHICLE RECOMMENDED PRACTICE	J1939™-91C	XXXX
	Issued	
CAN FD Network Security		

RATIONALE

This document facilitates the secure use of SAE J1939 CAN networks with flexible data rate (CAN FD) data frame format for communication use cases. This recommended practice provides methods for establishing trust and securing messages on CAN FD networks with optional encryption.

FOREWORD

The SAE J1939 communications network is developed for use in heavy-duty environments and suitable for the integration of different combinations of loose package components, such as engines and transmissions, that are sourced from many different component suppliers. The SAE J1939 common communication architecture strives to offer an open interconnect system that allows the ECUs associated with different component manufacturers to communicate with each other.

SAE J1939-91A and SAE J1939-91C provide cybersecurity recommendations for secure architectures, secure vehicle to infrastructure communications and secure on-board communications. The SAE J1939-91A document focuses on security measures related to on-vehicle network architecture and security measures for communication interfaces between devices or networks.

This particular document, SAE J1939-91C, describes the cryptographically secure transfer of messages between ECUs with optional encryption. It describes the techniques required to establish trust through key management and extend it throughout the vehicle's life cycle.

SAE Executive Standards Committee Rules provide that: "This report is published by SAE to advance the state of technical and engineering sciences. The use of this report is entirely voluntary, and its applicability and suitability for any particular use, including any patent infringement arising therefrom, is the sole responsibility of the user."

SAE reviews each technical report at least every five years at which time it may be revised, reaffirmed, stabilized, or cancelled. SAE invites your written comments and suggestions.

Copyright © 2021 SAE International

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SAE.

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
 Tel: +1 724-776-4970 (outside USA)
 Fax: 724-776-0790
 Email: CustomerService@sae.org
 http://www.sae.org

SAE WEB ADDRESS:

For more information on this standard, visit
<https://www.sae.org/standards/content/PRODCODE/>

TABLE OF CONTENTS

1.	Scope	7
1.1	Audience	7
1.2	Goals	7
1.3	Adversary	8
1.4	Out-of-Scope	9
2.	References	9
2.1	SAE Documents	9
2.2	NIST Documents	9
2.3	Request for Comments (RFC)	9
2.4	ISO Documents	10
2.5	International Telecommunication Union Telecommunication (ITU-T) Documents	10
3.	Definitions	10
3.1	AUTHENTICITY	10
3.2	AUTHORIZED ECU	10
3.3	CERTIFICATE / CERTIFICATE CHAIN TERMS	10
3.4	CIPHER-BASED MESSAGE AUTHENTICATION CODE (CMAC)	10
3.5	CONFIDENTIAL COMMUNICATION MESSAGE	10
3.6	CONTROLLER APPLICATION (App)	10
3.7	CRYPTOGRAPHICALLY STRONG	11
3.8	DATA BYTES	11
3.9	E (ENCRYPT BIT)	11
3.10	ENTROPY	11
3.11	EPOCH	11
3.12	FOLLOWER	11
3.13	FORWARD SECRECY	11
3.14	FRESH	11
3.15	FRESHNESS VALUE (FV)	11
3.16	INTEGRATED MEMBER	11
3.17	INTEGRITY	11
3.18	KEY DERIVATION FUNCTION (KDF)	11
3.19	MEMBER	12
3.20	MESSAGE	12
3.21	MUTUAL AUTHENTICATION	12
3.22	NETWORK AUTHORITY (NA)	12
3.23	NETWORK IDENTIFIER (NID)	12
3.24	NETWORK KEY	12
3.25	NETWORK LEADER	12
3.26	NID CERTIFICATE	12
3.27	NONCE	12
3.28	NON-MEMBER	13
3.29	PACKET	13
3.30	PARAMETER GROUP NUMBER (PGN)	13
3.31	PSEUDORANDOM NUMBER GENERATOR (PRNG)	13
3.32	RECEIVING MEMBER	13
3.33	REKEY	13
3.34	SECURE MESSAGE	13
3.35	SECURE NETWORK	13
3.36	SESSION	13
3.37	SESSION KEY	13
3.38	SESSION KEY FOR CMAC	13
3.39	SESSION KEY FOR ENCRYPTION	13
3.40	SOURCE ADDRESS (SA)	13
3.41	STALE FV	14
3.42	Tag	14
3.43	Tag_Calc	14
3.44	Tag_CMAC	14
3.45	TRANSFERABLE MEMBER	14
3.46	TRANSMITTING MEMBER	14
3.47	TRUE RANDOM NUMBER GENERATOR (TRNG)	14

3.48	WRAPPED KEY	14
4.	Abbreviations	14
5.	Conventions, Constants and TimeOuts	15
5.1	Requirements	15
5.2	Conventions	15
5.2.1	Bit-wise Shift Left Operator (<<)	15
5.2.2	Concatenation ()	15
5.2.3	RQST(<i>value</i>)	15
5.3	Constants	15
5.3.1	Const _{SecOC-ENC}	15
5.3.2	Const _{SecOC-CMAC}	15
5.3.3	FV_INIT	16
5.3.4	FV_REKEY_TRIGGER	16
5.3.5	SLIDING_WINDOW_SIZE	16
5.3.6	MAX_AUTH_FAILURE	16
5.3.7	MAX_NF_FAIL_COUNT	16
5.3.8	MAX_MS	16
5.4	Timeout Defaults	16
6.	Secure Messaging Overview	16
6.1	Secure Messages	17
6.2	Vehicle Production	18
6.2.1	Network Formation	19
6.3	Network Operation	19
6.3.1	Establish Session Key – Rekeying	20
6.3.2	Exchanging Secure Messages	20
6.4	Vehicle Maintenance	20
7.	Cryptography Primitives	21
7.1	Network Formation	21
7.2	Rekey and Exchange Messages	21
7.3	Network Member Security Level	22
7.3.1	Secure Key Storage	22
8.	Freshness Value (FV)	22
8.1	FV Concepts	22
8.1.1	FV at Transmitter	22
8.1.2	FV at Receiver	23
8.1.3	FV Size	23
8.2	FV Design Considerations	23
8.3	S _s Keys and FV	23
8.4	Sliding Window	24
8.4.1	Checking Freshness	25
8.4.2	Updating the window	25
8.5	Resetting FV	26
8.6	Counter-based approach to SecOC/E error state	26
8.7	FV Requirements	26
9.	Key Management	28
9.1	When and Where Used	28
9.2	Types of Keys	29
9.3	X.509 NID Certificate	29
9.3.1	NID and SCM-VIN	30
9.4	Crossing Trust Boundaries	30
9.5	Key Life Cycle Management	30
9.6	Network Authority	31
10.	Network Operation	32
10.1	Simplified State Machine	32
10.1.1	Rekeying (R)	32
10.1.2	Exchange Message (E)	32
10.1.3	Network Formation (N)	33
10.1.4	Simplified Transitions	33
10.2	Cryptography for Secure Messages	34
10.3	Detailed State Machine	36
10.3.1	Detailed Transitions	36

10.3.2	Failure Transitions	37
10.3.3	State Prioritization	37
10.4	Sequence Diagram	37
10.5	Network Operation Requirements	38
11.	Rekeying	39
11.1	Simplified Process	39
11.1.1	Generate nonce	39
11.1.2	Send RQST(Rekey)	39
11.1.3	Send Rekey Message	40
11.1.4	Rekey Timer (T_R) Processes	40
11.1.5	Calculate S_S	40
11.1.6	Announce Network	40
11.2	Rekey Concepts	41
11.2.1	Freshness	41
11.2.2	Adding/removing a node	41
11.2.3	Transition between old and new S_S	41
11.3	Timing and Control	41
11.4	Detailed Process Flow	41
11.5	Missing Leader Exception Handling	43
11.6	Rekeying Requirements	43
12.	Network Formation	44
12.1	Simplified Process	45
12.1.1	Mutual Authentication	45
12.1.2	Derive Session Keys	46
12.1.3	Share Encrypted Key (S_N)	46
12.2	Rekey and Network Formation Integration	46
12.3	Detailed Process Flow	48
12.3.1	Transition Flags	49
12.3.2	Generation of Network Key (S_N)	49
12.4	NF Requirements	50
13.	Exchanging Messages	52
13.1	Structure of Nonce	52
13.2	Transmitting	52
13.3	Receiving	53
13.4	Unsecured Message	56
13.5	Exchanging Message Requirements	56
14.	Message Definitions	57
14.1	SecOC/E Announce Leader PG (PGN 64002)	57
14.1.1	SecOC/E Announce Leader Channel (SPN 23123)	58
14.1.2	SecOC/E Leader Protocol Version (SPN 23124)	58
14.1.3	SecOC/E Leader Nonce (SPN 23125)	58
14.1.4	SecOC/E Leader Certificate (SPN 23126)	58
14.2	SecOC/E Join Network PG (PGN 64003)	58
14.2.1	SecOC/E Join Network Channel (SPN 23127)	59
14.2.2	SecOC/E Join Network Protocol Version (SPN 23128)	59
14.2.3	SecOC/E Follower Nonce (SPN 23129)	59
14.2.4	SecOC/E Follower Certificate (SPN 23130)	60
14.3	SecOC/E Send Network Key PG (PGN 18176)	60
14.3.1	SecOC/E Send Network Key Channel (SPN 23131)	60
14.3.2	SecOC/E Send Network Key Protocol Version (SPN 23132)	61
14.3.3	SecOC/E Encrypted S_N (SPN 23133)	61
14.3.4	SecOC/E Encrypted S_N CMAC (SPN 23134)	61
14.4	SecOC/E Rekey Nonce PG (PGN 64004)	62
14.4.1	SecOC/E Rekey Channel (SPN 23135)	62
14.4.2	SecOC/E Rekey Protocol Version (SPN 23136)	62
14.4.3	SecOC/E Rekey Nonce (SPN 23137)	63
14.4.4	SecOC/E Member NID CMAC (SPN 23138)	63
14.5	SecOC/E Announce Network Message (PGN 64005)	63
14.5.1	SecOC/E Announce Network Channel (SPN 23139)	63
14.5.2	SecOC/E Announce Network Protocol Version (SPN 23140)	64
14.5.3	SecOC/E Leader NID CMAC (SPN 23141)	64

15.	Protocol Limitations.....	64
15.1	Classic CAN.....	64
15.2	Functional Safety	64
15.3	Multiple Network Leaders	64
APPENDIX A - Rekeying Timing Diagrams.....		65
APPENDIX B – Technical Rationales.....		70
APPENDIX C – Temporary Network Key		72
APPENDIX D - Test Vectors.....		73
APPENDIX F - Secure Diagnostics Network.....		91
APPENDIX G – SecOC/e Example Message.....		93
Figure 1 – SAE J1939-91C Scope.....		7
Figure 2 – Message Types and Security Overhead		17
Figure 3 – Message Authentication Process		18
Figure 4 – Vehicle Production Trust Diagram.....		19
Figure 5 – Vehicle Operation Trust Diagram.....		20
Figure 6 – Vehicle Maintenance Trust Diagram		21
Figure 7 - Example of S _s FV Sliding Window Data Structure.....		23
Figure 8 – Sliding Window Code Example		24
Figure 9 – Checking Freshness Code Example		25
Figure 10 – Updating the Window Code Example.....		26
Figure 11 – Relationship between Keys, Parameters and SecOC/E Phases		28
Figure 12 – NIST 800-57 Figure 5 Key-Management States and Phases.....		31
Figure 13 – Simplified State Machine.....		32
Figure 14 – Creating an Authentic Message		34
Figure 15 – Creating a Confidential Message		35
Figure 16 – Detailed State Machine		36
Figure 17 – Rekey Sequence and Timers		38
Figure 18 – Simplified Rekey Process.....		39
Figure 19 – Rekey Message Processing.....		42
Figure 20 – Detailed Rekey Process Flow.....		43
Figure 21 – Simplified Network Formation Process		45
Figure 22 – Service Part Network Formation.....		47
Figure 23 – Network Formation Sequence		49
Figure 24 – Nonce Construction		52
Figure 25 – Transmit Messages Process Flow		53
Figure 26 – Receive Messages Process Flow		54
Figure 27 – Drop Message Process Flow.....		55
Figure 28 – SecOC/E Announce Leader Message		57
Figure 29 – SecOC/E Join Network Message		59
Figure 30 – SecOC/E Send Network Key Message.....		60
Figure 31 – SecOC/E Rekey Nonce Message		62
Figure 32 – SecOC/E Announce Network Message.....		63
Figure A1 - Rekey Simplest Case.....		65
Figure A2 - Rekey Power On Best Case		66
Figure A3 - Rekey with Power On Race Conditions.....		67
Figure A4 - Rekey Power On with Late Member		68
Figure A5 - Power On missed Rekey		69
Figure D1 - Cryptographic Operations Test Vectors.....		77
Figure E1 – SDL Flow Chart Symbol Descriptions.....		78
Figure E2 – SDL Chart Symbols, continued.....		79
Figure E3 – Timer Verb Definitions.....		80
Figure E4 – States for an ECU participating in SecOC/E		80
Figure E5 – Rekey: Address Claim and Start Processes		81
Figure E6 – Rekey: Init and Message Transmit Processes		82
Figure E7 – Leader: Rekey and Announce Network Process		83
Figure E8 – Follower: Rekey and Announce Network Processes.....		84
Figure E9 – Rekey Timeout Process		85
Figure E10 – Receiving Secure Message Process		86
Figure E11 - Send Secure Message and Timeout Processes		87
Figure E12 – Leader: Network Formation and Timeout Process		88

Figure E13 – Follower: Network Formation Process	89
Figure E14 –Follower: Receive Network Key and Timeout Processes	90
Figure F1 - Secure Diagnostics Network Concept	91
Figure G2 – Example J1939-22 message with SecOC/E Assurance Data.....	93
Table 1 - Timeout Definitions	16
Table 2 – Network Formation Crypto Primitives	21
Table 3 – Rekey and Exchanging Messages Crypto Primitives	22
Table 4 – Key Summary	29
Table 5 – X.509 v3 Digital Certificate	29
Table 6 – Authentic Message Components.....	34
Table 7 – Additional Confidential Message Components	35
Table 8 – Rekey Controls	41
Table 9 – Rekey Timers.....	41
Table 10 – Transmitter Tag Calculations.....	53
Table 11 – Transmitter Ciphertext Calculations	53
Table 12 – Receiver Tag Calc Calculations.....	55
Table 13 – Receiver CT Decryption.....	56
Table B1 - Epoch Durations for varing FV Size and Message Rates.....	70
Table B2 - Tag Length Relationships	71

1. SCOPE

SAE J1939-91C describes Secure On-board Communications with optional Encryption (SecOC/E) for networks compliant with SAE J1939-22 (CAN FD) and the processes and infrastructure required to make it secure. A simplified network lifecycle with the critical SAE J1939-91C components in green is shown in Figure 1. Authorized ECUs authenticate themselves within the vehicle by means of digital certificates in a public key infrastructure and subsequently obtain a shared secret used to secure relevant SAE J1939 messages. These processes and techniques can be used to extend the security to multiple segments and non-SAE J1939 networks as required.

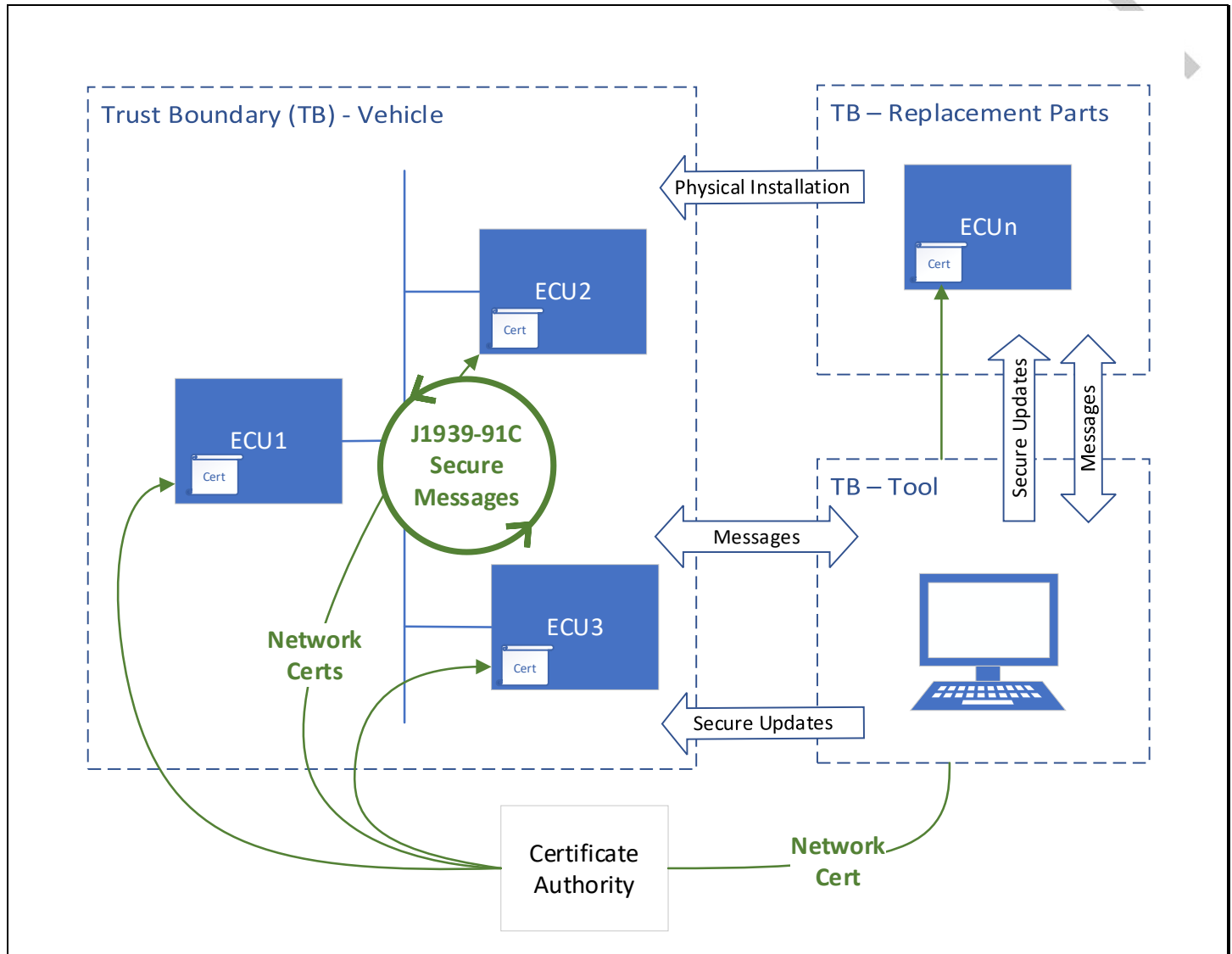


Figure 1 – SAE J1939-91C Scope

1.1 Audience

This document is suitable for two audiences – those looking for a brief overview as well as those seeking the details required to create an implementation. The overview can be found in sections 1 to 6 with the details in section 7 through 13.

1.2 Goals

There are no fleet wide secrets within an electronic control unit (ECU). Original Equipment Manufacturers (OEM) and Network Authorities (NA) can communicate securely with an ECU without having a secret key specifically for that ECU. The NA administrator will process the secure transfer of root certificate's private key; the process is not defined by this document.

Each network creates its own secret network key. Except for the Network Leader during Network Formation, an ECU never shares its secret keys with any other ECU, and ideally, not even with the application on the ECU. The application should only be able to use keys held within the Hardware Protected Security Environment (HPSE). It should not be able to extract them in plaintext form.

The following list is provided to set a common understanding of the goals of SecOC/E.

SecOC/E lifetime:

- cryptographic primitives should be selected to be strong enough to still be good five years from release of the latest version, though the extent of change can be constrained to the capability of cryptographic hardware accelerators available in the ECUs.
- significant progress in quantum computing can render the asymmetric cryptographic primitives insecure, resulting in the need for a completely new approach to SecOC/E.

SecOC/E provides:

- plaintext, authentic communication (SecOC) between nodes on the network.
- ciphertext, authentic communication (SecOC/E) between nodes on the network.
- SecOC/E gracefully handles defined windows where transmitters are not strictly FIFO due to message priority and bus arbitration

SecOC/E prevents:

- replaying messages, reordering/delaying messages outside a defined window
- an actor who has not extracted an ECU's secrets from changing the behavior of the machine by changing packets on the bus.
- an actor who has extracted an ECU's secrets from using these secrets to attack other machines. There is nothing in SecOC/E keys that can be leveraged into an attack on a fleet of machines since certificates are required as part of the association of a controller (or its stolen keys) with a vehicle and/or network.
- an actor from logging and getting meaning from encrypted messages.

SecOC/E does not prevent:

- an actor from logging and getting meaning from plain text authentic only messages.
- a malicious denial-of-service attack. It cannot prevent an adversary with physical control of the network from taking down the network.
- an actor who has extracted an ECU's secret from changing the behavior of the machine by changing packets on the bus.

SecOC/E requires:

- networks with frames that support messages of length 16 bytes (or longer).

1.3 Adversary

The following lists are provided to set a common understanding of the nature of the adversary's capabilities that SecOC/E is intended to protect against.

Adversary capabilities:

- Can read every packet on the network.
- Can be a Meddler-in-the-middle, anywhere in the network, which allows:
 - dropping packets selectively or en masse
 - re-ordering packets
 - modifying packets "in-flight"
- Can inject packets into the network, which allows:
 - replaying insecure, authentic or authentic and confidential packets, original or modified, from this or other machines
 - creating packets
 - packets can be made to appear with any source address
 - without limit on number of packets or rate of packet injection
- Access to ECUs from this network (or service parts).
 - can extract and use the ECU's secrets
- Remote access to ECUs (e.g., telematics).
- Can load malicious apps on any ECU that provides a mechanism to download and run code that originates outside the target environment.

- Can control an unbounded number of inauthentic nodes.
- Can join and leave the network at any time.

Adversary limitations:

- Cannot forge a Network Identifier (NID) certificate.
- Cannot break the cryptographic algorithms (this is not to be construed to mean they cannot find ways to go around or defeat the security intended to be provided by the crypto).

1.4 Out-of-Scope

The following is a list of capabilities that are not in the scope of SecOC/E.

- Reliable delivery of packets (lost secure packets are not retransmitted).
- In-order delivery of packets (SecOC/E allows re-ordering in a defined window).
- The ability to turn secure communication off. This is a policy decision and not one handled in the protocol.
- SecOC/E does not establish or require a specific network architecture design.
- SecOC/E refers policy decisions regarding the use of firewalls or gateways to SAE J1939-91A.
- SecOC/E does not describe techniques for functional safety.
- Fault reporting methods are defined by J1939-73.
- Forward secrecy is not provided with SecOC/E; this is relevant for messages intended to be kept confidential beyond the current power-cycle. The derivation of past session keys requires the network key and a log of all the rekey messages during that session.

2. REFERENCES

2.1 SAE Documents

SAE J1939021	Data Link Layer
SAE J1939-22	CAN FD Data Link Layer
SAE J1939-73	Application Layer - Diagnostics
SAE J1939-81	Network Layer
SAE J3101	Hardware Protected Security for Ground Vehicles
SAE J3138	Diagnostic Link Connector Security
SAE/ISO 21434: 2021	Road vehicles — Cybersecurity engineering

2.2 NIST Documents

SP 800-38B	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication
SP 800-56C Rev. 2	Recommendation for Key-Derivation Methods in Key-Establishment Schemes, August 2020
SP 800-57	Recommendation for Key Management

2.3 Request for Comments (RFC)

These RFCs can be useful for a deeper understanding of the cryptographic primitives.

RFC 2119	Key words for use in RFCs to Indicate Requirement Levels, MARCH 1997
RFC 3686	Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP), JANUARY 2004
RFC 4493	The AES-CMAC Algorithm, JUNE 2006

RFC 5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, MAY 2008
RFC 6234	US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF), MAY 2011
RFC 7748	Elliptic Curves for Security, JANUARY 2016
RFC 8410	Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure, AUGUST 2018
RFC 8418	Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS), AUGUST 2018

2.4 ISO Documents

ISO 20828	Road vehicles — Security certificate management
-----------	---

2.5 International Telecommunication Union Telecommunication (ITU-T) Documents

ITU-T X.509	Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks
-------------	--

3. DEFINITIONS

This section establishes terms used throughout this document:

3.1 AUTHENTICITY

The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator.

3.2 AUTHORIZED ECU

A controller on the network that is intended by the NA to participate in secure communications. For integrated member ECUs, the “intention” is provided through a NID certificate, which cryptographically ties the ECU to the network’s NID.

3.3 CERTIFICATE / CERTIFICATE CHAIN TERMS

Related to X.509 and ECU authenticity. An ECU is issued a certificate, and that certificate has usefulness in two ways: values of specific fields in the certificate and that the certificate chains back to the NA.

3.4 CIPHER-BASED MESSAGE AUTHENTICATION CODE (CMAC)

The CMAC value protects both a message’s data integrity as well as its authenticity, by allowing verifiers (who also possess the secret key) to detect any changes to the message content. Often expressed as: Result = CMAC(key, target).

3.5 CONFIDENTIAL COMMUNICATION MESSAGE

A message that has all characteristics found in authentic communication, with the added characteristic that the data in the messages has been encrypted and can be decrypted only by members.

3.6 CONTROLLER APPLICATION (App)

A physical ECU may have: a single controller application (App), multiple Apps that share resources, or multiple Apps that are completely independent of each other. Which Apps require unique or shared resources are system design options and are beyond the scope of SAE J1939-91C.

3.7 CRYPTOGRAPHICALLY STRONG

The only known way to tamper with the data is to provide a valid tag, and without the keys there is no way other than luck (1 in 2^{31} chance). A brute-force search through the 2^{31} space (e.g., repeatedly sending messages to a member, trying to use it as an oracle) isn't feasible since each message requires a new freshness value, and that freshness value is part of the tag calculation.

3.8 DATA BYTES

The command-and-control message content that is being protected by SecOC/E.

3.9 E (ENCRYPT BIT)

One-bit value that indicates to the receiver whether or not the message is encrypted.

3.10 ENTROPY

(Information theory) The average level of "information", "surprise", or "uncertainty" inherent in the variable's possible outcomes. For example, in SecOC/E there are members contributing uncertainty in the creation of S_s .

3.11 EPOCH

The period defined from freshness value initialization to FV_REKEY_TRIGGER. See Session (3.37).

3.12 FOLLOWER

All members except the Network Leader.

3.13 FORWARD SECRECY

A feature of specific key agreement protocols that gives assurances that session keys are not compromised even if long-term secrets used in the session key exchange are compromised.

3.14 FRESH

Secret keying material is considered "Fresh" when newly established secret keying material is statistically independent of any previously established keying material.

3.15 FRESHNESS VALUE (FV)

A 32-bit value used in calculating a CMAC that prevents message replays.

3.16 INTEGRATED MEMBER

A subset of the members; where the subset includes just those controllers that are, by design, only removed from the network when they fail. This is in contrast to controllers that are intended to be transferred (see "Transferable Member").

3.17 INTEGRITY

A property whereby data is not altered in an unauthorized manner after it is created, transmitted, or stored.

3.18 KEY DERIVATION FUNCTION (KDF)

A cryptographic function that derives one or more secret keys from a secret value using a pseudorandom function.

3.19 MEMBER

A controller on the network that fulfills the “authorized ECU” definition, as demonstrated by successful completion of mutual authentication. The set of member ECUs is the union of the set of integrated member ECUs and the set of transferable member ECUs. The set of member ECUs is also the union of the set of Follower(s) and the Network Leader. Informally:

members = integrated members + transferable members

members = Network Leader + Follower(s)

3.20 MESSAGE

The bytes in a network frame after the header. Constitutes the data that is being sent from a transmitter to one or more receivers. In a secure message it includes the security overhead (see SAE J1939-22 for more details).

3.21 MUTUAL AUTHENTICATION

The process of a Follower presenting its NID certificate to the Network Leader and the Network Leader presenting its NID certificate to the Follower. In order for the Network Leader to share S_N , both the Network Leader and Follower must have valid certificates and the NID in the certificates must be the same.

3.22 NAME_TIMEOUT

The SAE J1939 Name Management address claim duration. This timeout value and its constraints specified in SAE J1939-21 and SAE J1939-22. (See J1939-91C Appendix A for defined time).

3.23 NETWORK AUTHORITY (NA)

This entity is responsible for authenticating devices for use in SecOC/E networks. The OEM, or a third-party Certificate Authority, initiates this role but fleet operators or maintainers can do it.

3.24 NETWORK IDENTIFIER (NID)

An alpha numeric string, which may or may not be the Vehicle Identification Number (VIN), that identifies a SecOC/E network. NID covers use cases for non-vehicle applications and multiple secure networks on a vehicle. Each instance of a network has a unique NID.

3.25 NETWORK KEY

A long-lived symmetric network secret (S_N), shared amongst only members. It is the basis of all future session keys, S_s , used during the life of the machine.

3.26 NETWORK LEADER

The single member that is tasked with driving the processes related to the creation of the secure network. Typically, this is an integrated member and referred to more simply as “Network Leader”.

3.27 NID CERTIFICATE

A X.509 certificate chain that provides evidence that the ECU is authorized to be on the network. ECUs without a valid NID certificate cannot obtain the network key and hence cannot participate in the secure network.

3.28 NONCE

“Number used once”. In this document, there are many points where nonces are required, e.g., to prevent replay attacks, or to act as an initialization vector.

3.29 NON-MEMBER

A controller on the network that fails the “member” definition. These controllers are not able to participate in secure communication.

3.30 PACKET

Consists of a header, message data (the payload) and an optional trailer. Also known as network frame (see SAE J1939-22 for more details).

3.31 PARAMETER GROUP NUMBER (PGN)

A 24-bit structure as defined in SAE J1939-22. Used in calculating secure message CMAC.

3.32 PSEUDORANDOM NUMBER GENERATOR (PRNG)

An algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.

3.33 RECEIVING MEMBER

A member that is the consumer of a secure message. A receiving member tracks the FV of every transmitting member in which it receives messages.

3.34 REKEY

The process of generating a new session key, S_S .

3.35 SECURE MESSAGE

A message that has cryptographically strong evidence that it has Integrity, Authenticity and is Fresh.

3.36 SECURE NETWORK

A logical (as opposed to physical) construct that represents a communication link that offers authentic (and sometimes also confidential) messaging between members on the network.

3.37 SESSION

The use of S_S from its initial calculation until the transition to a new S_S has completed. See Epoch (3.11).

3.38 SESSION KEY

Symmetric session key (S_S); it is an AES-128 key. Note that S_S is a convenient shorthand for the fully qualified keys S_{S-CMAC} and S_{S-ENC} .

3.39 SESSION KEY FOR CMAC

Symmetric AES-128 session key used to generate the CMAC (S_{S-CMAC}) for each SecOC/E message.

3.40 SESSION KEY FOR ENCRYPTION

Symmetric AES-128 session key used to encrypt data (S_{S-ENC}) that is to be kept confidential.

3.41 SOURCE ADDRESS (SA)

Network address used by a controller application, per SAE J1939-22. Used in calculating secure message CMAC.

3.42 SS_n (or S_s)

A convenient shorthand for the fully qualified keys SS-CMAC and SS-ENC. A rekey sequence from SS0 to SS1 where both SS values are valid for TSS. The corresponding states are listed on the left with the valid SS_n as a subscript.

3.43 STALE FV

A freshness value, FV, that is either: too old (the value falls outside the FV window) or has already been seen (a replay).

3.44 Tag

The most significant 31 bits of Tag_CMAC. This is sent with a secure message as described in 6.1.

3.45 Tag_Calc

The most significant 31 Tag_CMAC bits that are calculated from a secure message to check the message.

3.46 Tag_CMAC

The full 128-bit CMAC.

3.47 TRANSFERABLE MEMBER

A subset of members that are designed to be transferred between networks in the normal course of use, for example engineering tools.

3.48 TRANSMITTING MEMBER

A member that is the source of a secure message. A transmitting member owns its FV and increments FV each time it creates a Tag.

3.49 TRUE RANDOM NUMBER GENERATOR (TRNG)

A device that generates random numbers from a physical process rather than from an algorithm. Typically, the physical process is a statistically random noise signal. If harvesting entropy is expensive in terms of time, then it is acceptable for the TRNG to be used as a seed to a pseudo-random number generator.

3.50 WRAPPED KEY

A combination of an encrypted key accompanied by a CMAC of the encrypted key. For example, an HPSE can create this combination when a key, such as S_N, needs to be stored securely outside of the HPSE. The keys used for encrypting and CMAC'ing are different, and are expected to be created in the HPSE when the chip was manufactured. The encryption and wrapping keys shall be at least as strong as AES-128.

4. ABBREVIATIONS

Abbreviations that are not defined in section 3 are expanded in this section.

AES-128	128-bit Advanced Encryption Standard
CA	Certificate Authority
CT	Cipher Text
CTR	Counter Mode for AES-128
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
HKDF	HMAC Key Derivation Function formally described by NIST 800-56C Rev 2.
HMAC	Hash based Message Authentication Code

HPSE	Hardware Protected Security Environment as defined by SAE J3101
HSM	Hardware Security Module
IV	Initialization Vector
MSB	Most significant bit
NF	Network Formation
OEM	Original Equipment Manufacturers
PKI	Public Key Infrastructure
PRNG	Pseudo Random Number Generator
PT	Plain Text
RQST	Request PG
SCM-VIN	Security Certificate Management Vehicle Identification Number
SecOC/E	Secure On-board Communications with optional Encryption
TARA	Threat Assessment and Remediation Analysis
VIN	Vehicle Identification Number

5. CONVENTIONS, CONSTANTS AND TIMEOUTS

5.1 Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

5.2 Conventions

5.2.1 Bit-wise Shift Left Operator (<<)

The double less than sign (<<) indicates a binary bit-wise left shift operation, where the left-hand operand is shifted by the number of bits defined by the right-hand operand. A left shift operation removes the specified number of high order bits and appends the specified number of binary zeroes in the low order bits.

5.2.2 Concatenation (||)

Two vertical bars (||) is the concatenation operator throughout this document. For example, A || B is used to indicate the data B is concatenated to the end of data A.

5.2.3 RQST(*value*)

Shorthand notation for an instance of the SAE J1939 Request PG (PGN 59904) soliciting the PG identified by *value*. In this shorthand, *value* identifies the specific PG by its Acronym, PG Name, or PGN. Refer to SAE J1939-22 for specifications on the Request PG.

5.3 Constants

5.3.1 Const_{SecOC-ENC}

A constant value of 1 used to assign the encryption role for a key from KDF.

5.3.2 Const_{SecOC-CMAC}

A constant value of 2 used to assign message authentication role for a key from KDF.

5.3.3 FV_INIT

The initial freshness value used after a rekey shall be 1. $FV = 0$ and $FV = 2^{32}-1$ are error conditions.

5.3.4 FV_REKEY_TRIGGER

The rekey trigger value for freshness is 2^{31} . Once the threshold is reached a rekey is initiated.

5.3.5 SLIDING_WINDOW_SIZE

The size, in Freshness Values, of a receiver's sliding window. It aligns with the size of the message transmit queue, typically 64 FV.

5.3.6 MAX_AUTH_FAILURE

A total of 100 failed message verifications (CMAC or FV) per session (see 3.36). This value has been selected to balance the false positive requirements of customer support and customer experience.

5.3.7 MAX_NF_FAIL_COUNT

Three failed Network Formation attempts.

5.3.8 MAX_MS

The maximum message data size, which for CAN FD is 512 bits (64 bytes) and is typically 12000 bits for Ethernet (1500 bytes).

5.4 Timeout Defaults

Table 1 lists the definitions and durations for timeouts used in the SecOC/E communications.

Table 1 - Timeout Definitions

Name	Description	Value (ms)
T_R	Rekey duration, reinitialized with each RQST(<i>Rekey</i>) reception	250
T_{SS}	Duration for S_S transition, begins at the end of T_R	250
T_{RFail}	Rekey timeout duration started with the first RQST(<i>Rekey</i>) reception	500
T_{NF}	Network Formation time for Network Leader, begins with the transmission of <i>AnnounceLeader</i> . These values may change pending implementation maturity.	1000
T_{NF_RxCert}	Follower failed to receive Network Leader's certificate, begins with the reception of the Network Leader's <i>AnnounceNetwork</i>	1000
T_{NF_RxSN}	Follower failed to receive Network Leader's Send Network Key message, begins with the member's transmission of <i>JoinNetwork</i>	1000

6. SECURE MESSAGING OVERVIEW

The focus of SAE J1939-91C is the cryptographical securing of Controller Area Network Flexible Data-rate (CAN FD) messages between ECUs. This document provides detailed descriptions of the processes and data flows required to ensure secure onboard communications with optional encryption (SecOC/E). It also describes how trust is established for the network and the required supporting infrastructure.

This section is a very simplified view intended to depict the entire process in a concise manner. Generic terms and abstracts are used rather than the absolute technical accuracy found in later sections.

6.1 Secure Messages

SecOC/E supports three types of messages: Insecure, Authentic and Confidential. Insecure Messages are known by all network members to not have any security features and do not include any security overhead. Figure 2 shows an example of each message type and an expansion of the security overhead. Note that “Authentic Message” and “Authentic and Confidential Message” has the same format for the Security Overhead, with the difference being E=0 for an Authentic Message and E=1 for an Authentic and Confidential Message.

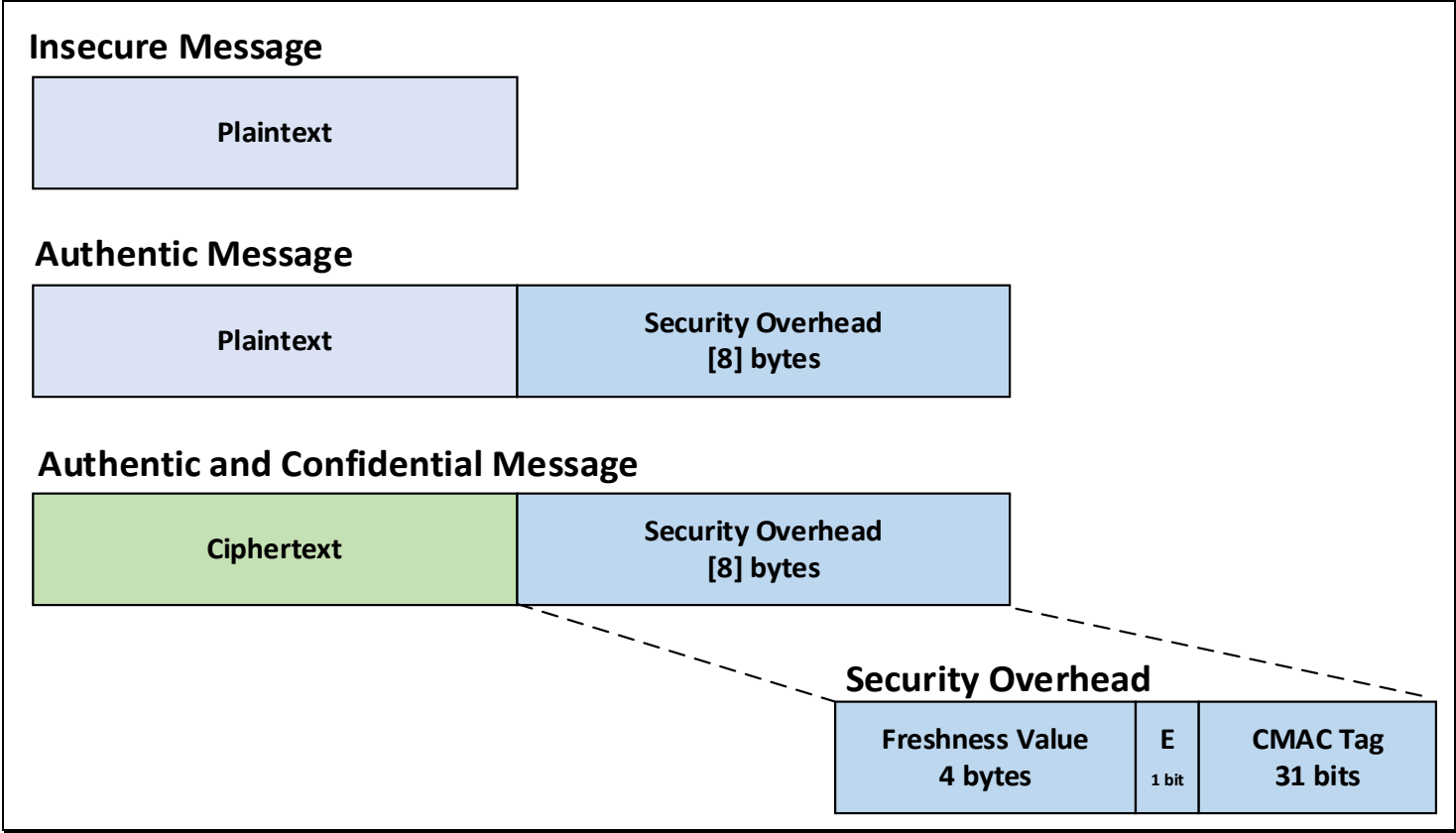


Figure 2 – Message Types and Security Overhead

- Authentic Messages have cryptographically strong evidence that:
- the message has not been tampered with (Integrity),
 - the message comes from a known network member (Authentic),
 - and is not a replay of a previous message (Freshness).

Confidential Messages have all the security properties of Authentic Messages, but also have the message data encrypted, reflected by the setting of the E bit, to create confidentiality for the network members.

SecOC/E secures messages by adding security overhead, in the form of a CMAC, to the message data. To prevent replay attacks, a FV is also added to the message payload. SecOC/E uses shared keys (symmetric encryption) as shown in Figure 3.

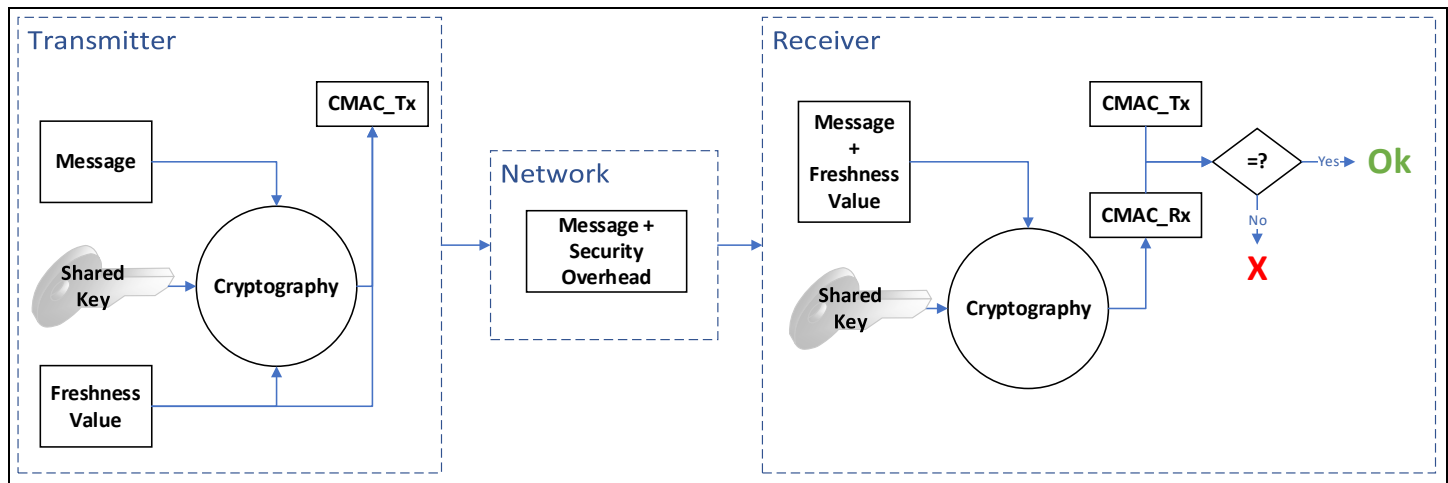


Figure 3 – Message Authentication Process

In the transmitter, the message is encrypted along with the freshness value to create a message authentication digest. This digest is larger than the original message, so it is truncated and added to freshness value to create the message authentication code, CMAC_TX. CMAC_TX is added to the message and both are transmitted on the network.

The receiver parses out the message, CMAC_TX, and the freshness value from the transmitted data. Using the shared key, it repeats the same cryptographic process on the message and freshness value. The resulting digest is truncated in the same manner as the transmitter to create the receiver's CMAC_Rx. The receiver then compares the two CMACs and if they match, the message is accepted, and if encrypted, it would then be decrypted.

6.2 Vehicle Production

SecOC/E relies on a shared secret key for network communication between ECUs, but before that can happen, trust must be established between the devices to form the network. This trust is created through the use of X.509 certificates. The Network Authority (NA), or a Certificate Authority working on the behalf of the NA, signs a Network Certificate for each ECU. This certificate contains the Network Identifier (NID) and the ECU's public key. This process is depicted in the trust diagram in Figure 4. The Trust Diagram Legend identifies the five object types used in a trust diagram. During production, generic ECUs are allocated for use in a specific network by securely updating the ECUs with the latest software and the appropriately signed network certificate. The installed ECUs are then ready for Network Formation.

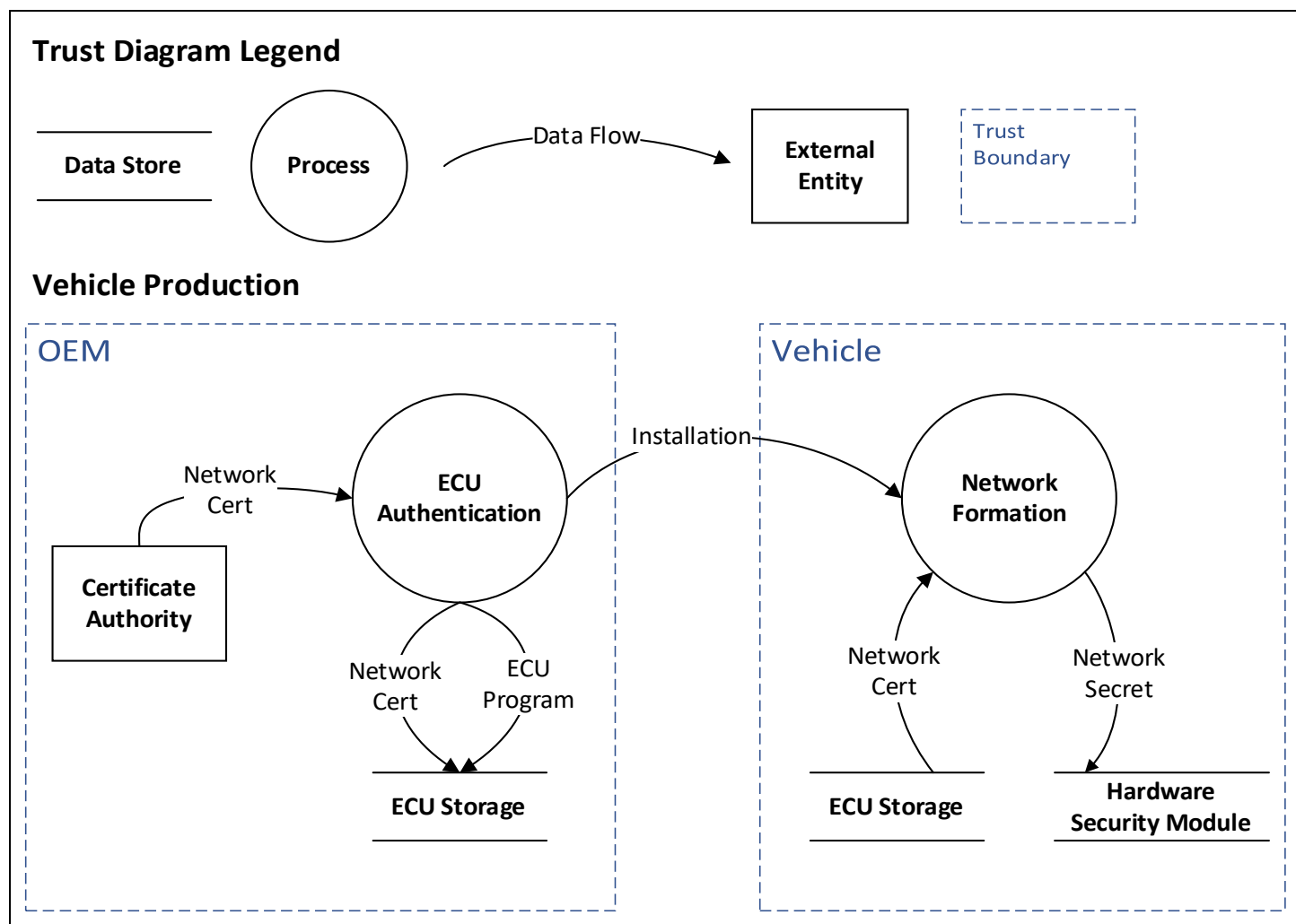


Figure 4 – Vehicle Production Trust Diagram

6.2.1 Network Formation

The OEM shall designate one of the ECUs to be the Network Leader. On the initial power up, the Network Leader (Leader) creates the secure network key (S_N) and looks for other devices, Network Followers (Follower). In a series of one-on-one dialogs, the Network Leader uses public key cryptography to check the authorization and authenticity of each member, and to securely pass the network key to each member.

6.3 Network Operation

The network messaging process, shown in Figure 5, starts each session with rekeying and uses a Hardware Security Module or Hardware Protected Security Environment to prepare CMACs for secure messages. Each time a data flow crosses a trust boundary a Threat Assessment and Remediation Analysis (TARA) should be conducted.

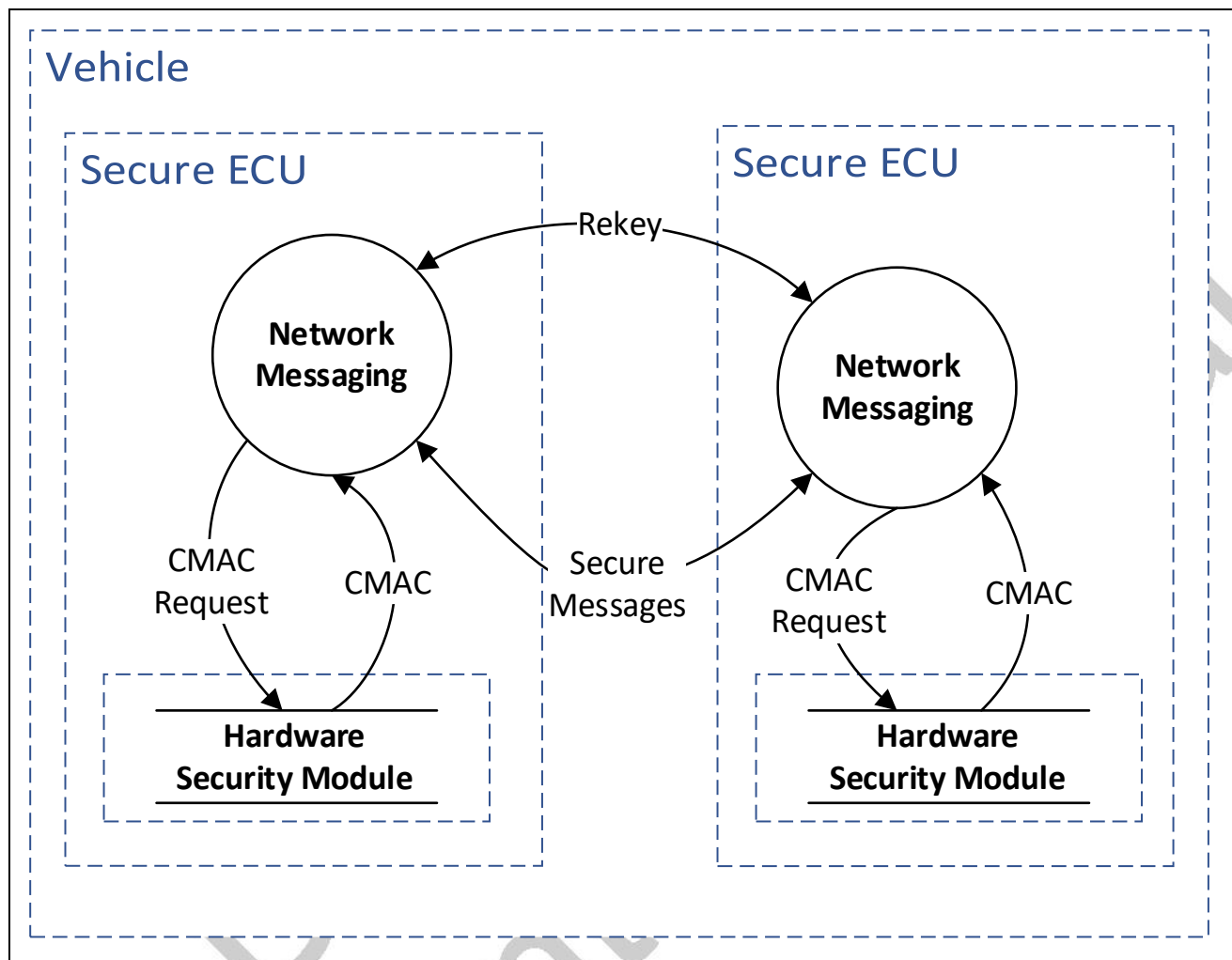


Figure 5 – Vehicle Operation Trust Diagram

6.3.1 Establish Session Key – Rekeying

Once the network is established, a short-term session key is created. Each member publishes a random number. These random numbers are combined with the long-lived network key, S_N , in a Key Derivation Function (KDF) to create a unique session key, S_S . The creation of a new S_S resets the Freshness Value to FV_INIT. The most common reasons for creating a new S_S are: For every power cycle, whenever a device attempts to join the network and whenever the Freshness Value approaches overflow.

6.3.2 Exchanging Secure Messages

All secure messages have authenticity, freshness and integrity properties as part of the CMAC. Network authenticity is created by the ECU passing through the Network Formation process. Freshness comes from the inclusion of the freshness value (FV) in the CMAC. Integrity is assured by the verification of the CMAC in the receiver.

6.4 Vehicle Maintenance

As shown in Figure 6, all devices on a secure network must establish trust before they can be accepted. Spare parts must be securely associated with their intended networks. A secure software update process is required to associate the ECU's software program with the device and the intended network. The secure software update may be performed on or off vehicle. Once the updated device is powered up on the network, it requests Network Formation. If the device is the Network Leader, it initiates Network Formation.

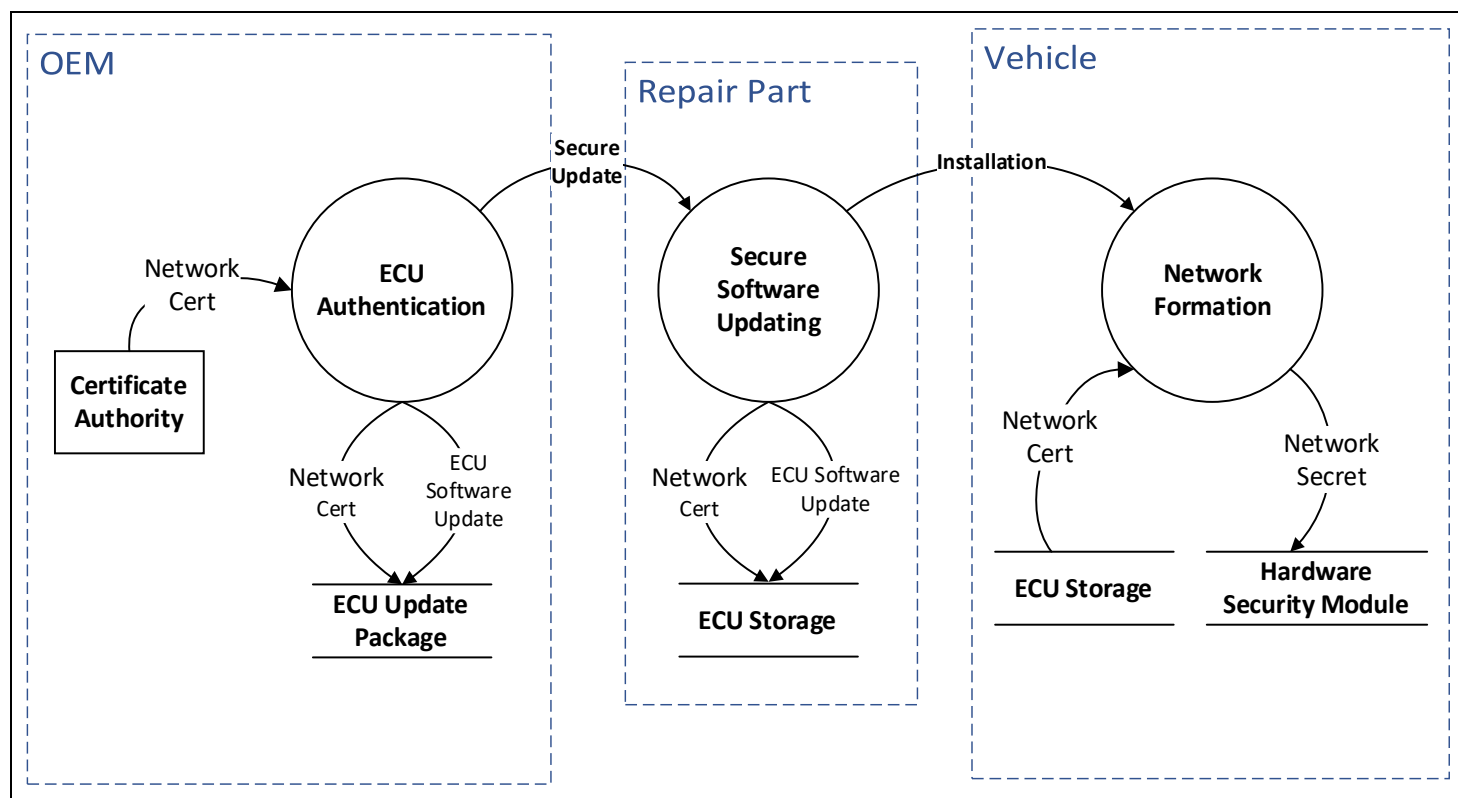


Figure 6 – Vehicle Maintenance Trust Diagram

7. CRYPTOGRAPHY PRIMITIVES

SecOC/E uses public and private keys to establish trust during the Network Formation process and symmetric encryption for message exchanges. This section describes which primitives are used for each operational state.

7.1 Network Formation

In SecOC/E, the elliptic curve primitives are used to verify certificate chains and not used to sign individual frames. The primitives used for Network Formation are shown in Table 2.

Table 2 – Network Formation Crypto Primitives

Feature	Primitive	Comment on Selection
CMAC	AES-128 CMAC	hardware accelerated in typical HPSEs
Hash	SHA-512/256	used in hashing
Key Agreement	X25519	fast and small footprint; out of reach of nation-states
Certificate	X.509	industry standard
Random	TRNG	at a minimum, hardware provided entropy to seed PRNG

7.2 Rekey and Exchange Messages

The security of messages is based on session keys derived from nonces and a long-term network key, S_N . S_N is created by the Network Leader and only shared with network members on the network. A compromise of S_N only breaks the long-term security for that network. S_N is never placed at risk via things like initialization vector (IV) mishaps, birthday-bound attacks, or Ferguson-style issues with truncated MAC collisions. When secured using the HPSE, it should be at least as protected as the X25519 private keys that anchor the root of trust. Compromising the X25519 private key breaks any scheme relying on Diffie-Hellman.

The primitives used for creating session keys (rekeying) and exchanging messages are shown in Table 3.

Table 3 – Rekey and Exchanging Messages Crypto Primitives

Feature	Primitive	Comment on Selection
Cipher	AES-128 CTR mode	128 bits is out of reach of nation-states
CMAC	AES-128 CMAC	hardware accelerated in typical HPSEs
Random	TRNG	at a minimum, hardware provided entropy to seed PRNG
KDF	HKDF-SHA256	HKDF key derivation function built with SHA-256 hash

7.3 Network Member Security Level

Although the complete requirements for Secure ECUs are outside the scope of this document, Secure ECUs are a basic requirement to implement secure network communications. The level of security that an ECU must meet to be considered secure is typically defined by the OEM, and/or fleet operator, based on the overall system security goals as informed by a threat and risk analysis. An important factor in this decision is that the upper bound on the network's security is limited by the member with the weakest security.

At a minimum, member ECUs should provide secure use and storage of cryptographic information (keys, certificates, counters, etc) and a secure environment for processing messages. This security should be based on a hardware root of trust. SAE J3101 - Hardware Protected Security for Ground Vehicles, provides a reference source for implementing hardware security.

7.3.1 Secure Key Storage

Secure key storage provides the means to make it extraordinarily difficult, time-consuming, and expensive to extract keys. If the keys are stored outside secure hardware they shall be encrypted and authenticated (i.e., key wrapped) using algorithms at least as strong as AES-128.

8. FRESHNESS VALUE (FV)

Secure messages require freshness values to prevent replay attacks. A freshness value appears in each secure message. Freshness values are included in the calculations to create the cryptographically secure message authentication code. There is no "global" freshness value shared by all members in the network. Rather, each member that transmits secure messages is responsible for its own FV. Each member that receives secure messages is responsible for tracking the FV for each transmitter in which it receives messages.

8.1 FV Concepts

8.1.1 FV at Transmitter

The FV is a monotonically increasing count that provides freshness for each message tag loaded into the transmit queue. The FV is associated with message tags and not a specific message type or PGN. The transmitter uses a single 32-bit unsigned integer for FV and increments FV for each message it secures. CAN hardware allows messages to queue before being transmitted. However, in preemptive systems, this queue is not necessarily First-In-First-Out (FIFO). The transmitter needs to ensure that a freshness value is not reused, i.e., enqueued Tx messages should each have a unique FV to ensure each message is valid. If not, then it is rejected as a 'replay' by the receiver.

Due to a message's CAN priority, messages may leave the queue "out of FV order". Therefore, the receiving member needs a sliding window to accept them. To support multiple concurrent S_s , such as during rekeying, each S_s requires its own FV sliding window (S_s FV structure) to ensure no re-use of FV.

Since the transmitter has clear knowledge of when the value of S_s changes (rekeys), the transmitter only needs one of the S_s FV structures.

Each controller application in a physical controller employing SAE J1939-91C shall have its own FV values used for broadcasts on a given CAN channel. The FV is incremented for each transmit message tag from that controller application regardless of the message transmitted.

8.1.2 FV at Receiver

The receiver has to accept out of order messages. Some messages may be tagged with the new S_S and other messages (that can arrive later) tagged with the old S_S . Therefore, the receiving member(s) need to have two S_S FV windows. The message's FV shall determine which S_S should be used for authenticating a message.

If there are NT transmitting members on a network, a receiver listens to at least one transmitter, and possibly all NT transmitting members. The receiving members needs two sliding windows (two S_S ::FV structures per received SA) to prevent replay attacks and allow for out-of-FV-order reception of packets for each transmitted SA from which it receives secure messages. The "2x" factor reflects the situation when a network is rekeying and two S_S FV pairs are valid.

The SLIDING_WINDOW_SIZE is based on the size of the hardware message transmit queue, typically 64 messages. There is no need to save the sliding window in Non-Volatile Memory as the system rekeys, and resets FV, on every power cycle.

8.1.3 FV Size

Rekeying is required to avoid FV overflowing and repeating values for an instance of S_S . The number of messages between FV_INIT and FV_REKEY_TRIGGER is known as an epoch. The duration of an epoch is determined by the device's message transmission rate and the number of bits in FV. Message transmissions shall continue during the rekeying process so FV_REKEY_TRIGGER was selected to have the capacity to increase FV without rolling over. The selected scheme uses the most significant bit as the rekey trigger and the 32-bit FV has 2^{31} (2,147,483,648) messages/epoch.

8.2 FV Design Considerations

The transmitter uses a 32-bit unsigned integer for FV and increments FV for each secure message it transmits. The transmitter shall not re-use a FV for a given S_S . FV is the transmitter's state, whereas current FV value (FV_curr) and the maximum observed FV (max_FV_observed) are the receiver's way to estimate state.

Due to rekeying, there are two realities from the receiver's view:

- sliding window 0 is the window for S_S / FV pair #0.
- sliding window 1 is the window for S_S / FV pair #1.

If idx is used as an index to indicate which pair, then max_FV_observed[idx] is the largest FV seen for pair #idx.

The member knows when it is in the process of a rekeying, and thus knows when both the "old" S_S / FV is useful while also using the "new" S_S / FV. There must be a timeout after which the old FV and S_S are no longer valid. Once rekey calculation has ended, the receiver starts a timer, T_{SS} . When T_{SS} times out, the member stops using the "old" S_S / FV.

8.3 S_S Keys and FV

To support the transition between S_S , the receiver needs to keep track of two sets of keys and their freshness values as shown in Figure 7.

```
-- SSFV is key and FV structure
-- It is an array to support transition between two SS's
SSFV[0].mac      -- mac key index
SSFV[0].enc      -- encryption key index
SSFV[0].fv       -- freshness value

SSFV[1].mac      -- mac key index
SSFV[1].enc      -- encryption key index
SSFV[1].fv       -- freshness value
```

Figure 7 - Example of S_S FV Sliding Window Data Structure

8.4 Sliding Window

Due to transmission priorities, the freshness values may be transmitted out of FV order so a Sliding Window is used to ensure that FV's are not repeated. The Sliding Window is a bitfield:

- a bit set to 1 indicates the receiver has seen that particular FV.
- the most significant bit (MSB) reflects the maximum FV value seen so far (`max_FV_observed[idx]`)
- FV values within `SLIDING_WINDOW_SIZE-1` counts of `max_FV_observed[idx]` (let us call the difference between a given observation and max “the delta”) that we have already seen are marked with a 1 at delta bits from MSB.

For a `SLIDING_WINDOW_SIZE` of 64, if we let the Sliding Window bitfield for transmitter $t = W$, then $W = b63 \cdots b0$ can be represented by $W[63] \cdots W[0]$, where $W[63]$ is always equal to 1 and corresponds to highest FV received from transmitting member t .

This is illustrated in Figure 8.

```
-- For each transmitter this member (SA) listens to there is a sliding window
-- W[t][k]
-- t : transmitter ID
-- k : SSFV index, can be [0|1]

fv_curr          -- FV just received
W[t][k].max_FV_observed -- max FV observed in this window
W[t][k].field     -- bitfield, 64 bits
```

Figure 8 – Sliding Window Code Example

8.4.1 Checking Freshness

Before validating a message, a receiving member must first confirm that it is a fresh message. It can use logic as shown in Figure 9.

```
-- function
-- fresh(fv_curr, W) --> true fv_curr is fresh for window W.
--                  --> false otherwise
--
-- W is sliding window structure array
-- shorthand note: W.<field> is shorthand for W[t][k].<field>
--
--     fv_curr the FV just received from transmitter t.
--     W.max_FV_observed maximum FV seen from transmitter t.
--     W.field the bitfield for transmitter t.

delta = fv_curr - W.max_FV_observed

if delta < 0 then                                -- older than max_FV_observed
  if abs(delta) >= SLIDING_WINDOW_SIZE then -- beyond bitfield; not fresh
    fresh = false
  else -- in bitfield
    if W.field[SLIDING_WINDOW_SIZE+delta] == 1 then
      fresh = false -- already seen this FV
    else
      fresh = true  -- haven't seen this FV before
    end
  end
else if delta == 0 then                          -- equal to max_FV_observed (so a replay)
  fresh = false
else                                             -- newer than max_FV_observed
  fresh = true
end

return fresh
```

Figure 9 – Checking Freshness Code Example

8.4.2 Updating the window

Slide the window when the member has received an FV from transmitter **t** that is greater than the maximum FV seen so far from transmitter **t**. A receiving member may use only a portion of the messages sent by the transmitter. In that case, for messages the receiver doesn't care about, it ignores or discards them as usual and does not update the window. When it does receive a message it cares about, it checks to see if the message is fresh and follows the window update algorithm. This is illustrated in Figure 10.

A given member may only receive a small fraction of secure messages from a given transmitter. In this case, the receiving member's bitfield for this specific transmitter will be sparse. In extreme cases, the receiving member's bitfield may always be 0 except for the most-significant bit.

```

-- subroutine
-- update_window(fv_curr, W) --> no output; updates W in place
--
-- W is sliding window structure array
-- W.<field> is shorthand for W[t][k].<field>
--
-- fv_curr the FV just received from transmitter t.
-- W.max_FV_observed maximum FV seen from transmitter t.
-- W.field the bitfield for transmitter t.
--
-- given we've already confirmed W.fv_curr IS FRESH.

delta = fv_curr - W.max_FV_observed

if delta > 0 then                                -- need to shift
  if delta >= SLIDING_WINDOW_SIZE then
    W.field = 0                                  -- clear the bitfield (rather than huge shift)
  else
    W.field = W.field >> delta                    -- shift window (shift in zeros)
  end
  setbit(W.field, SLIDING_WINDOW_SIZE-1) -- set MSB at SLIDING_WINDOW_SIZE-1 of W.field
  W.max_FV_observed = fv_curr                  -- new max
else                                           -- no shift
  setbit(W.field, SLIDING_WINDOW_SIZE-1+delta) -- set bit for fv_curr
end
                                           -- W was updated; nothing returned

```

Figure 10 – Updating the Window Code Example

8.5 Resetting FV

The mechanism for resetting FV is to request a rekey. If either the Tag is incorrect or the messages are stale, the member drops the message and increments the drop count (dropCount). When dropCount exceeds MAX_AUTH_FAILURE, the member requests a rekey. MAX_AUTH_FAILURE is the threshold per rekey and when exceeded the controller application shall also set a fault with a fault occurrence counter. MAX_AUTH_FAILURE is a fixed parameter. There may be some long running applications where dropCount and/or MAX_AUTH_FAILURE should be qualified over a period of time.

When in a faulted state the reception of more failed messages before the completion of the rekey process is still just one occurrence. A successful rekey resets the dropCount. If that is followed by a dropCount greater than MAX_AUTH_FAILURE, it is considered a second occurrence. The fault can be silent.

If a transmitting node dynamically changes its node address (SAE J1939 dynamic addressing), the freshness value continues incrementing and is not reset. A FV reset can lead to re-use of FV with the same S_s. Receiving node(s) have a process for creating a sliding window for each controller application so there is no need to start at FV_INIT.

8.6 Counter-based approach to SecOC/E error state

At the conclusion of a successful rekey, dropCount is reset to 0. Every time a message has an invalid CMAC or stale FV, dropCount is incremented. When a message is valid and fresh, there is no change to dropCount. dropCount is not decremented on the receipt of a valid MAC because doing so allows a half-bus-speed attack on messaging. An attacker can send forged messages at a speed just slightly less than the arrival of good messages and the attack can go undetected, masking the arrival of many invalid CMACs.

8.7 FV Requirements

1. Each transmitting member shall have its own FV. There is no need for the set of transmitting members to synchronize or to otherwise share a common FV.

2. A transmitting member, when starting the use of a new S_S , shall initialize the associated FV to FV_INIT.
3. A transmitting member shall monotonically increase (+1) FV each time it creates any secure message with a given S_S . A simple +1 increase serves the purpose and minimizes the number of times a valid message is rejected as stale simply because it is just outside of the receiving sliding window.
4. A transmitting member shall trigger a rekey on the event of its FV being incremented to a value greater than FV_REKEY_TRIGGER.
5. Each receiving member shall track the FV of each transmitting SA from which it processes messages. This includes maintaining the corresponding sliding window for each FV (S_{S0} and S_{S1}). A given member may only receive a small fraction of secure messages from a given transmitter (see 8.4.2).
6. A receiving member shall associate FV to a specific S_S being used.
7. A receiving member shall support two S_S / FV structures per SA it receives messages from.
8. A receiving member does not track all of the FVs it has seen, rather it shall use a sliding window, where the only FV value that is relevant is the maximum FV value seen for the given S_S . All of the other recent FV values, still in the window, are indicated through a simple bitfield.
 - a. The window shall not extend further back in time more than SLIDING_WINDOW_SIZE-1 counts less than the maximum FV received. This is not a configurable value because of the following considerations:
 - i. The trade-off is to limit how old or stale a message can be and yet still be accepted. Ideally the window can be smaller, to reduce the opportunities for mischief in re-ordering of messages. But a too narrow window can result in an unacceptable number of legitimate messages being rejected because they were delayed by a burst of higher priority messages.
 - ii. It is useful to map FV into a type that allows simple bit-shifting to drive the windowing process. A bigger number like, 128, does not map as easily.
9. A receiving member shall reject “stale” messages, that is, those with an FV less than the maximum FV and outside of the window.
10. A receiving member may accept out-of-order FV's that fall within the window.
11. A receiving member shall discard replayed messages, that is, a message whose FV falls within the window and which has already been seen.
12. A member whose FV rolls over shall fail out of SecOC/E.

9. KEY MANAGEMENT

The following sections identify keys, describe key properties, when keys are used and when or how keys are created.

9.1 When and Where Used

The relationship of keys and parameters to the states of SecOC/E is shown in Figure 11.

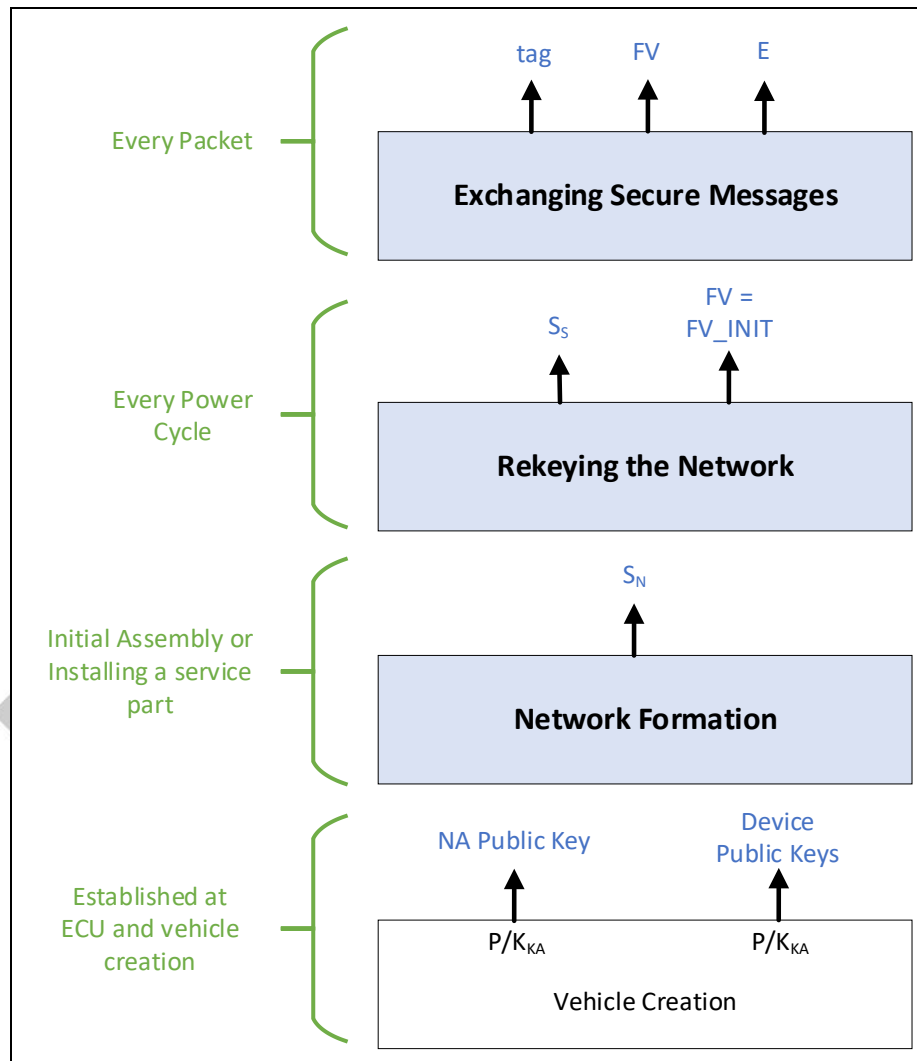


Figure 11 – Relationship between Keys, Parameters and SecOC/E Phases

The process requires that each member have a public/private key pair, where P is the public key and K is the private key and KA indicates that these keys are used for 'key agreement', P/K_{KA}. The Network Formation protocol results in a symmetric key, designated by S. The subscript N indicates it is the network key, S_N. For each power-cycle (and for other reasons, explained in section 11 Rekeying) members use the rekeying process to generate a symmetric key, S, used for vehicle network communication, hence the subscript V. When S_S is created, the freshness value, FV, is reinitialized. For each secure packet sent, S_S is used with FV and some other parameters (explained in the section 13 Exchanging Messages) to calculate a Tag which is sent, along with FV and the E bit to the other members.

9.2 Types of Keys

Network Formation establishes trust between the Network Leader and the Followers via mutual authentication of X.509 certificates. The X.509 certificates contain fields for the NID and the devices' public key, P_{FW-KA} for the Follower and P_{NL-KA} for the Network Leader. A summary of keys and their properties is found in Table 4.

Table 4 – Key Summary

Key	Description	Used in Phase
S_S	AES-128 symmetric secret created between all Followers and the Network Leader during rekeying.	Exchange Messages
S_N	AES-128 symmetric secret shared between all Followers and the Network Leader. Created by the Leader on its first power cycle.	Network Formation, Rekey
$S_{NL-FW-CMAC}$	AES-128 symmetric shared secret created between Network Leader and Follower for the CMAC of encrypted S_N before secure transfer.	Network Formation
$S_{NL-FW-ENC}$	AES-128 symmetric shared secret created between Network Leader and Follower for the encryption of S_N before secure transfer.	Network Formation
K_{NL-KA}	Network Leader's private key for X25519 key agreement. Created by the Network Authority.	Network Formation
P_{NL-KA}	Network Leader's public key for X25519 key agreement. Created by the Network Authority.	Network Formation
K_{FW-KA}	Follower's private key for X25519 key agreement. Created by the device manufacturer.	Network Formation
P_{FW-KA}	Follower's public key for X25519 key agreement. Created by the device manufacturer.	Network Formation
K_{NA-KA}	Network Authority's private key for X.509 certificate signing. Created by the Network Authority.	Network Formation
P_{NA-KA}	Network Authority's public key for X.509 certificate verification. Created by the Network Authority.	Network Formation

9.3 X.509 NID Certificate

The structure of an X.509 v3 digital certificate is shown in Table 5.

Table 5 – X.509 v3 Digital Certificate

Certificate Field	Description	Default Value
Version Number	Certificate format version number V3	X.509 V3 (0x02)
Serial Number	The CA gives a serial number to each certificate it issues for identification.	Nonce
Signature Algorithm ID	This field contains the algorithm identifier for the algorithm used by the CA to sign the certificate (e.g., <i>sha256WithRSAEncryption</i>)	
Issuer Name	The name of the Certificate Authority that issued this certificate.	Network Authority or OEM
Validity period	The universal time type, UTCTime, is a standard ASN.1 type intended for representation of dates and time.	NA specific as some implementations are not able to validate time. Defaults to a 99 year cert.
Not Before	The cert cannot be used before this date.	UTC 01/01/2022 12:00:00
Not After	The cert cannot be used after this date.	UTC 01/01/2121 12:00:00
Subject name	The subject field identifies the entity associated with the public key stored in the subject public key field.	Network Authority or OEM name

Certificate Field	Description	Default Value
Subject Public Key Info	These fields are used to carry the public key and identify the algorithm with which the key is used (e.g., RSA, DSA, or Diffie-Hellman).	
Public Key Algorithm	Algorithm identifier	X25519
Subject Public Key	Public Key	
Issuer Unique Identifier (optional)	The subject and issuer unique identifiers are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time.	
Subject Unique Identifier (optional)	See above.	
Extensions (optional)	1.0.20828.3	SCM-VIN Network Number
...		
Certificate Signature Algorithm	Identifies the algorithm used to sign this certificate.	
Certificate Signature	Signature	

9.3.1 NID and SCM-VIN

The X.509 v3 certificate format allows communities to define private extensions to carry information unique to those communities. ISO 20828, Road vehicles — Security certificate management (SCM), defined an extension for the inclusion of Vehicle Identification Number (VIN) in object identifier (OID) SCM-VIN 1.0.20828.

The SCM-VIN is used to store this protocol's NID where the NID should consist of VIN || Network Number where the Network Number identifies one of (optionally) many networks.

9.4 Crossing Trust Boundaries

SecOC/E relies on cryptography and its use of many keys for its data integrity and optional confidentiality. The described protocol handles the on-board transfer of keys (Network Formation and Rekeying) but these same properties must be maintained during the secure transfer of keys across the following trust boundaries:

- ECU to NA
 - Each device will have to be supplied with an association between its serial number and public key (P_{FW-KA})
 - The association will require its own integrity mechanism and its creation should be audited by the NA.
- NA to Vehicle and Diagnostic Tool to Repair Part
 - Network security relies on the secure software update process for code, keys and certificate installation.
 - Network authority may be separated from software update authority but will still rely on its integrity.
- Vehicle to Diagnostic Tools (DT)
 - This standard does not extend to diagnostic tools (DT). Although, if the DT had the appropriate certificates, an extension would be possible in a Unified Diagnostic Services (UDS) Standard non-compliant manner. This extension may expose the network's S_N via a compromise of the DT.
 - On non-encrypted SecOC/E networks the DT can record and interpret network traffic but it will be unable to verify message integrity.
 - On encrypted SecOC/E networks the DT can record traffic but without participating in Network Formation it will not be possible to interpret the data. Recording Network Formation will not allow for post processing to interpret recorded data.

9.5 Key Life Cycle Management

SecOC/E does not require recording or tracking of the network keys for any network. SecOC/E does require the creation and control of private and public keypair for each device on a network (K/P_{FW-KA}) and at least one private and public keypair for a fleet of networks (K/P_{NA-KA}). Like everything else on a network, keys have a life cycle and an example from NIST 800-57 is shown in Figure 12.

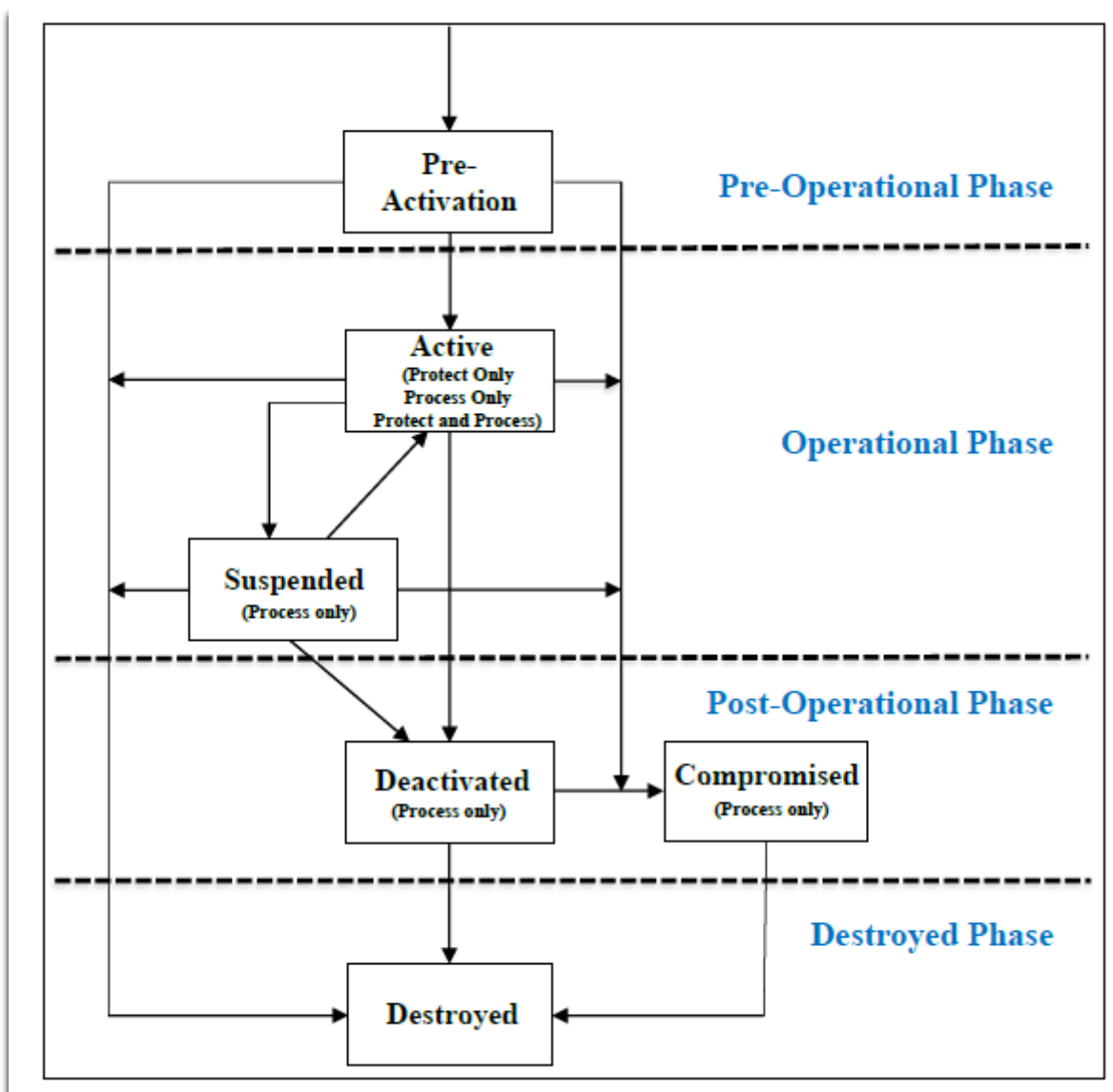


Figure 12 – NIST 800-57 Figure 5 Key-Management States and Phases

The NIST key life cycle model has four management phases, 6 key states and 15 state transitions. Up front planning for key management saves organizations time, money and provide a framework for keeping their keys secure.

9.6 Network Authority

The ultimate arbitrator for authentication and authorization is the Secure Programming process as it installs the algorithms that establish the security layers. Once established, the authentication and authorization of SecOC/E can be administered independently by a Network Authority (NA) through the secure transfer of the SecOC/E root certificate's private key to the NA.

10. NETWORK OPERATION

10.1 Simplified State Machine

There are three SecOC/E states for controller applications: Rekeying, Exchange Messages, and Network Formation. A simplified version of these states and their transitions are shown in Figure 13 without faults and retries.

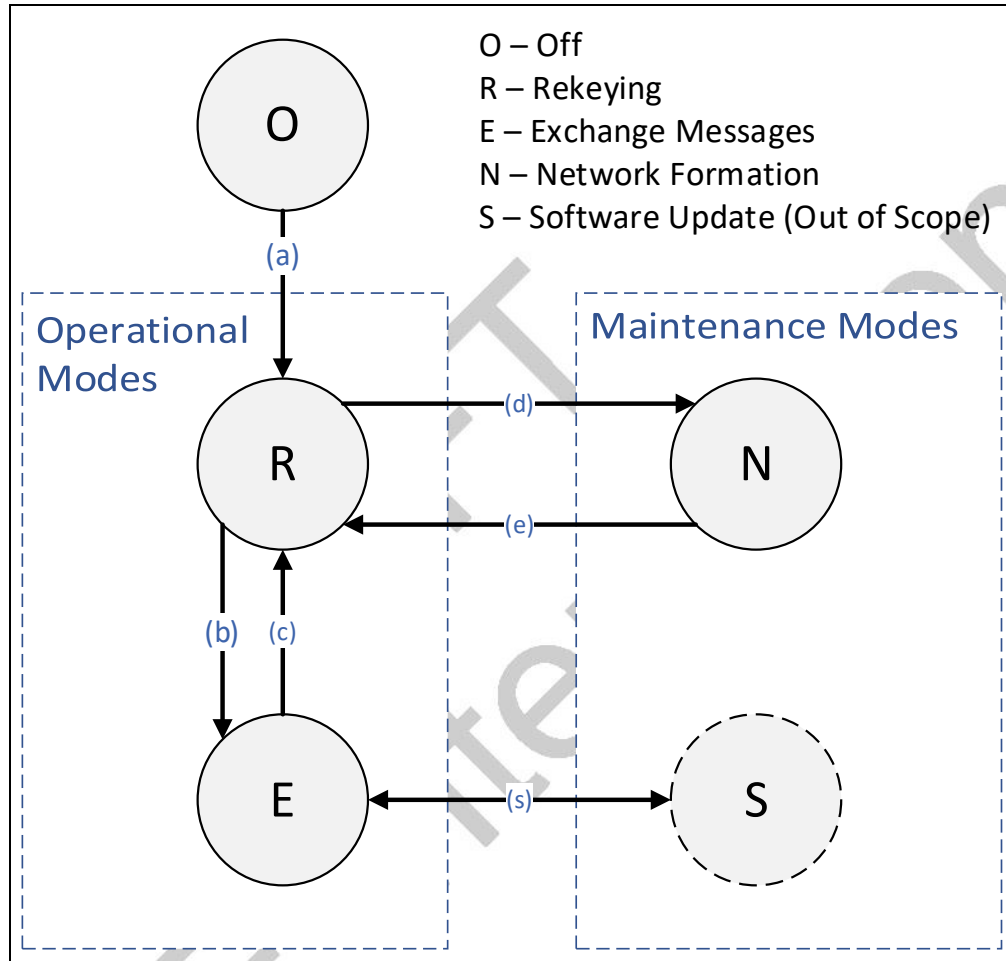


Figure 13 – Simplified State Machine

These states are not mutually exclusive and can be active simultaneously for different members of a network. Simultaneous states and transitions between S_S are described in section 10.3.

At a vehicle level, transitions to Network Formation may be handled similar to Software Updates where the vehicle is not fully operational and may be in a Maintenance Mode. Implementations should consider when NF may or may not be allowed.

10.1.1 Rekeying (R)

Rekeying is the first network activity for each power cycle. At the start of rekeying, each member uses S_N to send a cryptographic hash of the NID to the Network Leader. If the member's hash does not match the Network Leader's hash Network Formation is required. Rekeying uses S_N along with entropy provided by each member to create a short-lived session key, S_S . When S_S is created, the freshness value (FV) is reset to FV_INIT.

10.1.2 Exchange Message (E)

All members with a common S_N exchange messages using: S_S , an updated FV, the data being sent, along with 26 bits containing the PGN and SA, and the E bit to calculate a 128-bit message authentication Tag_CMAC. Only a portion of the Tag_CMAC, called Tag, is sent in the secure message.

10.1.3 Network Formation (N)

Network Formation uses public key cryptography to check authorization and authenticity of potential members of the secure network. Each potential member has a key-agreement keypair P/K that allows for exchanging S_N over the network, securely, in the face of an active adversary. The result of this state is the shared knowledge of a long-lived network key, S_N .

Network Formation is required whenever a Follower's CMAC of the NID does not match the Network Leader's. This may happen when the Network Leader is new to the network, during assembly or repair, at which time all Followers transition to (N). Similarly, all Followers transition to (N) when installed for the first time.

At the conclusion of Network Formation, a rekey is required for all members so they share a common S_S .

10.1.4 Simplified Transitions

- (a) ECU power cycle.
- (b) Rekeying has created a new S_S .
- (c) Rekeying has been requested.
- (d) CMAC of the NID mismatch between Network Leader and member(s).
- (e) Network Formation has completed and the members know the value of S_N .
- (s) Software Update is out of scope for this standard but has been referenced here as another example of a transition to maintenance mode and to point out that all messages that affect the operational mode should be secure.

The first state after a power-cycle is always rekey, (R). If Network Formation is not required all members transition to (E). If Network Formation is required, only the Follower(s) without the correct S_N transition, along with the Network Leader, to (N). The remaining members continue exchanging messages in state (E). If Network Formation was required for any Follower, then all members transition to (R) once Network Formation completes. This allows the new Follower(s) and the previous Follower(s) to arrive at a common S_S and freshness.

10.2 Cryptography for Secure Messages

Authentic Messages are created using an AES-128 CMAC function to produce a Tag from the message data and a nonce, as shown in Figure 14.

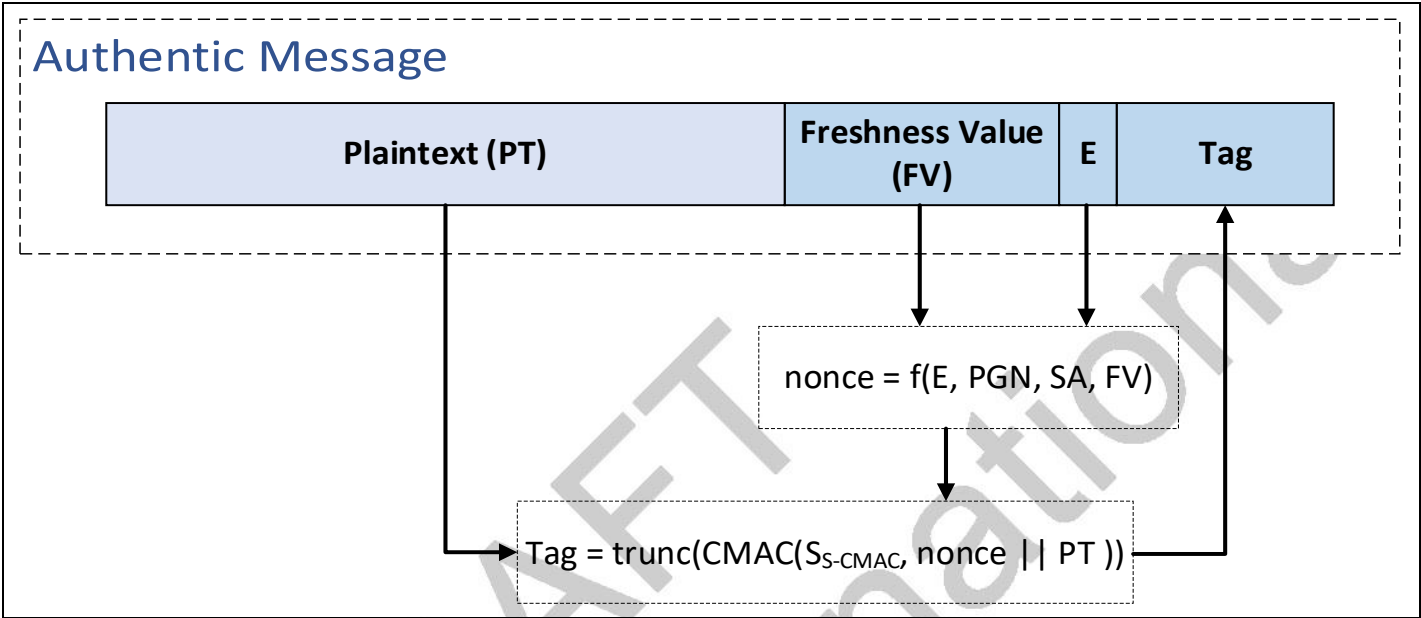


Figure 14 – Creating an Authentic Message

The terms used in Figure 14 are defined in Table 6.

Table 6 – Authentic Message Components

Term	Definition	Length, Bits
CMAC	cipher message authentication code function using AES-128	128
FV	Freshness Value updated for this message, and only for this message	32
E	encryption indicator bit, zero for non-Confidential Messages	1
Nonce	see section 13.1	64
PGN	3-byte Parameter Group Number as per SAE J1939-22 section 6.1.3	24
PT	Plain Text (data), padded as per NIST 800-38B if required for CMAC calculation, not transmitted.	MAX_MS – Security Overhead
SA	message's source address	8
Tag_CMAC	Output from CMAC function	128
Tag	31 most significant bits of Tag_CMAC	31
Security Overhead	FV E Tag	64

Confidential (and Authentic) messages use the same process for generating the Tag but with encrypted data (ciphertext). The encrypted data is created using AES-128 in Counter Mode (CTR) and an Initialization Vector consisting of the nonce concatenated with a block count.

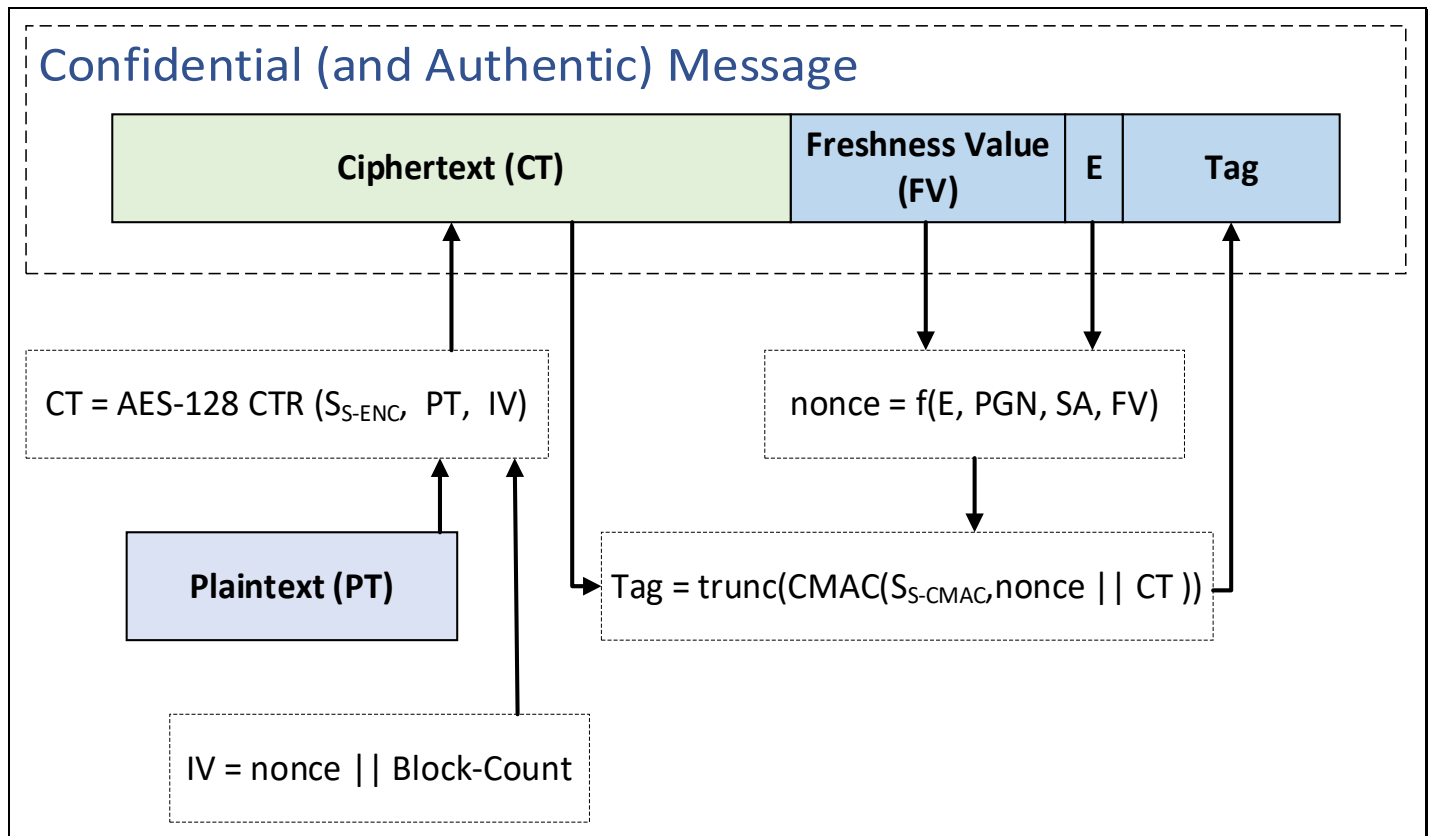


Figure 15 – Creating a Confidential Message

The additional terms introduced in Figure 15 are defined in Table 7.

Table 7 – Additional Confidential Message Components

Term	Definition	Length, Bits
Block-Count	counter-mode AES, incremented for each block encrypted	64
CT	ciphertext; AES-128-CTR(S_{S-ENC} , PT, IV)	[0, MAX_MS – Security Overhead]
IV	initialization vector: nonce block count	128

The transmitter sequence, encrypt then CMAC, has been selected to be the most secure (prevents padding oracle attacks) and efficient for the receiver. For the receiver: if CMAC is good then decrypt, else do nothing.

The Application Layer may request any Authentic Message to become Confidential, i.e.: encrypted. The encrypted data only exists within SecOC/E Presentation Layer services and below. The Application Layer never sees the encrypted data or the E-bit, nor does it need to. The transmitting application requests *SendSecOC/E(PGN, data, Encrypted)*. The Presentation Layer has access to SecOC/E keys to encrypt and tag the data while setting the E-bit. This message is then passed along to J1939-22 services for Session, Transport and Network layers/services.

The receiving device gets the encrypted/tagged message from its J1939-22 services which it then passes to its SecOC/E Presentation Layer services. Those services verify the tag, check the E-bit, then decrypt the data and pass *ReceivedSecOC/E(PGN, data)* to the application. The application knows it has verified data from an authorized device and can use the data.

10.3 Detailed State Machine

The state machine shown in Figure 16 is more detailed than the one in 10.1. It shows many times where a member may have two concurrent states, for example: (R) and (E) for Followers and (N) and (E) for Network Leaders. Concurrent states are indicated with the “,” notation in the state. Also included are retry loops, failure paths and a new Failure state. It depicts the transition between S_S values where both values are valid for a short period of time. The key transition phases (E0, E1 and E1, E0) shall end without waiting for TSS to expire if network formation is required.

The transition event out of Network Formation is different for the Network Leader when compared to Follower(s) as indicated with the “/” notation in the transition. Once an ECU goes from “off” (O) to (E), the normal flow is through (R) to (E) with an opportunity to reform the network in a branch, through (N). That usually happens only on startup with the recognition of new ECUs on the network.

The Network Leader’s (E, N) state is processor intensive and some implementations may not be capable of doing both. System designers may have to allow for the case where the Network Leader does (N) only and cannot send secure messages during Network Formation.

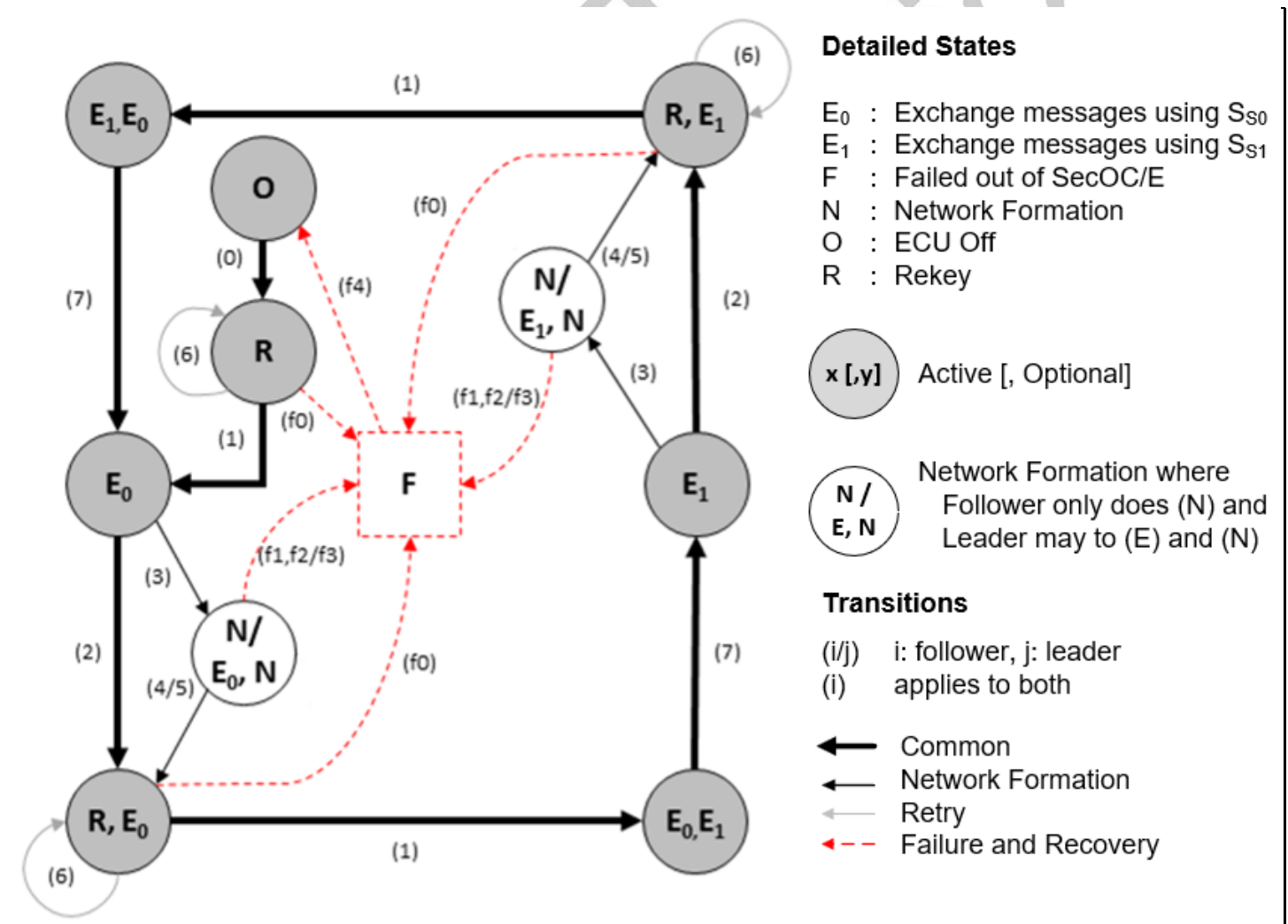


Figure 16 – Detailed State Machine

10.3.1 Detailed Transitions

(0) ECU power cycle.

- (1) T_R timeout, successful rekey
- (2) Rekey triggered
- (3) S_N mismatch detected and ready for Network Formation. Not all leading devices are capable of simultaneous secure message transmission and Network Formation. See 12.3.1 for more information.
- (4) Follower received S_N from *SendNetworkKey* message
- (5) Network Leader T_{NF} timeout or sent last *SendNetworkKey* message
- (6) T_{Rfail} timeout, a member failed to receive either the Network Leader's cert or S_N , the member triggers rekey after T_{SS} , while others continue to communicate securely
- (7) T_{SS} timeout

10.3.2 Failure Transitions

- f0 : rekey failed after 3 tries
- f1 : NID error, invalid certificate, or wrong NID (Follower)
- f2 : Network Formation failed after 3 retries (Follower)
- f3 : Network Leader fails to find or generate S_N
- f4 : power off ECU

10.3.3 State Prioritization

Boot times are different for devices. Therefore, is it possible for a Follower to boot, and request rekeying, while Network Formation is in progress. Followers receiving *AnnounceNetwork* and *AnnounceLeader* shall process these messages after rekeying. This may create short term errors, but the error correction process is rekeying and rekeying occurs at the end of every Network Formation.

10.4 Sequence Diagram

Figure 17 shows a rekey sequence from S_{S0} to S_{S1} where both S_S values are valid for T_{SS} . The corresponding states are listed on the left with the valid S_{Sn} as a subscript.

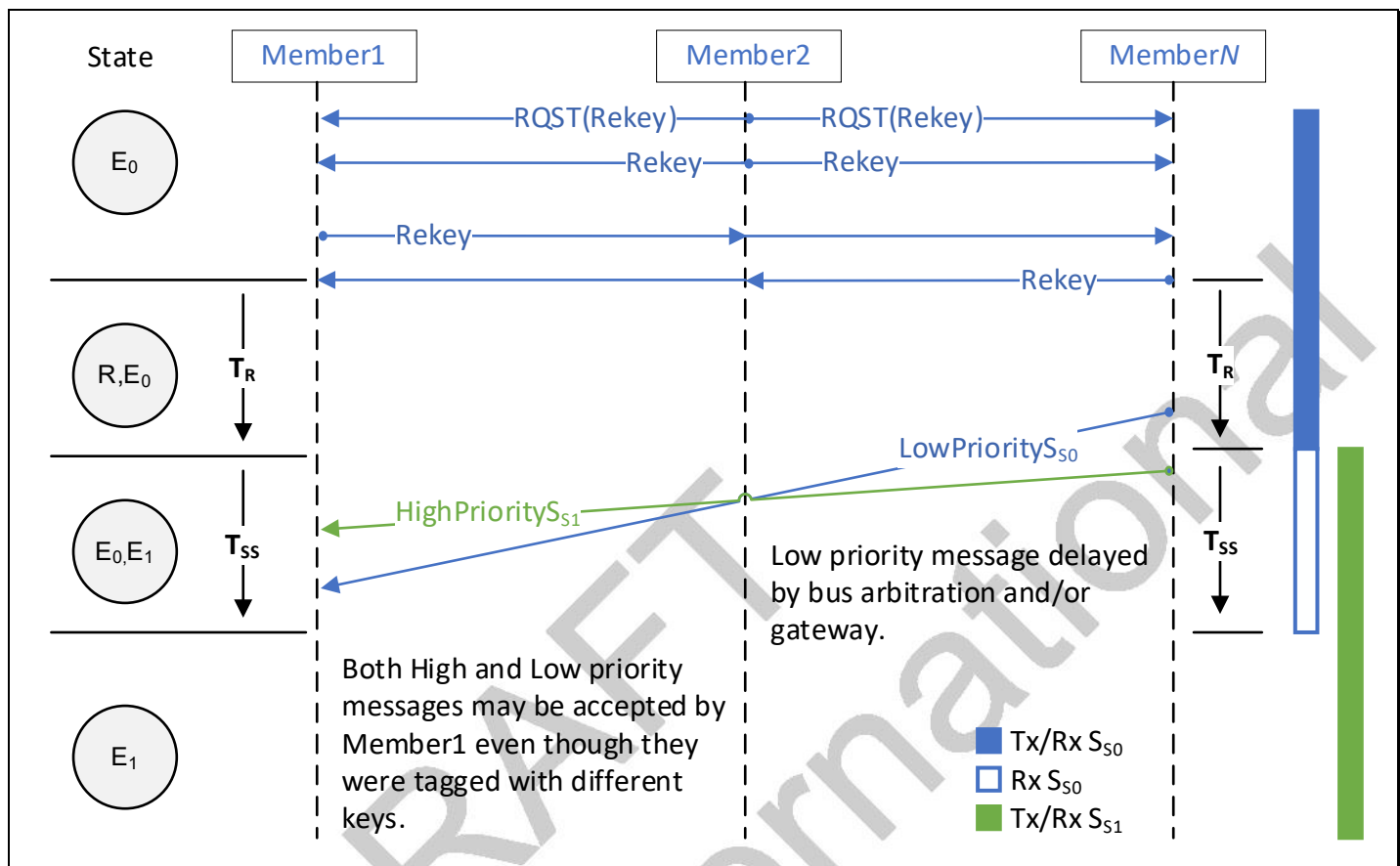


Figure 17 – Rekey Sequence and Timers

The process is kicked off by member2 requesting a rekey by transmitting RQST(Rekey) and its Rekey. The other members respond by transmitting their Rekey message. The reception of each Rekey message resets the T_R timer and re-initiates the S_{S_1} calculation. Once T_R expires, the members start using S_{S_1} for all messages and start the T_{SS} timer. It is during this T_{SS} period that messages created by both S_{S_0} and S_{S_1} are valid and may be received out of S_S sequence due to bus arbitration or gateway delays. There are two ways to end the key transition: S_{S_0} message count and T_{SS} timeout.

Once a message created using S_{S_1} has been received, a maximum of SLIDING_WINDOW_SIZE messages following the largest sequence number seen under the S_{S_0} shall be accepted no matter how quickly they arrive. This allowance represents the hardware transmit queues within which messages may have been generated and “sent” before the key changeover occurred, yet are actually still waiting to succeed at priority arbitration.

The timeout of T_{SS} timer is useful for nodes with few transmits (potentially none if the node only responds to requests, or only listens to others’ data) and might not exhaust the message window.

10.5 Network Operation Requirements

1. There shall be three operational states: Rekeying, Exchanging Messages and Network Formation.
2. The states are not mutually exclusive and shall be capable of simultaneous operation.
3. Authentic Messages shall be created using the functions declared in Figure 14 using the terms defined in Table 6.
4. Confidential Messages shall be created using the functions declared in Figure 15 using the terms defined in Table 6 and Table 7.

11. REKEYING

11.1 Simplified Process

Rekey is how the members generate and share a session key, S_S . Rekey uses symmetric key cryptography using a pre-shared long-lived secret, the network key, S_N . The normal result of rekeying is a new symmetric key, S_S and its associated freshness value is re-initialized. The simplified Rekey process is shown in Figure 18.

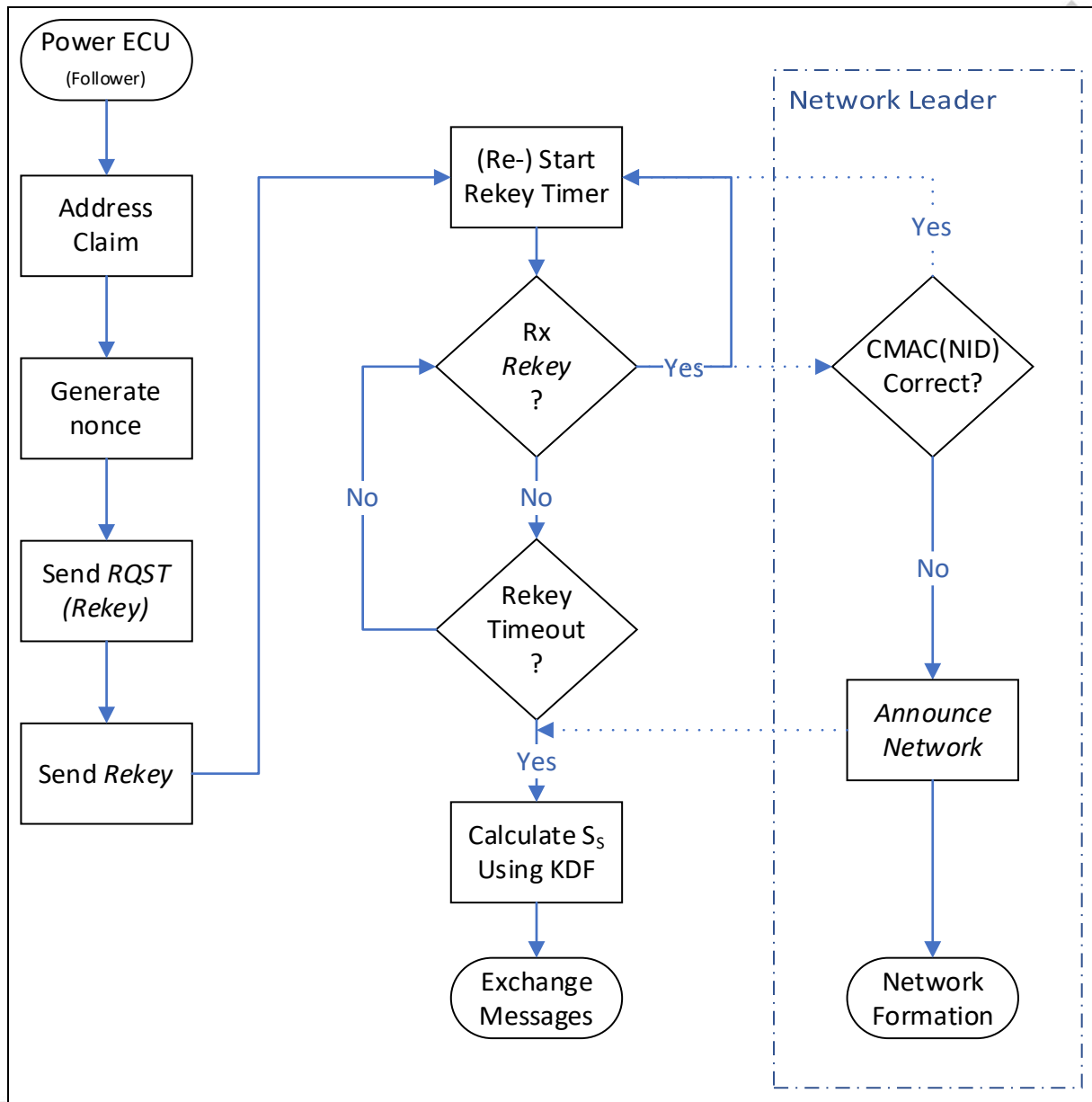


Figure 18 – Simplified Rekey Process

11.1.1 Generate nonce

The nonce is a random number (128-bit unsigned integer) that should be generated by a True Random Number Generator (TRNG) or a Pseudo Random Number Generator (PRNG) seeded by a TRNG.

11.1.2 Send RQST(Rekey)

A member transmits the RQST(Rekey) message to the global source address when it wants to request a rekey. Any member may send the RQST(Rekey). Ideally this is sent only once, by a single member, per rekey instance; however, this is unlikely

during system power on. The protocol handles the cases where slower booting members miss the initial start and act as if they are kicking off the process. See APPENDIX A for examples.

11.1.3 Send Rekey Message

The *Rekey* message is sent by every member in reply to the RQST(*Rekey*) message. The message contains both entropy for the new session key, S_S , and an indication of the S_N being used by the member. The entropy is the 128-bit unsigned nonce described in 11.1.1. The S_N indicator is a CMAC of the NID || nonce that is created using S_N without revealing S_N . If the Network Leader detects a Follower with the wrong CMAC result, it will start the Network Formation process. See 11.1.6. The nonce has been added to the CMAC to make each member's message unique and to prove the member is authorized to participate in this rekey (i.e., shares the understanding of S_N). Members shall reject Rekey messages that fail CMAC verification.

11.1.4 Rekey Timer (T_R) Processes

Rekeying is a timed event that starts with the first *Rekey* transmission and restarts with the reception of every *Rekey* message. During the T_R period, each member, including the Network Leader, is gathering nonces. The nonces are ordered in a table from smallest to largest values.

$$Nonce_{All} = nonce_{Smallest} || nonce_{Small} || nonce_{Large} || nonce_{Largest}$$

Should an attack attempt to replace entropy values, each member shall fail out of Rekey if its nonce value is not part of $Nonce_{All}$. At the expiration of the T_R timer, each member shall calculate and begin using the new S_S value with the FV set to FV_INIT.

11.1.5 Calculate S_S

The network's session key, S_S , is calculated using a HMAC Key Derivation Function, HKDF-SHA256.

$$S_S = KDF(S_N, Const_{SecOC}, H(nonce_{all}))$$

where,

S_N = long-lived secret network key.

$Const_{SecOC}$ = fixed "role" constant to ensure some other protocol also using the same X25519 keys cannot function as an oracle. This is a standard practice when using a KDF in case the ECUs do Elliptic Curve Diffie-Hellman (ECDH) with each other for another purpose.

$H(nonce_{all})$ = SHA512/256 for the hash operation to get a usable symmetric key from the entropy. This provides collision resistance, whitening and reduction of cipher's key size. If even just one member chooses its nonce randomly, the bits of $H(nonce_{all})$ are unpredictable.

It does not matter if untrusted nodes listen in to the rekey (or even if they try to "participate"), because they cannot know S_N , and, therefore, cannot compute the KDF and learn S_S .

In systems using the optional encryption, the S_S calculation is completed twice to create unique keys for encryption and message authentication.

$$S_{S-ENC} = KDF(S_N, Const_{SecOC-ENC}, H(nonce_{all}))$$

$$S_{S-MAC} = KDF(S_N, Const_{SecOC-CMAC}, H(nonce_{all}))$$

11.1.6 Announce Network

The Network Leader monitors all the *Rekey* messages so it can calculate S_S and to detect new Followers. The Network Leader detects new Followers when their CMAC does not match the value calculated with the Network Leader's S_N . When a mismatch is detected, the Network Leader broadcasts the *AnnounceNetwork* message when the Rekey Timer (T_R) expires. The *AnnounceNetwork* message uses the 16 byte result of CMAC(S_N , NID) as defined in section 14.

The CMAC of NID using S_N protects the value of S_N while providing a quickly calculated value for members to assess. If a device's CMAC does not match, it is to immediately stop broadcasting secure messages and wait for Network Formation. If the Network Leader detects a Follower with the wrong CMAC result, it shall wait for rekeying to complete, and then start the Network Formation process.

11.2 Rekey Concepts

11.2.1 Freshness

Rekeying also synchronizes freshness. Therefore, there is no need for separate freshness synchronization function in SecOC/E. Rekeying is a lightweight process typically consisting of N+1 CAN-FD frames, 1 TRNG, N+1 single-block hashes, a maximum of two KDF calls for a single node. For N=40 members, that can be about 10 ms of bus utilization and 0.5 ms of HSM calculation based on a 200+ MHz automotive controller with an HSM.

11.2.2 Adding/removing a node

Nodes leaving the network will take S_N with them. Followers new to a secure network will go through the Network Formation process in order to obtain a new S_N . Leaders will use Network Formation to establish their value of S_N for the new network. This is explained in section 12. There is no impact to the network when a node is removed from the network.

11.2.3 Transition between old and new S_S

In the process of transitioning to use the new S_S , a member may receive messages that were tagged using the previous key. The member should retain the previous key and FV sliding window to allow the member to validate messages during the S_S transition.

11.3 Timing and Control

There are two broadcast messages that are part of every rekey. This seemingly simple design could collapse under a runaway cascade of replies caused by members missing the start of the process. This is avoided through the use of timers and flags described in Table 8 and Table 9. Scope refers to visibility of data where global is visible throughout SecOC/E code and local is only needed within the rekeying logic.

Table 8 – Rekey Controls

Data	Scope	Description
S_N	global	symmetric network key, AES-128
nonce_table	local	dictionary of nonces; key = source address, value = nonce
first	local	flag; true = this App started rekey process; false = another App started process
repeat	local	flag; true = this App shall repeat its nonce on reception of Rekey

Each controller application has their own timers. There is no global synchronization of timers.

Table 9 – Rekey Timers

TIMER	Description
NAME	SAE J1939 address claim stabilization
T_R	Timeout indicates the end of rekeying and allows for the calculation of S_S
T_{RFAIL}	Timeout indicates that rekey did not complete within the maximum allowed time.

11.4 Detailed Process Flow

The *RQST(Rekey)* message means the sender of the message is preparing to start the rekey process. The arrival of *Rekey* is the event that triggers a member to participate in the rekey state.

As shown in Figure 19, the reception of a *RQST(Rekey)* message and the state of the rekey timer will determine what happens when a *Rekey* message is received.

The receiving *Rekey* logic will also handle the cases where it is received without a *RQST(Rekey)* message. Note the SA of the *Rekey* message is associated with the nonce as a part of a table of all nonces received.

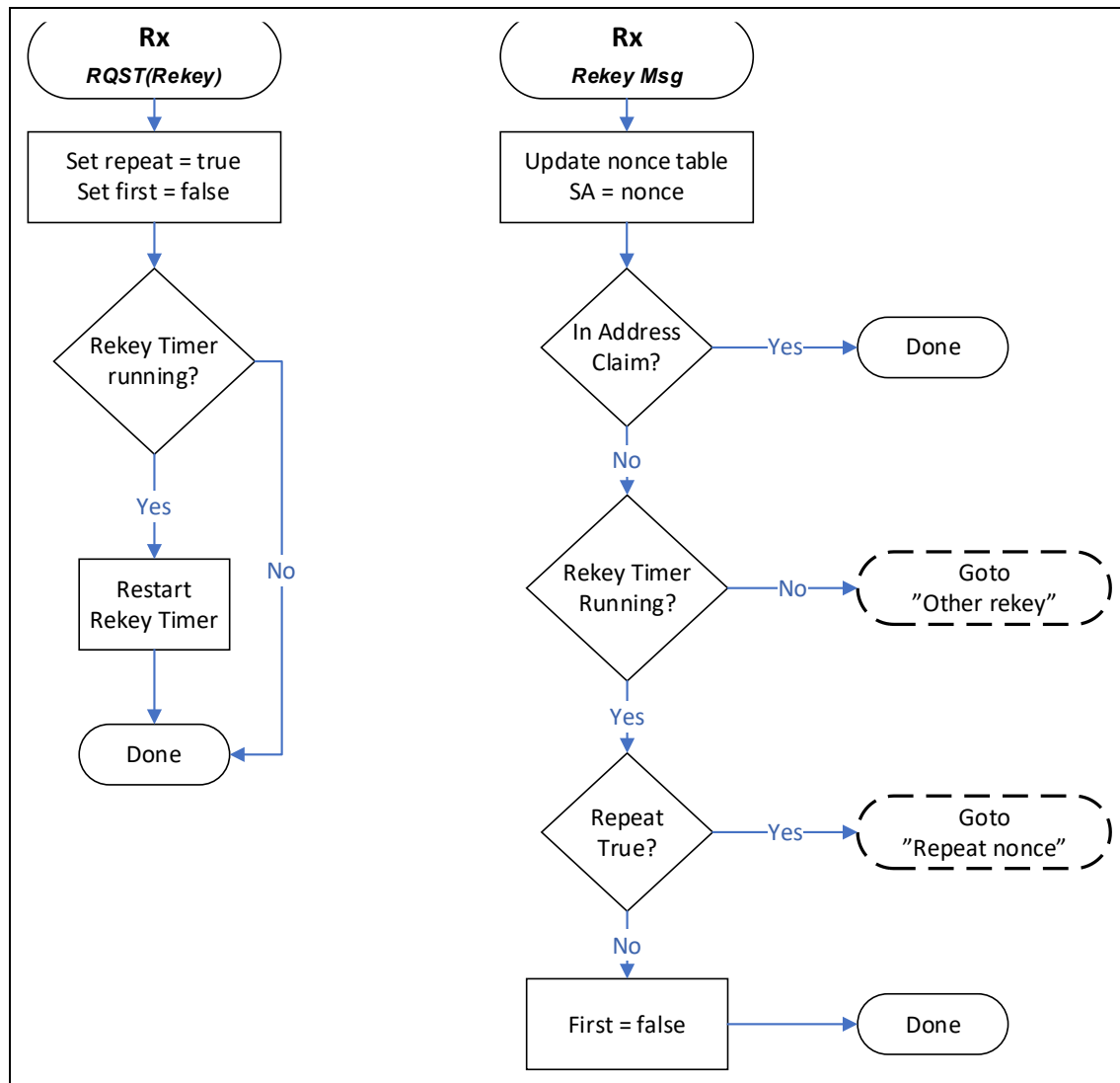


Figure 19 – Rekey Message Processing

The rekey process has three entry points depending on the source of the rekey trigger. The “Power On ECU” entry point is used on boot. The “Other rekey” entry point is used during operation, when either this ECU needs to rekey or another ECU indicates the need to rekey. The “Repeat nonce” entry point is used when a node (A) detects another node (B) has sent a *RQST(Rekey)* message in the middle of a rekey. On detecting the repeated *RQST(Rekey)*, node (A) enters the rekey process at “Repeat nonce”. These entry points and a more detailed process flow are shown in Figure 20. A nonce broadcast may be repeated for a CA that missed it but nonces are never re-used after a rekeying is complete. Once the S_S KDF has been completed the nonce table is cleared.

The “repeat” flag is set when a *RQST(Rekey)* message was received. It is used to determine whether a member (A) will resend its own *Rekey* message after having received another ECU’s (B) *Rekey* message because ECU B has missed the first transmission of ECU’s (A) *Rekey* message.

The “first” flag is set at power-on and every time S_S is calculated. When “first” is set, the ECU shall broadcast a *RQST(Rekey)* when it requires rekeying. It is cleared whenever the ECU receives a *RQST(Rekey)* from another ECU.

The Rekey Timer indicates valid window for receiving *Rekey* messages. It is Initiated by first *Rekey* message and restarted with any subsequent *Rekey* or *RQST(Rekey)* messages. When the timer expires (times out), the ECUs shall calculate the new S_S .

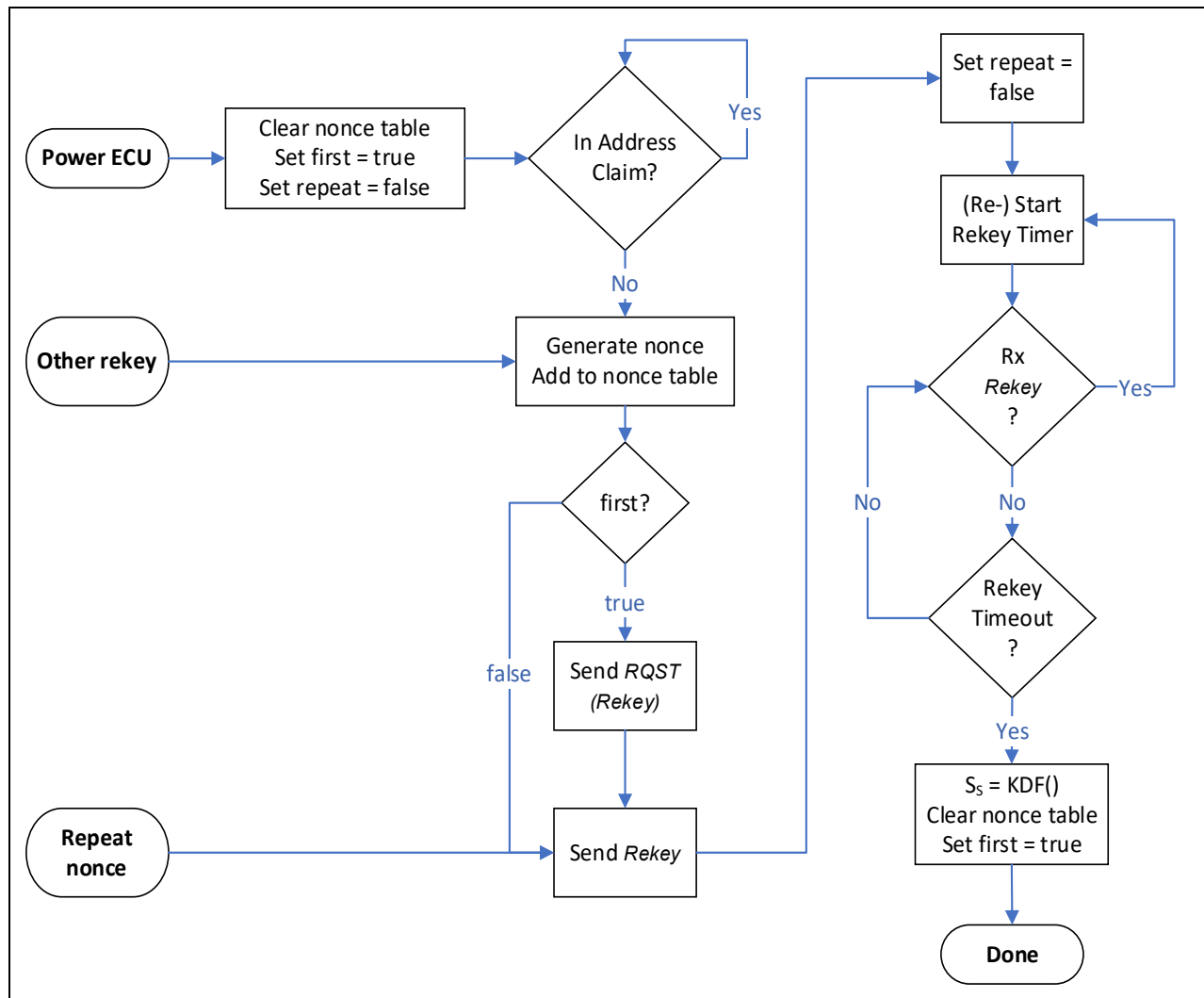


Figure 20 – Detailed Rekey Process Flow

11.5 Missing Leader Exception Handling

At power up, a new member with the wrong S_N will try to participate in the rekeying process. The new member will be successful in rekeying but with an incorrect S_S and since the Network Leader is not present, *AnnounceNetwork* will not be broadcast. After rekeying, the new member will not be able to authenticate any message. Shortly, it will cross the $MAX_AUTH_FAILURE$ threshold to trigger new rekey and it will set a flag. After second rekeying, if it is still not able to authenticate any messages and flag is set, then it will detect by itself that it does not have the correct S_N and also no Network Leader on the bus to provide correct S_N . Under these circumstances, it may set a fault and stop participating in secured communication.

11.6 Rekeying Requirements

1. The secure network shall be rekeyed on every power cycle, where 'rekeyed' means generating and using a new S_{S0} .
2. S_S shall be established before secure message transmission begins to avoid members needlessly accumulating message errors due to invalid tags.
3. A new S_S shall be mathematically unrelated to a previous network key.
 - a. Each member shall share a nonce during rekey.

- b. Nonces shall be unsigned 128 bits.
 - c. Nonces shall be newly generated before each use.
4. A member's nonce shall come from a TRNG or a PRNG seeded from a TRNG.
 5. Each member shall fail out of Rekey if its nonce value is not part of *Nonce_{All}*.
 6. Only members shall have access to the network key S_S .
 7. Members shall not divulge S_S .
 8. Any member shall be capable of requesting a network rekey.
 9. Each pass through the Rekeying process shall increment the rekey count.
 10. Successfully receiving a secure message shall reset the rekey count.
 11. If the Rekey count exceeds four per power cycle the device shall stop transmitting and wait for a power cycle.
 12. During the transition from one S_S to another (T_{SS}), both S_S keys shall be considered valid for receiving secure messages.
 13. During the transition from one S_S to another (T_{SS}), only the new S_S key shall be used for transmitting secure messages.
 14. A request for Rekeying shall not be issued before expiration of T_{SS} .
 15. A transmitting member shall reset its freshness value, FV, to FV_INIT when it starts using the new S_S .
 16. A transmitting member whose FV exceeds FV_REKEY_TRIGGER shall ask for a rekey.
 17. A receiving member whose FV exceeds FV_REKEY_TRIGGER may ask for a rekey. However, it may defer that task to the transmitting member that has exceeded FV_REKEY_TRIGGER.
 18. During rekey, all members shall share their understanding of S_N without revealing the actual value S_N .
 19. Members shall request rekey whenever their SA changes. (The FV should not be reset in the case where a transmitting member changes its source address due to dynamic addressing; see Section 8.5).

12. NETWORK FORMATION

This section provides an overview of the Network Formation process, a detailed description of the interactions between Rekeying and Network Formation, and detailed process flow with message descriptions. Using SecOC/E on multiple virtual secure networks, with different memberships, within a physical network is beyond the scope of this document.

12.1 Simplified Process

Network Formation will cryptographically assure members that they belong on the network and can trust each other. This process is shown in Figure 21 and described in the following sections.

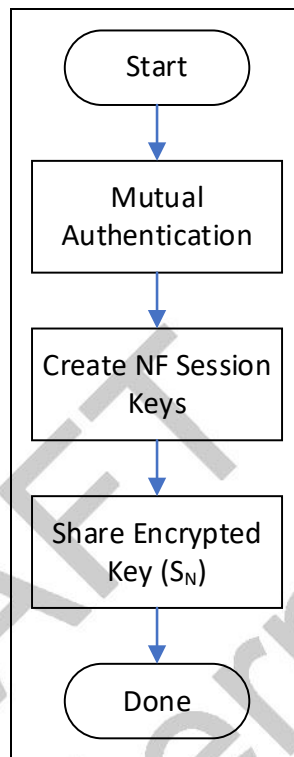


Figure 21 – Simplified Network Formation Process

In general terms, Public Key Infrastructure (PKI) techniques are used to establish trust between the Network Leader and a Follower. On the basis of this trust, Elliptic Curve Diffie-Helman is used to create shared symmetric session keys known only to the Network Leader and a single Follower. The Network Leader then uses those session keys to encrypt and wrap (CMAC) the encrypted long-term network key for transfer to the Follower. This is a one-to-one process so the Network Leader will have to conduct it for each joining Follower but may do so in parallel for multiple Followers.

12.1.1 Mutual Authentication

An ECU becomes an authorized member of a secure network by presenting an X.509 certificate that is valid and with a NID that matches that of the Network Leader. The Network Leader announces its certificate and a 128-bit nonce (nl_nonce) to prevent replays. The Follower checks the validity of the certificate by ensuring that it chains back to the NA/CA's public key and has the same NID as its certificate. The Follower needs the Network Leader's full chain in order to validate it, but all it receives is the leaf. It can get what it needs by assuming its chain is the same as the Network Leader's chain, and just uses its own in place of the Network Leader's.

If either test fails, the Follower does not join in Network Formation and raises a fault. When both tests pass, the Follower joins the Network Formation by broadcasting its NID certification and its own 128-bit nonce (fw_nonce). The Network Leader performs similar tests to the Follower's NID certificate and when they pass, it will move on to creating a session key.

12.1.1.1 Issuing NID Certificates

The NA, or a third-party Certificate Authority (CA), acting as the Network Authority creates a P/K_{NA-CA} pair that is used to self-sign the Network Authority's Root Certificate. The root certificate is then used to create end-entity or leaf certificates for network members. These leaf certificates shall contain a specific NID and the ECU public key. The signing of these NID certificates indicates that the NA authorizes the use of that specific ECU for use on that specific network.

For an integrated member being installed by the OEM during manufacturing, its certificate is issued at manufacture time and proves this ECU belongs on this NID. This is a very strong attestation of authorization.

For integrated members being installed as a service part, its certificate is issued by the NA at the request of some person doing the repair. This is a weaker attestation.

Transferable members may have their certificate is issued later, in the field. In this case, the certificate means that someone wanted to tie this ECU to this network and the NA honors the request. This is a weaker attestation.

It is likely that at some point an ECU will have their P/K_{KA} pair exfiltrated and then the device may be cloned. The onus is then upon the NA to recognize and reject unreasonable requests for NID certs using the same ECU.

12.1.2 Derive Session Keys

The exchange of NID certificates, with their public keys, and the nonces provide enough information for both the Network Leader and Follower to derive shared secrets using X25519. The secure transfer of S_N shall require each Network Leader/Follower pair to derive two shared keys; one for encrypting the network key (ENC) and another for signing the encrypted key with a cryptographic message authentication code (CMAC):

$$\text{derivedKey} = \text{KDF}(\text{key}, \text{salt}, \text{FixedInfo})$$

$$S_{\text{NL-FW-ENC}} = \text{KDF}(\text{X25519}, \text{Const}_{\text{SecOC-ENC}}, \text{nonce}_{\text{joint}})$$

$$S_{\text{NL-FW-CMAC}} = \text{KDF}(\text{X25519}, \text{Const}_{\text{SecOC-CMAC}}, \text{nonce}_{\text{joint}})$$

The KDF structure is defined in NIST SP-800-56C Rev 2 Two-Step Variant as a HKDF key derivation function built with SHA-256 hash. As per NIST, the left most 128 bits are used to create the keys from the 256-bit hash.

X25519 is an ECDH key agreement scheme where the Network Leader uses $\text{X25519}(P_{\text{FW-KA}}, K_{\text{NL-KA}})$ and the Follower uses $\text{X25519}(K_{\text{FW-KA}}, P_{\text{NL-KA}})$ to create a shared key.

$\text{Const}_{\text{SecOC-ENC}}$ and $\text{Const}_{\text{SecOC-CMAC}}$ are role assignment constants for the $\text{KDF}()$ that determine which type (encrypting – ENC or message authentication – CMAC) key is being generated.

$\text{Nonce}_{\text{joint}}$ is a unique nonce for each Network Leader/Follower pair that is derived using SHA512/256 and is added to the KDF to prevent a replay attack.

$$\text{Nonce}_{\text{joint}} = \text{SHA512/256}(\text{fw_nonce} \parallel \text{nl_nonce})$$

12.1.3 Share Encrypted Key (S_N)

The Network Leader shares the network key (S_N) as encrypted cipher text using $S_{\text{NL-FW-ENC}}$. The message is signed with a CMAC of the CT using $S_{\text{NL-FW-CMAC}}$.

$$\text{CT} = \text{AES-128-CTR}(\text{key}, \text{plaintext}, \text{InitializationVector})$$

$$\text{CT} = \text{AES-128-CTR}(S_{\text{NL-FW-ENC}}, S_N, \text{nl_nonce})$$

$$\text{CMAC} = \text{AES-CMAC}(S_{\text{NL-FW-CMAC}}, \text{CT})$$

12.2 Rekey and Network Formation Integration

Network Formation and rekeying are related. During rekey, the Network Leader inspects Follower's *Rekey* message for mismatched S_N . When it detects a mismatch, it sends an *AnnounceNetwork* message so that affected Followers can detect that they do not have the correct S_N . Both the Network Leader and any impacted Follower(s) need to have Network Formation performed. A state machine integration example is provided in Figure 22 where a service part is added to a network.

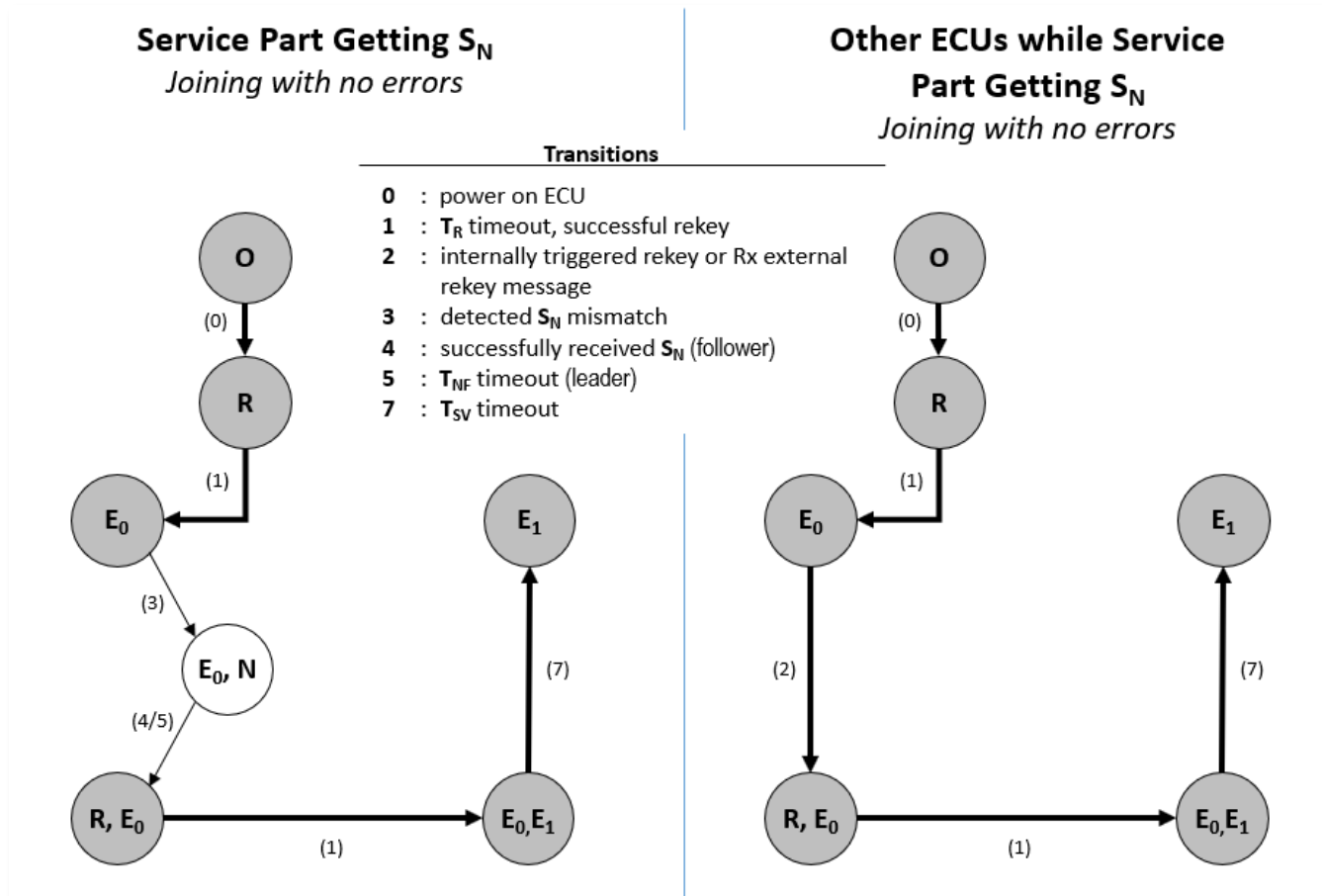


Figure 22 – Service Part Network Formation

The left-side of Figure 22 shows the state transitions for a new Follower and the Network Leader as they participate in Network Formation. The right-side of Figure 22 shows the state transitions for other Followers that do not need to participate in Network Formation. All members participate in rekeying after the conclusion of Network Formation. Network Formation occurs as needed and is expected to be very rare for most members.

In this example, service part transitions:

- Assume the service part is a Follower.
- The service part miscalculates S_{S0} during the first (**R**) because its understanding of S_N is incorrect.
- The service part may send a few incorrectly tagged messages in (**E0**) until it detects S_N mismatch.
- Once it detects its S_N is wrong, it immediately stops using S_{S0} .
- At completion of Network Formation, the service part triggers a rekey.
- Leaving (**R, E0**) indicates completion of rekey. At this point the service part has a common S_{S1} with all members.
- In (**E0, E1**), the service part only participates using S_{S1} . This is unlike the other members, which can use both S_S values.

Other member transitions:

- The other members ignore the Network Formation that is going on between the Network Leader and the service part and stay in (**E0**).
- They transition to (**R, E0**) when the rekey is called for by the service part.
- Once they transition out of (**R, E0**) all members will have a common S_{S1} .

12.3 Detailed Process Flow

A sequence diagram of the Network Formation process is shown in Figure 23. It starts off with the *AnnounceNetwork* message that was required as a result of the Network Leader identifying Follower(s) with an incorrect S_N during rekey. The Network Leader starts T_{NF} at the completed sending of *AnnounceNetwork* while waiting for complete reception of *JoinNetwork*. A Follower identifies that it has an incorrect S_N , ceases secure message transmissions, and waits for the *AnnounceLeader* message to indicate the start of Network Formation. This period has a Follower Network Formation timer, T_{NF_RxCert} , that if it expires, the Follower will increment a Network Formation fail count and trigger rekey. After receiving and verifying the Network Leader's NID certificate, the Follower broadcasts its certificate and nonce. This starts a second Follower timer, T_{NF_RxSN} . The Network Leader verifies the Follower's certificate and derives the session keys. These keys are used to encrypt and sign the encryption with a CMAC. The Network Leader then transmits the *Send Network Key* message. The Follower will verify the CMAC, decrypt and securely store S_N , and then trigger a rekey.

If the Follower does not receive *SendNetworkKey* before T_{NF_RxSN} , it will increment a Network Formation fail count and trigger a rekey. If the Network Formation fail count exceeds $MAX_NF_FAIL_COUNT$, the Follower may set a fault and stop attempting to join SecOC/E for this power cycle.

Each Network Leader and Follower(s) have their own timers. There is no global synchronization of timers.

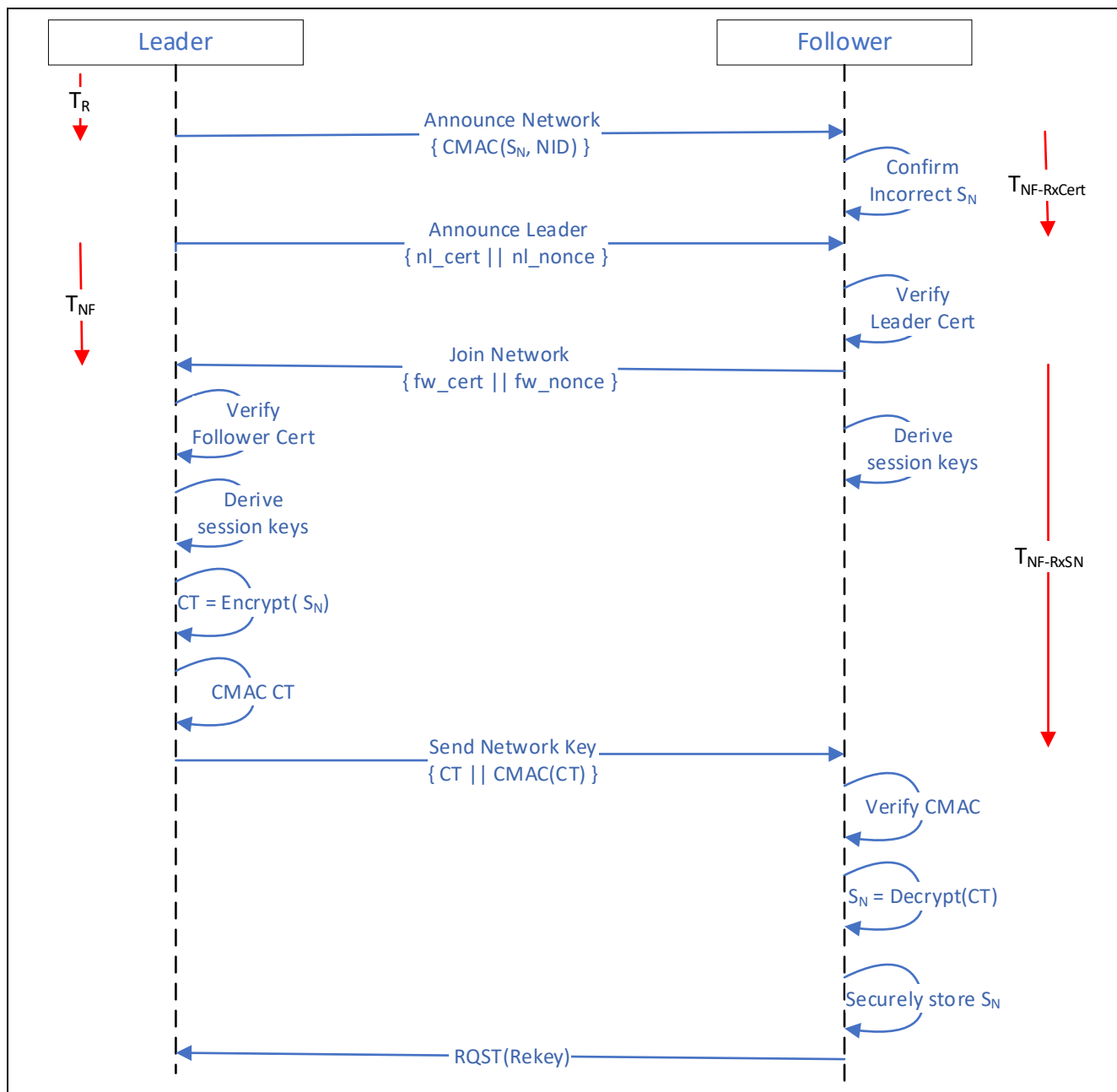


Figure 23 – Network Formation Sequence

12.3.1 Transition Flags

The detection of a S_N mismatch is only one of the conditions required to transition to Network Formation, (3). Other considerations are the operational state of the network and the Network Leader's capability to simultaneously process both (E)'s secure messages and (N)'s cryptography. The network requires a method to retain the requirement for Network Formation when it has been postponed. The network also requires a method to postpone Network Formation until it is safe to perform. Examples may be an extended power hold at shutdown when the network is quiet or a maintenance mode that can allow the Follower to join safely.

12.3.2 Generation of Network Key (S_N)

The generation of S_N happens in a Network Leader, before Network Formation, on its initial power cycle. This is typically a once in lifetime event for a Network Leader. Depending on the software build environment and/or the technology used (NAND or NOR FLASH), the Network Leader's initial value for S_N will be 0_{128} or $2^{128}-1$ (128 bits, all 1's). When the Network

Leader detects either of these two conditions, it shall generate a new 128-bit S_N from a true random number generator (TRNG) or a Pseudo Random Number Generator (PRNG) seeded from a TRNG.

The Network Leader and all subsequent Followers must keep S_N secure, preferably in an HPSE. If stored outside of an HPSE it may be wrapped using an authenticating encryption mode using an HPSE. See *Send Network Key* as an example.

A network Follower that detects S_N equal to 0_{128} or $2^{128}-1$ (128 bits, all 1's) shall use 0_{128} so that it may communicate with other uninitialized devices during network assembly.

12.4 NF Requirements

Network Formation is the process of the Network Leader confidentially distributing the network key S_N to Followers. At the end of Network Formation, all members have the same network key; this is what makes them members. Typically, a member goes through the Network Formation process only once in its service life, either during integration at the factory by the OEM, or as a service part being introduced to the network.

1. The initial Network Formation (typically at the OEM's assembly factory with a network of integrated members) should take no more than 1 minute. The intent is that adding security to an OEM's product will not cause production delays in their assembly factories.
2. Adding a member to an existing secure network shall take less than 1 minute.
3. Each member shall have a unique, device specific public/private key pair for X25519 key agreement.
4. The Network Leader shall be designated at network design-time, as opposed to a dynamic scheme where any member node can be elected the Network Leader.
5. All Followers shall start with $S_N = 0_{128}$ as a temporary key on their first power cycle. The intent of this initial key is to allow controllers that have not yet joined a secure network the ability to send messages "securely" in the absence of a Network Leader. This is to simplify the member's code by avoiding the need to create two separate branches: one for communicating before getting S_N and the other after. Clearly using 0_{128} for S_N is not secure, but as this occurs only before the Network Leader is available (such as at a sub-assembly area in a factory) this is appropriate for secure environments.
6. The temporary S_N shall only be used until S_N is established and available to the member.
7. Replacing temporary S_N with the S_N provided by the Network Leader should occur without the need of a power cycle.
8. Once a member starts using an S_N that replaced the temporary value, there is no provision to "go back" to using the temporary value.
9. A Follower that has the opportunity to participate in Network Formation shall do so.
10. A Follower that fails out of Network Formation (e.g., due to a NID mismatch with the Network Leader) shall not exchange secure messages.
11. A Network Leader shall generate a new S_N if it detects S_N is equal to 0_{128} or $2^{128}-1$ (128 bits, all 1's) or fails verification for non-HPSE storage.
12. The network key, S_N , shall be generated by the Network Leader.
13. Sufficient entropy shall be used to make S_N effectively random. The network key, S_N , is to be mathematically unrelated to information that describes the Network Leader (for example, do not use the part number or serial number to generate the network key).
14. A Network Leader shall set a fault whenever S_N has been replaced.
15. A Follower shall form a network with a Network Leader only when the NID certificate provided by the Network Leader is valid and the NID field in the Follower's and Network Leader's NID certificates match.

16. A Network Leader shall form a network with a Follower only when the NID certificate provided by the Follower is valid and the NID field in the Follower's and Network Leader's NID certificates match.
17. The NID field in the NID certificate shall be unique for each network. There is no generic or universal NID, nor a NID re-used by a product family.
18. A member ECU shall have an X25519 key-agreement key, where the private key, K_{FW-KA} or K_{NL-KA} for Follower(s) and Network Leader, respectively, is secured by the member and the public key, P_{FW-KA} or P_{NL-KA} for Follower(s) and Network Leader, respectively, is contained in the same X.509 NID Certificate that chains back to the Network Authority.
19. When the Network Leader shares S_N , it shall do so with confidentiality specific to the Follower; specifically, with a symmetric key derived from X25519 key agreement with the Follower.
20. The member shall authenticate that S_N was provided by an authentic and authorized Network Leader.
21. A returning member that retained S_N may use it.
22. A returning member that has not retained S_N shall use the Network Formation process to obtain it.
23. Once a Network Leader has completed Network Formation with Follower(s), any subsequent new Follower(s) shall get the same S_N .
24. The replacement of the Network Leader shall require the Network Formation process to share S_N with Followers.
25. Member ECUs shall store S_N securely.
 - a. Securely shall mean that a successful attack on S_N required an adversary to remove the ECU from the network and perform a hardware level attack (that is, not an on-line attack over a network, and not the insertion of a USB, or other like dongle, that provides root access).
 - b. ECUs shall require a password to enable their debug port.
 - c. To protect S_N , ECUs should implement the highest level of security from this list that that they are capable of; where the first entry offers the highest security.
 - i. S_N stored in HPSE.
 - ii. S_N stored in internal flash and protected by HPSE.
 - iii. S_N stored in internal flash and protected by hardware-based security.
 - iv. S_N stored in external flash and protected by hardware-based security.
 - v. Where protected means encryption of the key for confidentiality and a CMAC of encrypted value to provide tamper-evidence.
26. Non-Members shall not be allowed to participate in secure network communications on the network.
27. An ECU that fails out of the Network Formation process shall set a fault.
28. Once a member (other than the leader) discovers that it needs to perform Network Formation, it shall stop sending secure messages and not resume until it has a new S_N and S_S . Secure messages waiting to be transmitted may be canceled, but is not required.

13. EXCHANGING MESSAGES

13.1 Structure of Nonce

The nonce used in secure message exchanges is composed of data specific to the message being sent. The creation of the nonce is shown in Figure 24. The nonce is computed as:

$$\text{nonce} = ((E \ll 23) \text{ OR'ed PGN}) \parallel \text{SA} \parallel \text{FV}$$

where,

E = the encryption status,

PGN = the SAE J1939 PGN of the message data,

SA = the SAE J1939 source address of the message sender,

FV = freshness value to prevent replay attacks.

As described in SAE J1939-22, section 6.1.3, when a PGN is expressed as a 24-bit value, the 6 most significant bits are set to zero. To keep the nonce components on even byte boundaries, the E bit will overwrite the most significant bit of the PGN.

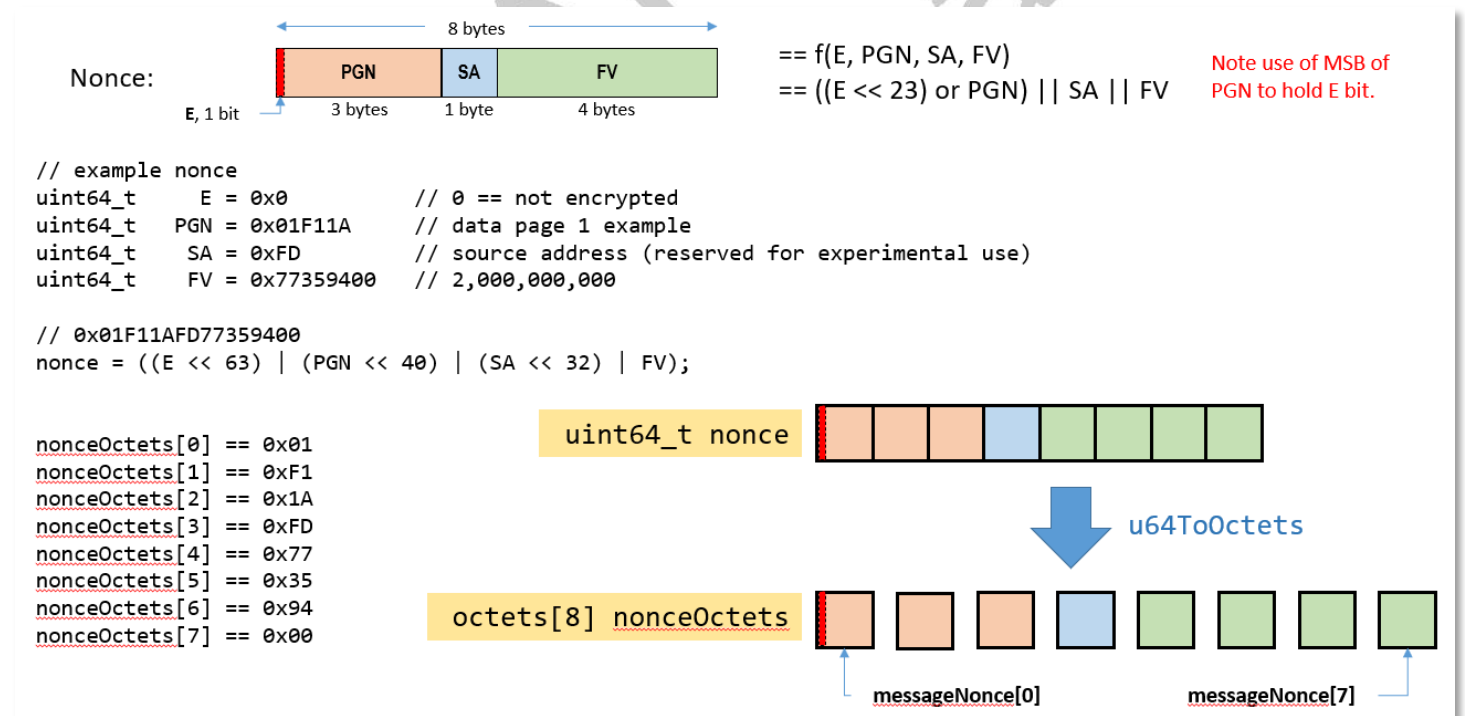


Figure 24 – Nonce Construction

13.2 Transmitting

After rekeying (T_R), the transmitter calculates the new S_S for all messages it transmits during this epoch. The process flow for the transmitting of messages is shown in Figure 25. The freshness value is incremented. If the encryption bit is set, then the plain text is encrypted as per Table 11 and the cipher text (CT) replaces PT to become the new Data. A nonce, created from the E, PGN, SA and FV, is concatenated to the Data and used with S_S to create the Tag_CMIC. See Table 10. Tag_CMIC is then truncated to form Tag which is then recombined with E, FV and Data for transmission.

If the new FV value is larger than the maximum allowed value ($RV_REKEY_TRIGGER$), a flag is set and the device should request a rekey at its earliest opportunity.

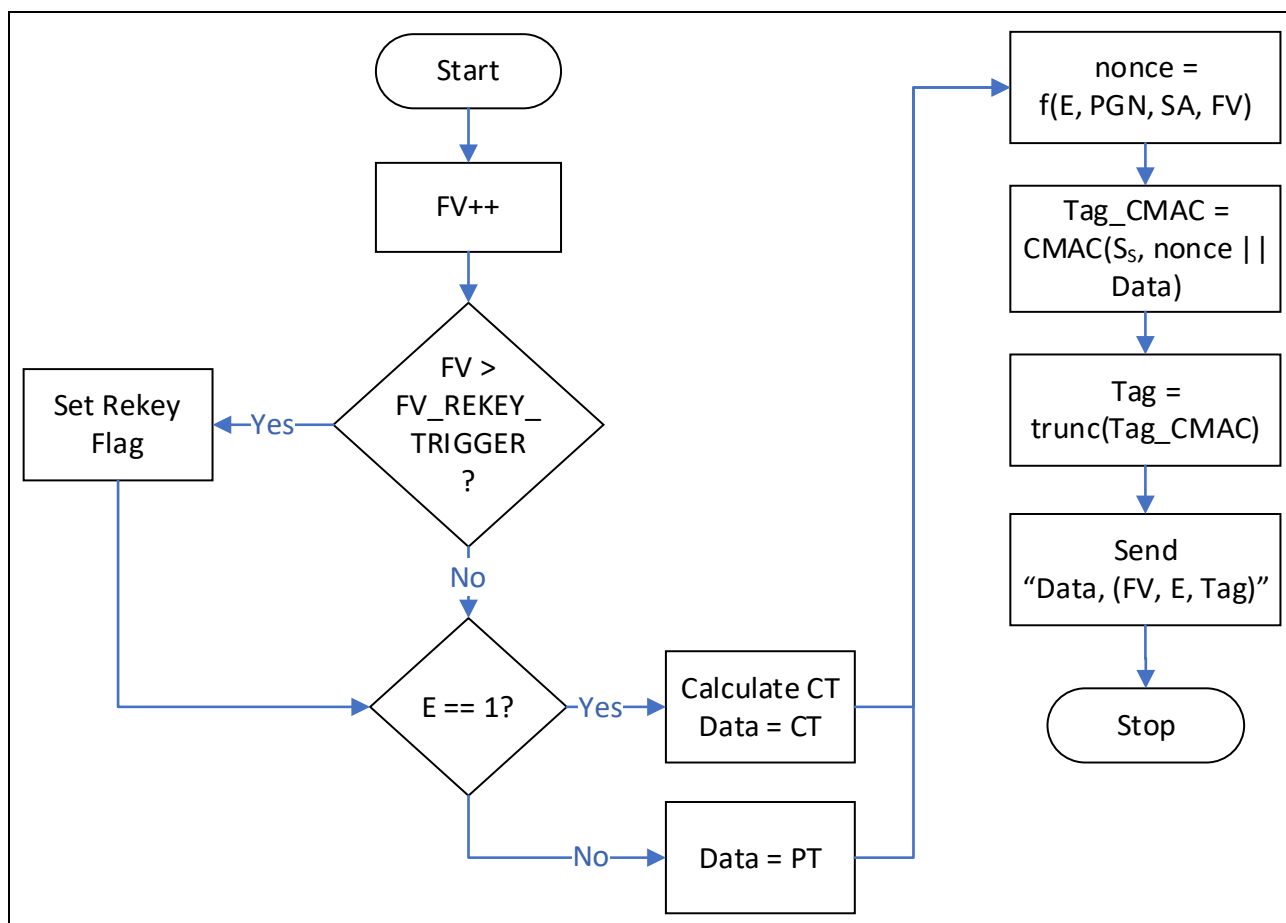


Figure 25 – Transmit Messages Process Flow

Table 10 – Transmitter Tag Calculations

Term	Definition	Length, bits
Tag_CMAC	CMAC(S_{S-CMAC} , nonce PT)	128
Tag	most significant bits of Tag_CMAC	31
nonce	see section 13.1	64

Table 11 – Transmitter Ciphertext Calculations

Term	Definition	Length, bits
CT	AES-128-CTR(S_{S-ENC} , PT, IV)	[0, MaxMS-64]
IV	initialization vector: nonce block count	128
nonce	see section 13.1	64
block count	init = 0; increment by 1 for each subsequent block	64

13.3 Receiving

SecOC/E allows for a transition between two network keys as a result of rekeying. The flow in Figure 26 shows the steps of receiving, including verifying the CMAC tag, and checking freshness while allowing for the transition between the old S_S and new S_S . The T_{SS} timer starts immediately after the completion of the rekey timer, T_R . After T_{SS} completes, the system

will no longer accept messages with the old S_S /FV set. A subscript, Key Index (0 or 1), has been introduced to represent which S_S /FV structure is being used.

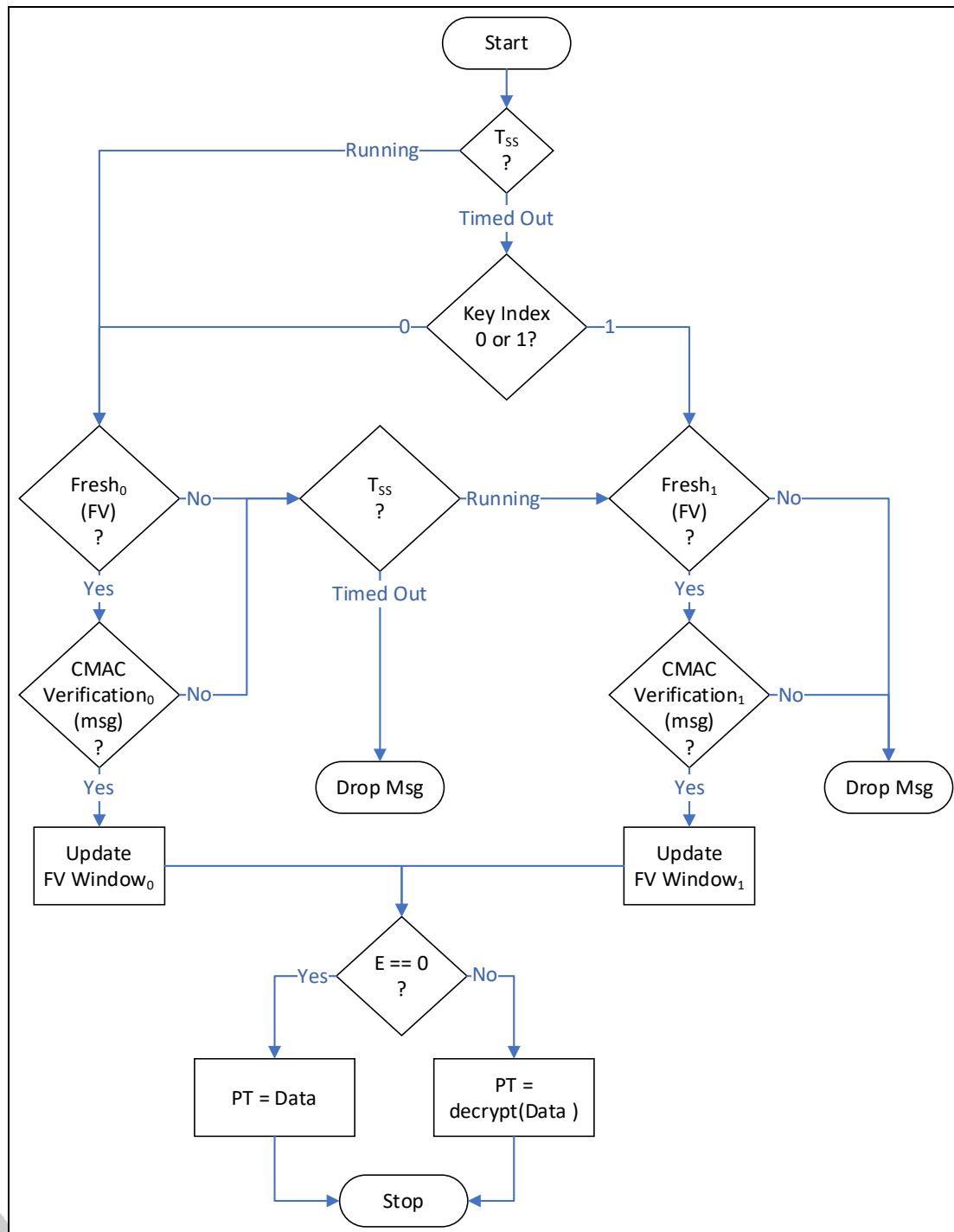


Figure 26 – Receive Messages Process Flow

Messages that fail either the freshness or CMAC verification will be dropped and the process depicted in Figure 27 shall be followed.

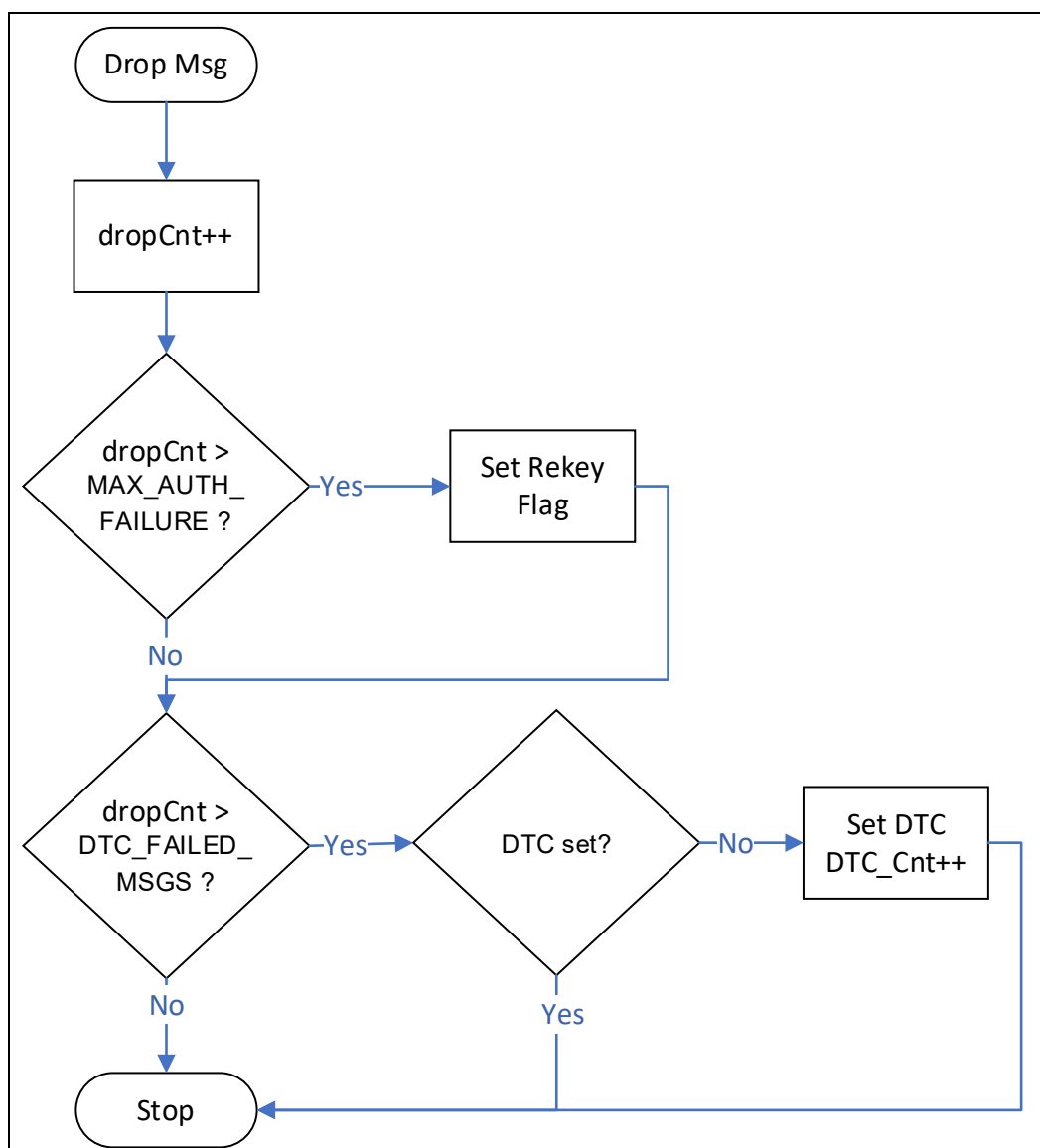


Figure 27 – Drop Message Process Flow

Verifying the truncated CMAC Tag requires the receiver to calculate a candidate Tag_CMAL value which is then truncated to produce Tag_Calc. Messages are accepted when Tag is equal to Tag_Calc and the message is fresh. The terms used for calculating Tag_Calc are defined in Table 12. The terms used for decrypting received cipher text are defined in Table 13. References to Diagnostic Trouble Codes (DTC) are presented as a recommended action but are beyond the scope of this document.

Table 12 – Receiver Tag_Calc Calculations

Term	Definition	Length, bits
Tag_CMAL	CMAC(S _{S-CMAC} , nonce PT)	128
Tag_Calc	most significant bits of Tag_CMAL	31
nonce	see section 13.1	64

For messages with confidentiality, the receiver needs to decrypt CT to get PT. This is done after Tag_Calc is calculated and verified equal to Tag.

Table 13 – Receiver CT Decryption

Term	Definition	Length, bits
PT	AES-128-CTR(S_{S-ENC} , CT, IV)	[0, MaxMS-64]
IV	initialization vector: nonce block count	128
nonce	see section 13.1	64
block count	init = 0; increment by 1 for each subsequent block	64

13.4 Unsecured Message

Messages without security requirements are sent as plain text. When the Application Protocol Data Unit (A_PDU) contains SAE J1939 Parameter Group (PG) data, it is sent in a Multi-PG using an SAE J1939 PG Type of Service (e.g., TOS 1 or 2) and applicable Trailer Format.

13.5 Exchanging Message Requirements

- Transmitters and receivers of secure messages shall have an agreed upon a session key, S_s .
- Transmitters and receivers of secure messages shall have a method to transition between two values of S_s .
- During a transition, they shall accept up to SLIDING_WINDOW_SIZE messages from the old session key until the end of the S_s transition, T_{SS} .
- Authenticity, freshness and integrity evidence shall appear in each secure message.
- Authenticity, freshness and integrity evidence shall take the form of an AES-128 CMAC, otherwise known as the tag.
- For secure messages without confidentiality, Tag shall be calculated using the plaintext data and a nonce, using the CMAC key S_{S-CMAC} .
- All symmetric cryptography shall use a nonce, see section 13.1.
- For secure messages with confidentiality, Tag shall be calculated using the ciphertext data and an initialization vector, using the CMAC key S_{S-CMAC} .
- The CT initialization vector shall be nonce || block count.
- Block_count shall start at 0₆₄.
- Block_count shall increment with each block encrypted.
- The ciphertext shall be created using AES-128 CTR mode with the key S_{S-ENC} .
- A 31-bit truncated form of Tag_CMAC, called Tag, shall appear in the secure message.
- A receiving member shall calculate a Tag_CMAC based on the information it received using the same formula that the transmitter used.
- A receiving member shall calculate a truncated version of Tag_CMAC, which is called Tag_Calc.
- A receiving member shall discard a message when its Tag_Calc not equal to Tag.
- To avoid creating an explicit oracle, each receiving member shall quietly count all discarded message events and not directly share the count.
Note: An oracle is any system which can give some extra information on a system, which otherwise would not be available. A cryptographic oracle is a service that performs encryption and/or decryption on behalf of an attacker. It allows an attacker to read from or write into an encrypted system without knowledge of the encryption key. This effectively destroys the confidentiality and integrity assurances intended by the encryption. Therefore, by

discarding message counts, the only known way to tamper with the data is to provide a valid tag, and without the keys there is no way other than luck (1 in 2³¹ chance).

18. It is recommended that a receiving member should count and may log all occurrences of invalid messages – where invalid means stale or replayed FV, or incorrect Tag.
19. A receiving member shall trigger a rekey when the number message verification failures exceed MAX_AUTH_FAILURE. Transmitters and receivers shall have a priori knowledge that communications using authentication, or communications using both authentication and confidentiality, are necessary.
20. SecOC/E messages shall set an E bit to distinguish when the data is encrypted.

14. MESSAGE DEFINITIONS

The following messages are event triggered and form the control plane for the SAE J1939-91C protocol. The nonces and CMACs are in MSB-first order. All messages have a common header (Channel, Version and Reserved byte) for future growth. These parameters have been included in Version 1 to minimize future message format changes and will be defined in later versions of the protocol.

The FV, E, and Tag only apply to C-PGs inside a Multi-PG service message, and they cannot apply to FD Transport service messages.

The FV, E, and Tag only appear as part of an assurance data trailer for a C-PG in a Multi-PG service message.

14.1 SecOC/E Announce Leader PG (PGN 64002)

The SecOC/E Announce Leader PG is used by Network Leader to share its NID certificate and its nonce with one or more Followers that are going through Network Formation. This message may be used by the Network Leader at the end of the Rekey Timer (T_R) to announce that it is the Network Leader and to identify the network to all potential Followers. A randomly chosen 128-bit nonce is added to prevent replays. The length of the PG data varies with the length of the Leader Certificate and is typically about 821 bytes. The PG definition is shown in Figure 28.

Parameter Group Name:		SecOC/E Announce Leader (<i>AnnounceLeader</i>)			
Transmission repetition rate:		As needed			
Data length:		Variable			
Extended Date Page:		0			
Data Page:		0			
PDU Format:		FA _h			
PDU Specific:		02 _h			
Default priority:		7			
Parameter Group Number:		64002 (00FA02 _h)			
Data Field:					
Byte:	1 to 2	SecOC/E Announce Leader Channel	SPN 23123	See 14.1.1	
	3	SecOC/E Leader Protocol Version	SPN 23124	See 14.1.2	
	4	SAE Reserved			
	5 to 20	SecOC/E Leader Nonce	SPN 23125	See 14.1.3	
	21 to n	SecOC/E Leader Certificate	SPN 23126	See 14.1.3	

Figure 28 – SecOC/E Announce Leader Message

14.1.1 SecOC/E Announce Leader Channel (SPN 23123)

The channel number facilitates future multi-channel secure networks. Multi-channel networks require more sophisticated authentication processing that is beyond the scope of this document's initial release. The default value shall be 0 which indicates no channel assigned.

0 = No channel assigned

1 to 65535 = Reserved for SAE assignment

Data Length: 2 bytes
Resolution: 65536 states
Data Range: 0 to 65535
Type: Status
PGN reference: 64002

14.1.2 SecOC/E Leader Protocol Version (SPN 23124)

The Protocol Version represents the message's format and content and shall be set to 1. Messages with a value other than 1 shall be ignored.

0 = Reserved for SAE assignment

1 = Version 1

2 to 255 = Reserved for SAE assignment

Data Length: 1 byte
Resolution: 1 version / bit
Data Range: 0 to 255
Type: Status
PGN reference: 64002

14.1.3 SecOC/E Leader Nonce (SPN 23125)

The nonce field is a 16-byte (128-bit) random number. This number shall be generated from the random number generator within the Leader.

Data Length: 16 bytes
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 64002

14.1.4 SecOC/E Leader Certificate (SPN 23126)

The Leader NID certificate field contains the Network Leader's NID certificate. The length of the certificate is typically about 800 bytes. See section 9.3 for a definition of the NID certificate structure.

Data Length: Variable
Resolution: see 9.3
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 64002

14.2 SecOC/E Join Network PG (PGN 64003)

The SecOC/E Join Network PG is used by a network Follower to share its NID certificate and its nonce with the Network Leader to complete Network Formation. The message is used by a network Follower to show that it is on the correct network and wishes to join in secure message exchanges. If a Follower realizes that it is on the wrong network (i.e., has a different

NID in its certificate), it raises a fault and drops out of the SecOC/E protocols. The length of the PG data varies with the size of the Follower Certificate and typically about 821 bytes. The PG definition is shown in Figure 29.

Parameter Group Name		SecOC/E Join Network (<i>JoinNetwork</i>)		
Transmission repetition rate:		As needed		
Data length:		Variable		
Extended Date Page:		0		
Data Page:		0		
PDU Format:		FA _h		
PDU Specific:		03 _h		
Default priority:		7		
Parameter Group Number:		64003 (00FA03 _h)		
Data Field:				
Byte:	1 to 2	SecOC/E Join Network Channel	SPN 23127	See 14.2.1
	3	SecOC/E Join Network Protocol Version	SPN 23128	See 14.2.2
	4	SAE Reserved		
	5 to 20	SecOC/E Follower Nonce	SPN 23129	See 14.2.3
	21 to n	SecOC/E Follower Certificate	SPN 23130	See 14.2.3

Figure 29 – SecOC/E Join Network Message

14.2.1 SecOC/E Join Network Channel (SPN 23127)

The channel number facilitates future multi-channel secure networks. The default value shall be 0 which indicates no channel assigned.

0 = No channel assigned

1 to 65535 = Reserved for SAE assignment

Data Length: 2 bytes
 Resolution: 65536 states
 Data Range: 0 to 65535
 Type: Status
 PGN reference: 64003

14.2.2 SecOC/E Join Network Protocol Version (SPN 23128)

The Protocol Version represents the message's format and content and shall be set to 1. Messages with a value other than 1 shall be ignored.

0 = Reserved for SAE assignment

1 = Version 1

2 to 255 = Reserved for SAE assignment

Data Length: 1 byte
 Resolution: 1 version / bit
 Data Range: 0 to 255
 Type: Status
 PGN reference: 64003

14.2.3 SecOC/E Follower Nonce (SPN 23129)

The nonce field is a 16-byte (128-bit) random number to prevent replay attacks. This number shall be generated from the random number generator within the Follower.

Data Length: 16 bytes
 Data Range: 0 to 255 per byte
 Type: Status
 PGN reference: 64003

14.2.4 SecOC/E Follower Certificate (SPN 23130)

The Follower NID certificate field contains the NID certificate of the responding ECU. The length of the certificate is typically about 800 bytes. See section 9.3 for a definition of the NID certificate structure.

Data Length: Variable
Resolution: see 9.3
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 64003

14.3 SecOC/E Send Network Key PG (PGN 18176)

The SecOC/E Send Network Key message is used by the Leader to securely share the network key with a Follower. The Network Leader sends a unique version of this message to each Follower that is joining the network. The *AnnounceLeader* and *JoinNetwork* messages provide enough information to derive the secrets required to encrypt and decrypt the Send Network Key message. The PG definition is shown in Figure 30.

Parameter Group Name		SecOC/E Send Network Key (<i>SendNetworkKey</i>)		
Transmission repetition rate:		As needed		
Data length:		Variable (minimum length of 36 bytes)		
Extended Data Page:		0		
Data Page:		0		
PDU Format:		47 _h		
PDU Specific:		00 _h		
Default priority:		7		
Parameter Group Number:		18176 (004700 _h)		
Data Field:				
Byte:	1 to 2	SecOC/E Send Network Key Channel	SPN 23131	See 14.3.1
	3	SecOC/E Send Network Key Protocol Version	SPN 23132	See 14.3.2
	4	SAE Reserved		
	5 to 20	SecOC/E Encrypted S _N	SPN 23133	See 14.3.3
	21 to 36	SecOC/E Encrypted S _N CMAC	SPN 23134	See 14.4.4

Figure 30 – SecOC/E Send Network Key Message

14.3.1 SecOC/E Send Network Key Channel (SPN 23131)

The channel number facilitates future multi-channel secure networks. The default value shall be 0 which indicates no channel assigned.

- 0 = No channel assigned
- 1 to 65535 = Reserved for SAE assignment

Data Length: 2 bytes
Resolution: 65536 states
Data Range: 0 to 65535
Type: Status
PGN reference: 18176

14.3.2 SecOC/E Send Network Key Protocol Version (SPN 23132)

The Protocol Version represents the message's format and content and shall be set to 1. Messages with a value other than 1 shall be ignored.

- 0 = Reserved for SAE assignment
- 1 = Version 1
- 2 to 255 = Reserved for SAE assignment

Data Length: 1 byte
Resolution: 1 version / bit
Data Range: 0 to 255
Type: Status
PGN reference: 18176

14.3.3 SecOC/E Encrypted S_N (SPN 23133)

S_N , encrypted for the Follower, using a key agreed to via x25519 and shared nonces. See section 12, in general, and section 12.1.3 for short summary. Note the use of AES-CTR mode.

- IV = SecOC/E Leader Nonce (from Announce Leader message, see 14.1)
- PT = S_N
- Key = $S_{NL-FW-ENC}$
- CT = Encrypted S_N

Data Length: 16 bytes
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 18176

14.3.4 SecOC/E Encrypted S_N CMAC (SPN 23134)

To provide authenticated encryption the CMAC of CT is also sent. See section 12, in general, and section 12.1.3 for a short summary, where key = $S_{NL-FW-MAC}$.

Data Length: 16 bytes
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 18176

14.4 SecOC/E Rekey Nonce PG (PGN 64004)

The SecOC/E Rekey Nonce message is sent by a network follower. This message data is received by the Network Leader and used to establish a new session key (S_s) with the follower. It also securely shares the CA's understanding of the NID. The PG definition is shown in Figure 31.

Parameter Group Name:		SecOC/E Rekey Nonce (Rekey)			
Transmission repetition rate:		As needed and On Request			
Data length:		Variable (minimum length of 36 bytes)			
Extended Data Page:		0			
Data Page:		0			
PDU Format:		FA _h			
PDU Specific:		04 _h			
Default priority:		7			
Parameter Group Number:		64004 (00FA04 _h)			
Data Field:					
Byte:	1 to 2	SecOC/E Rekey Channel	SPN 23135	See 14.4.1	
	3	SecOC/E Rekey Protocol Version	SPN 23136	See 14.4.2	
	4	SAE Reserved			
	5 to 20	SecOC/E Rekey Nonce	SPN 23137	See 14.4.3	
	21 to 36	SecOC/E Member NID CMAC	SPN 23138	See 14.4.3	

Figure 31 – SecOC/E Rekey Nonce Message

14.4.1 SecOC/E Rekey Channel (SPN 23135)

The channel number facilitates future multi-channel secure networks. The default value shall be 0 which indicates no channel assigned.

0 = No channel assigned

1 to 65535 = Reserved for SAE assignment

Data Length: 2 bytes
 Resolution: 65536 states
 Data Range: 0 to 65535
 Type: Status
 PGN reference: 64004

14.4.2 SecOC/E Rekey Protocol Version (SPN 23136)

The Protocol Version represents the message's format and content and shall be set to 1. Messages with a value other than 1 shall be ignored.

0 = Reserved for SAE assignment

1 = Version 1

2 to 255 = Reserved for SAE assignment

Data Length: 1 byte
 Resolution: 1 version / bit
 Data Range: 0 to 255
 Type: Status
 PGN reference: 64004

14.4.3 SecOC/E Rekey Nonce (SPN 23137)

The nonce field is a 16-byte (128-bit) random number. This number shall be generated from the random number generator within the ECU. See section 13.1. See section 11.1.3.

Data Length: 16 bytes
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 64004

14.4.4 SecOC/E Member NID CMAC (SPN 23138)

To share the CA's understanding of S_N , without sharing the value of S_N , the App calculates the AES-128 CMAC of NID, where key = S_N . See section 11.1.3.

Data Length: 16 bytes
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 64004

14.5 SecOC/E Announce Network Message (PGN 64005)

The SecOC/E Announce Network message is used by the Network Leader to securely confirm understanding of both S_N and NID with all Followers. The PG definition is shown in Figure 32.

Parameter Group Name:		SecOC/E Announce Network (<i>AnnounceNetwork</i>)			
Transmission repetition rate:		As needed			
Data length:		Variable (minimum length of 20 bytes)			
Extended Data Page:		0			
Data Page:		0			
PDU Format:		FA _h			
PDU Specific:		05 _h			
Default priority:		7			
Parameter Group Number:		64005 (00FA05 _h)			
Data Field:					
SPN Byte:	1	SecOC/E Announce Network Channel	SPN 23139	See 14.5.1	
	3	SecOC/E Announce Network Protocol Version	SPN 23140	See 14.5.2	
	4	SAE Reserved			
	5 to 20	SecOC/E Leader NID CMAC	SPN 23141	See 14.5.3	

Figure 32 – SecOC/E Announce Network Message

14.5.1 SecOC/E Announce Network Channel (SPN 23139)

The channel number facilitates future multi-channel secure networks. The default value shall be 0 which indicates no channel assigned.

- 0 = No channel assigned
- 1 to 65535 = Reserved for SAE assignment

Data Length: 2 bytes
Resolution: 65536 states
Data Range: 0 to 65535
Type: Status
PGN reference: 64005

14.5.2 SecOC/E Announce Network Protocol Version (SPN 23140)

The Protocol Version represents the message's format and content and shall be set to 1. Messages with a value other than 1 shall be ignored.

- 0 = Reserved for SAE assignment
- 1 = Version 1
- 2 to 255 = Reserved for SAE assignment

Data Length: 1 byte
Resolution: 1 version / bit
Data Range: 0 to 255
Type: Status
PGN reference: 64005

14.5.3 SecOC/E Leader NID CMAC (SPN 23141)

To share the Leader's understanding of S_N , without sharing the value of S_N the Network Leader calculates the AES-128 CMAC of NID, where key = S_N . See section 11.1.6.

Data Length: 16 bytes
Data Range: 0 to 255 per byte
Type: Status
PGN reference: 64005

15. PROTOCOL LIMITATIONS

15.1 Classic CAN

This protocol was developed for the SAE J1939-22 CAN FD assurance data trailer and does not support Classic CAN.

15.2 Functional Safety

While Network Security and Functional Safety are related, SAE makes no claim to Functional Safety levels achievable through the SAE J1939-91C Network Security interface. Users are responsible for analysis to determine the functional safety level (e.g., IEC 61508, ISO 26262) that their architecture may require, and any Functional Safety benefit that SAE J1939-91C may provide.

15.3 Multiple Network Leaders

This standard assumes that each secure network is designed to have a single Network Leader. If the implementation requires multiple Network Leader capable devices (redundancy), an arbitration scheme must be developed to ensure only one active Network Leader at a time.

APPENDIX A - REKEYING TIMING DIAGRAMS

This section has a collection of scenarios related to rekey, including race conditions.

A.1 SIMPLEST PATH

Figure A1 represents the simplest path where rekey not related to power on and all members receive the *RQST(Rekey)* message.

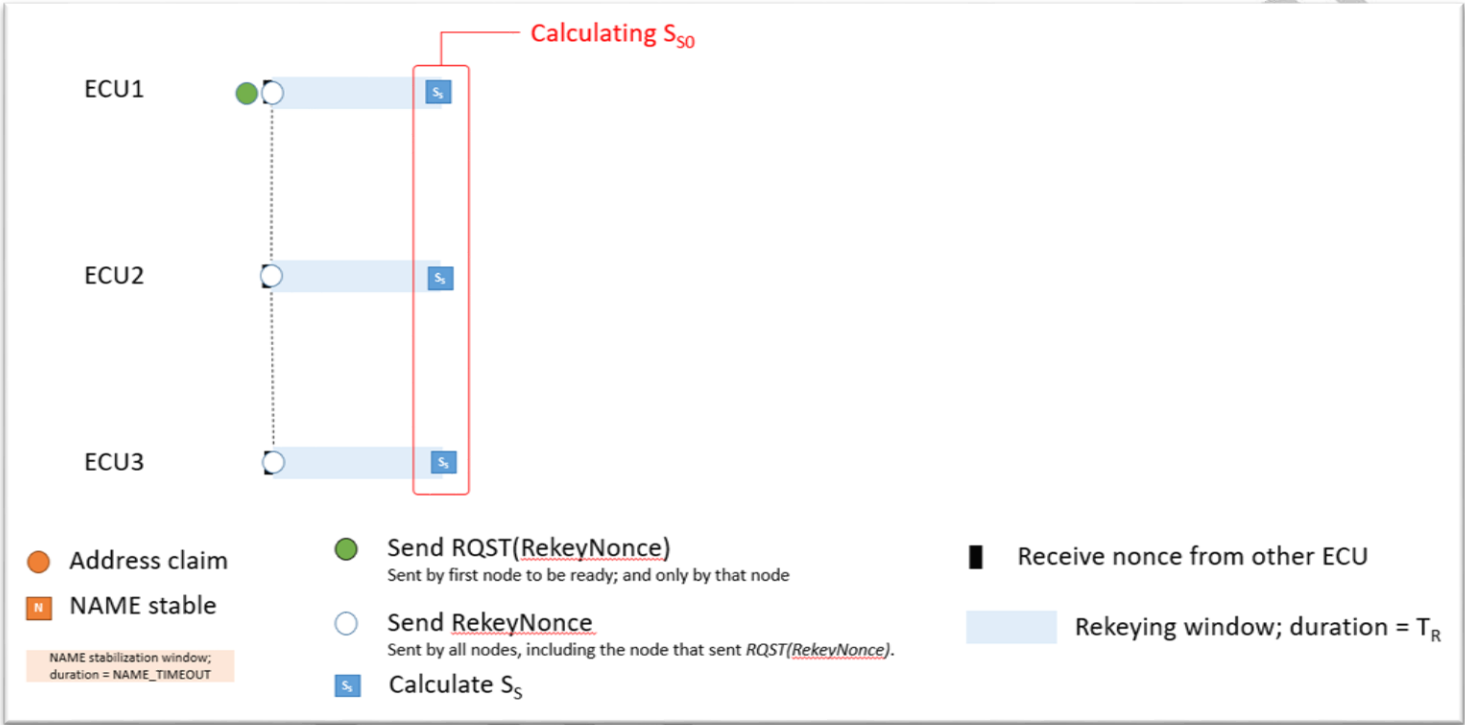


Figure A1 - Rekey Simplest Case

A.2 POWER ON, BEST CASE

Figure A2 illustrates the ideal power on case where all members see the *RQST(Rekey)* message.

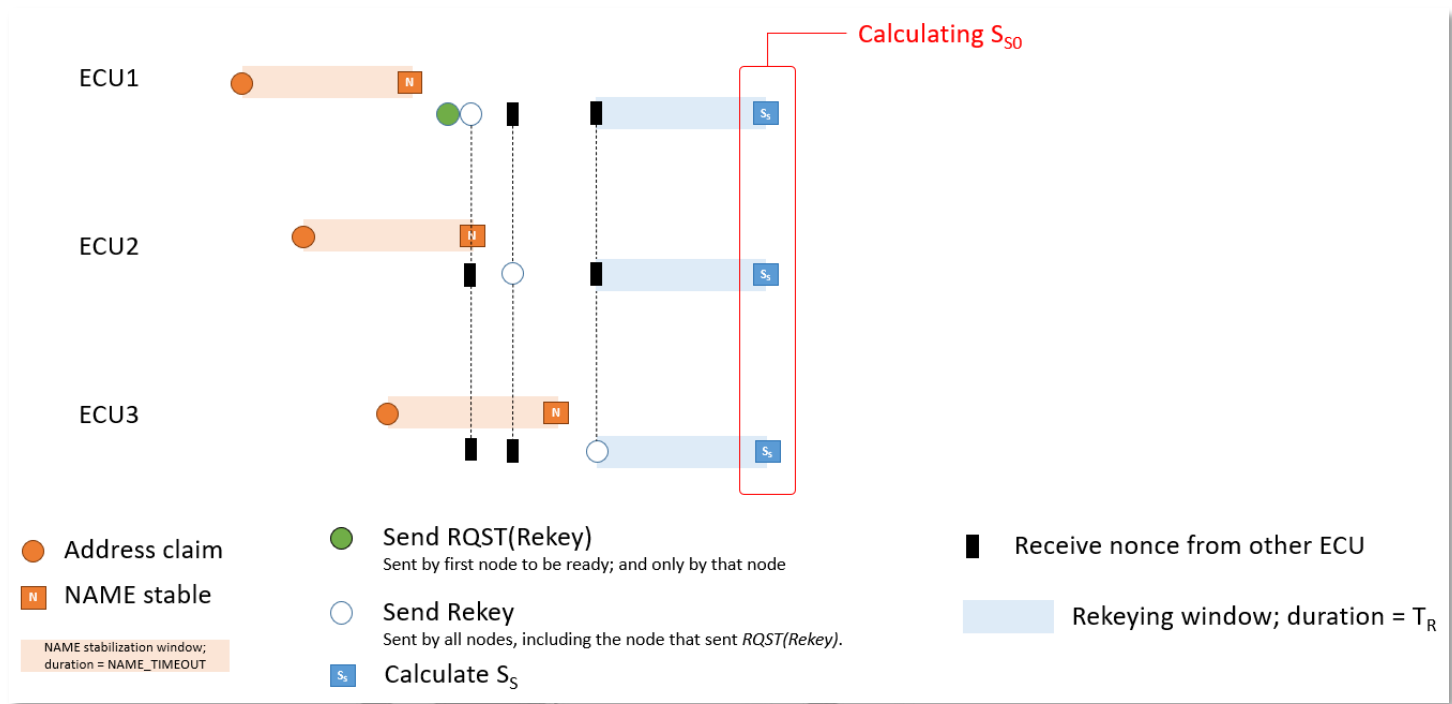


Figure A2 - Rekey Power On Best Case

A.3 POWER ON, RACE CONDITION

Figure A3 is an example where a race condition leads to two RQST(Rekey) messages. In this example, ECU2 didn't see ECU1's RQST(Rekey) message, so it sent its own RQST(Rekey). ECU1 and ECU2 correctly handle this by repeating their nonce and not generating new nonces. ECU3 correctly handles the repeated nonce as well.

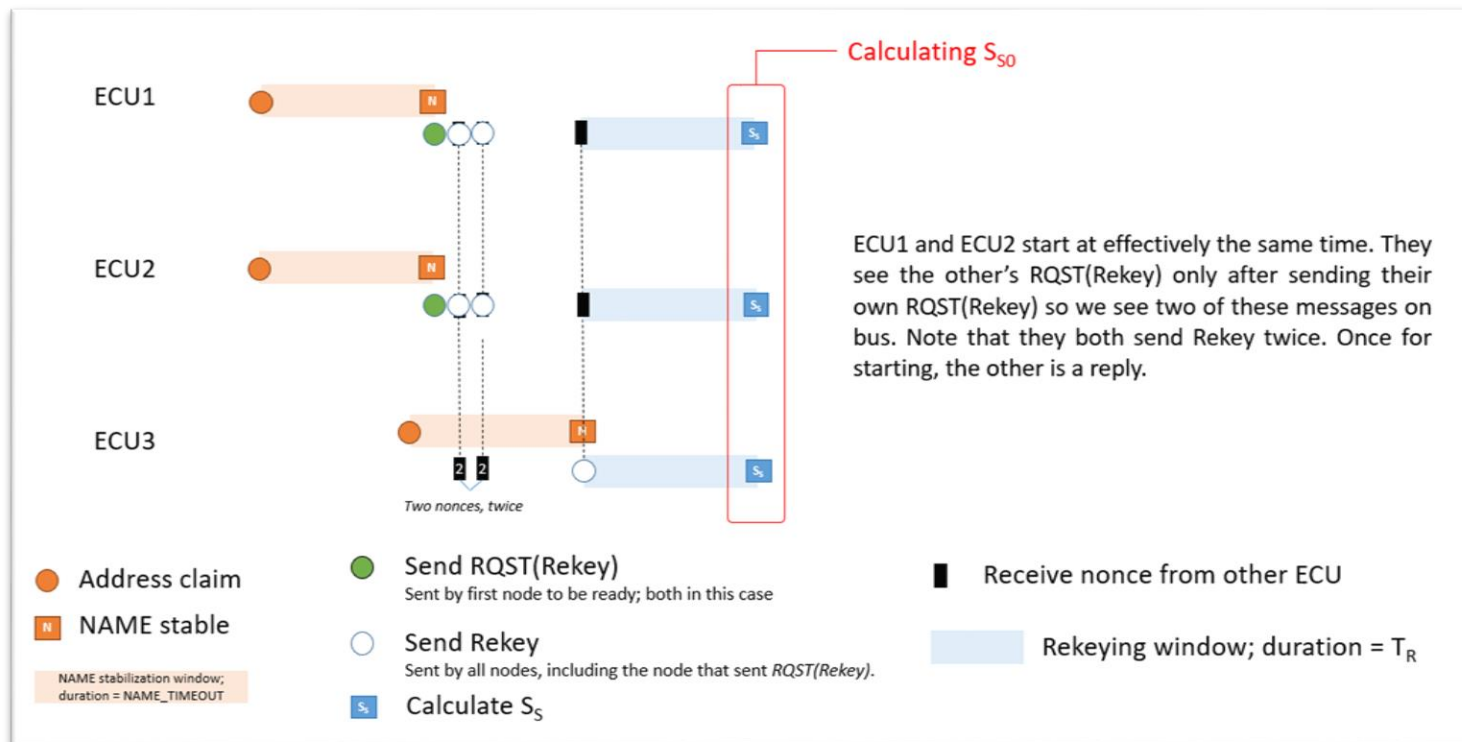


Figure A3 - Rekey with Power On Race Conditions

A.4 POWER ON WITH LATE MEMBER

Figure A4 is an example of a late to start ECU5 does not see the first *RQST(Rekey)* message.

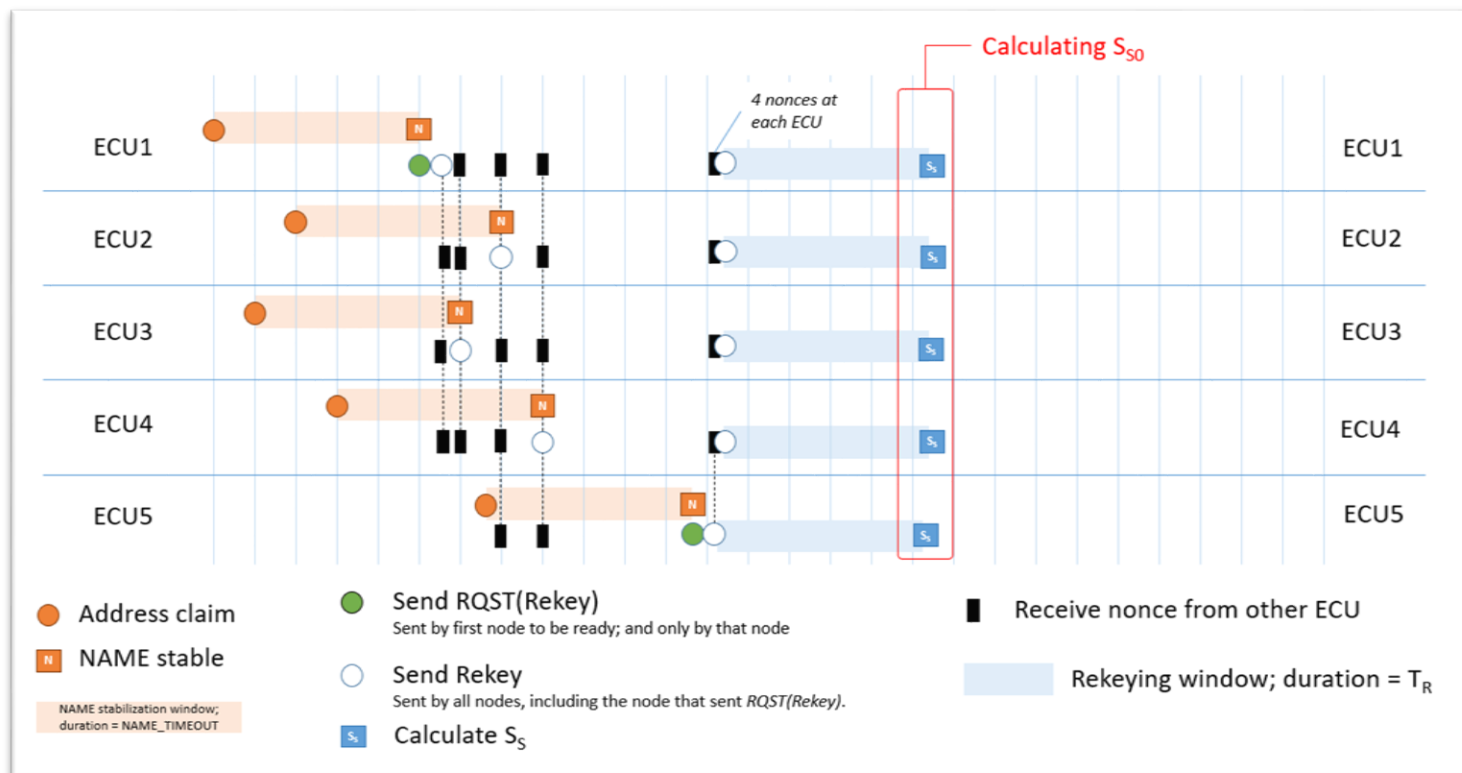


Figure A4 - Rekey Power On with Late Member

A.5 POWER ON WITH VERY LATE MEMBER

Figure A5 is an example of an ECU that was so late to start that it completely missed the rekey process. It causes a new rekey process and the calculation of a new S_s .

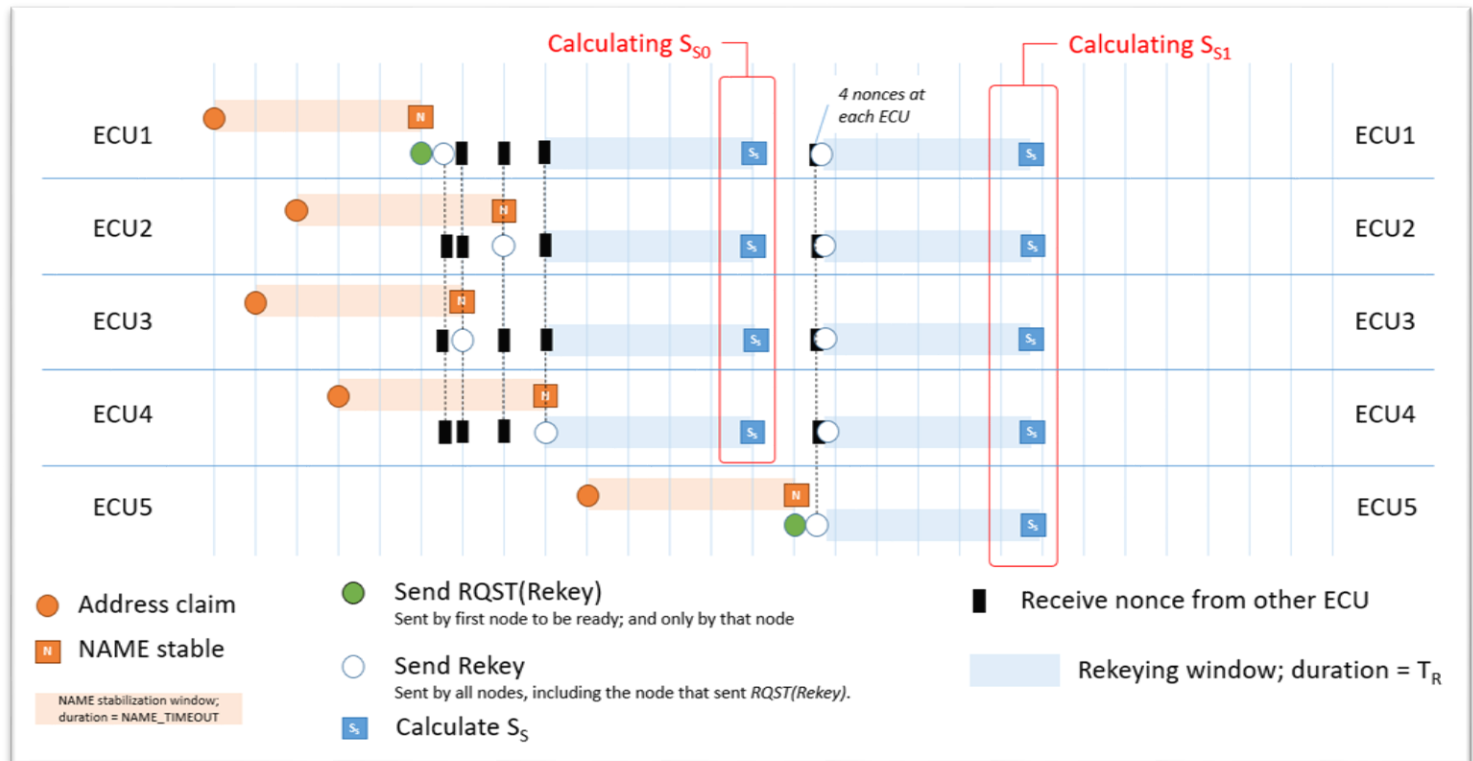


Figure A5 - Power On missed Rekey

APPENDIX B – TECHNICAL RATIONALES

The following sections expand on the rationale used to determine the size of various parameters.

B.1 FV SIZE

Rekeying is required to avoid FV overflowing and repeating values for an instance of S_s . The number of messages between FV_INIT and FV_REKEY_TRIGGER is known as an epoch. The duration of an epoch is determined by the message rate and the number of bits in FV. Message transmissions will continue during the rekeying process so FV must have capacity to increase without rolling over. The selected scheme is to use the most significant bit as the rekey trigger. For a 16-bit FV that will mean 2^{15} (32,768) messages/epoch and a 32-bit FV will have 2^{31} (2,147,483,648) messages/epoch.

Using a nominal and worst-case message rate provides the rationale for selecting FV size. A 100 messages/second rate reflects a transmitter with just a few messages to send securely. A 5,000 message/second rate reflects a transmitter that is filling a CAN FD 500k/2M bus at approximately 100% utilization with 16-byte messages (8 bytes of data, 8 bytes of security overhead). Table B1 shows the relative durations for combinations of FV size and message rates. A 16-bit FV will result in too many rekeys so a 32-bit FV will be used.

Table B1 - Epoch Durations for varying FV Size and Message Rates

	16-Bit FV		32-Bit FV	
Duration	100 msg / sec	5000 msg / sec	100 msg / sec	5000 msg / sec
Seconds	328	6.6	21,474,836	429,497
Hours	0.1	0.0	5,965	119.3
Days	0.0	0.0	249	5.0

Long form of the calculation:

$$2^{31} \frac{\text{msg}}{\text{epoch}} \times \frac{1 \text{ sec}}{5000 \text{ msg}} = 429,497 \frac{\text{sec}}{\text{epoch}} \times \frac{1 \text{ hr}}{3600 \text{ sec}} = 119.3 \frac{\text{hr}}{\text{epoch}} \times \frac{1 \text{ day}}{24 \text{ hr}} = 5.0 \frac{\text{day}}{\text{epoch}}$$

B.2 NONCE SIZE

The nonce needs to be large enough to make first or second pre-image attacks infeasible, but not to resist birthday-bound attacks. 64-bits is probably sufficient, but 128-bits matches AES block size and does not consume significantly more computational resources so the random nonce is an unsigned integer of 128-bits.

B.3 TRUNCATED CMAC TAG SIZE

This standard specifies the use of a CMAC to protect the contents of individual packets. This section references NIST 800-38B for the justification of the 31-bit tag length. NIST 800-38B chooses the length of tag in section 6.2, step 7 of the algorithm. It refers to the tag length as T_{len} in Appendix A.2 "Selection of the MAC Length" where we find this guidance:

Larger values of T_{len} provide greater assurance against guessing attacks. The performance tradeoff is that larger values of T_{len} require more bandwidth/storage for the MAC.

For most applications, a value for T_{len} that is at least 64 should provide sufficient protection against guessing attacks. A value of T_{len} that is less than 64 shall only be used in conjunction with a careful analysis of the risks of accepting an inauthentic message as authentic.

In particular, a value of T_{len} smaller than 64 should not be used unless the controlling protocol or system sufficiently restricts the number of times that the verification process can return INVALID, across all implementations with any given key. For example, the short duration of a session or, more generally, the low bandwidth of the communication channel may preclude many repeated trials.

This guidance can be quantified in terms of the following two bounds: 1) the highest acceptable probability for an inauthentic message to pass the verification process, and 2) a limit on the number of times that the output is the error message INVALID before the key is retired, across all implementations of the verification process for the key. Given estimates of these quantities, denoted Risk and MaxInvalids, respectively, T_{len} should satisfy the following inequality:

$$Tlen \geq \lg(\text{MaxInvalids} / \text{Risk})$$

For example, suppose that the MAC verification process(es) within a system will not output INVALID for more than 1024 messages before the key is retired (i.e., $\text{MaxInvalids} = 2^{10}$), and that the users can tolerate about a one in a million chance that the system will accept an inauthentic message (i.e., $\text{Risk} = 2^{-20}$). In this case, any value of $Tlen$ that is greater than or equal to 30 satisfies the inequality.

Table B2 shows the relationship of 'Risk' and 'MaxInvalids'; green cells indicate where the above inequality from is satisfied. SecOC/E has a MaxInvalids value of MAX_AUTH_FAILURE.

Table B2 - Tag Length Relationships

Tlen 31		Tlen $\geq \lg(\text{MaxInvalids} / \text{Risk})$			
Risk \ MaxInvalids		64	100	256	1024
risk = $1 / 2^x =$ 2^{-x}	-20	26.0	26.6	28.0	30.0
	-21	27.0	27.6	29.0	31.0
	-22	28.0	28.6	30.0	32.0
	-23	29.0	29.6	31.0	33.0
	-24	30.0	30.6	32.0	34.0
	-25	31.0	31.6	33.0	35.0

The table tells us that a Tag, or Tlen, of 31 is aligned with an expectation of a 1 in 16 million chance of accepting an inauthentic message where the system rekeys after MAX_AUTH_FAILURE failed validations.

B.4 E-BIT

The E-bit is required so that the system can determine whether decryption to the payload must be applied. It was decided to reduce the truncated CMAC tag to 31 bits to accommodate the additional E-bit.

APPENDIX C – TEMPORARY NETWORK KEY

This section notes a security weakness that results from using a well-known temporary key, $S_N = 0_{128}$, during network assembly. An example is a smart sensor that communicates confidentially. An adversary could install a service part smart sensor and man-in-the-middle all communication to it. By isolating the sensor from the Network Formation messages the service part could use the temporary key as its S_N . The adversary then runs the rekey process with the sensor. Both the sensor and the adversary will arrive at the same S_S since they both started with $S_N = 0_{128}$. The smart sensor is totally unaware. It behaves as if it is on a factory workbench and starts to send its data using the S_S that it and the adversary agreed to. Since the adversary knows the temporary key, it can decrypt sensor packets.

Depending on how the network interacts with the sensor it may be very hard, or impossible, to make this attack work in practice. For instance, as the member(s) that expect to receive secure message from the sensor will not be able to, they might possibly be in a limp-home mode until the sensor is fixed.

APPENDIX D - TEST VECTORS

This section contains test vectors for various cryptographic operations required in this document. Figure D1 presents the vectors in JSON format as per an approach created by Google for testing crypto libraries. The Google environment is called Project Wycheproof (<https://github.com/google/wycheproof>).

The tests are divided into 5 categories:

1. Generating Nonce given PGN, SA, and FV.
2. Generating Nonce given 29-bit ID.
3. Generating Subkeys for AES-CMAC Algorithm.
4. Generating AES-CMAC with e-bit=0.
5. Generating AES-CMAC with e-bit=1.
6. SecOC/E.

Each category has its own test vectors with different input and output vectors.

```
{
  "standard": "J1939-91C",
  "Tests Categories": [ "Generate Nounce I", "Generate Nounce II", "Generate Subkeys", "AES-CMAC Tag
/ E=0", "AES-CMAC Tag / E=1", "SecOC/E" ],
  "Tests": [
    {
      "Category": "Generate Nounce I",
      "comment": "Generate Nounce Given PGN,SA & E_bit",
      "Test Vectors": [
        {
          "Test_Id": 1,
          "pgn": "01F11A",
          "sa": "FD",
          "e-bit": "0",
          "fv": "77359400",
          "result": "01F11AFD77359400"
        },
        {
          "Test_Id": 1,
          "pgn": "01F11A",
          "sa": "FD",
          "e-bit": "1",
          "fv": "77359400",
          "result": "81F11AFD77359400"
        }
      ]
    },
    {
      "Category": "Generate Nounce II",
      "comment": "Generate Nounce Given Message 29-bit ID",
      "Test Vectors": [
        {
          "Test_Id": 1,
          "Msg_ID": "19f11afd",
          "fv": "77359400",
          "e-bit": "1",
          "result": {
            "pgn": "01F11A",
            "sa": "FD",
            "nonce": "81F11AFD77359400"
          }
        }
      ]
    }
  ],
}
```

```
{
  "Test_Id": 2,
  "Msg_ID": "19f11afd",
  "fv": "77359400",
  "e-bit": "0",
  "result": {
    "pgn": "01F11A",
    "sa": "FD",
    "nonce": "01F11AFD77359400"
  }
},

{
  "Test_Id": 3,
  "Msg_ID": "18c12ec1",
  "fv": "25484012",
  "e-bit": "1",
  "result": {
    "pgn": "00c12e",
    "sa": "c1",
    "nonce": "80c12ec125484012"
  }
},

{
  "Test_Id": 4,
  "Msg_ID": "07669abc",
  "fv": "25496012",
  "e-bit": "0",
  "result": {
    "pgn": "03669a",
    "sa": "bc",
    "nonce": "03669abc25496012"
  }
}
],

{
  "Category": "Generate Subkeys",
  "comment": "Generate [K1,K2] used for calculating AES-CMAC",
  "Test Vectors": [
    {
      "Test_Id": 1,
      "Key": "2b7e151628aed2a6abf7158809cf4f3c",
      "result": {
        "AES_128_0": "7df76b0c1ab899b33e42f047b91b546f",
        "K1": "fbeed618357133667c85e08f7236a8de",
        "K2": "f7ddac306ae266ccf90bc11ee46d513b"
      }
    },
    {
      "Test_Id": 2,
      "Key": "2b7e151628aed2a6abf7158809cf4f3c",
      "result": {
        "AES_128_0": "1c67fee794e14161d94a6ea4836194d7",
        "K1": "38cffdcf29c282c3b294dd4906c329ae",
        "K2": "719ffb9e538505876529ba920d86535c"
      }
    }
  ]
},
```

```
{
  "Category": "AES-CMAC Tag / E=0",
  "comment": "Calculate AES-CMAC Tag and E_Tag for a given message with E_bit=0 ",
  "Test Vectors": [

    {
      "Test_Id": 1,
      "Msg": "27d9",
      "Key": "b151f491c4c006d1f28214aa3da9a985",
      "e-bit": "0",
      "result": {
        "Tag": "bdbbbbac982dd62b9f682618a6a604e9",
        "E_Tag": "5eddf5d6"
      }
    },

    {
      "Test_Id": 2,
      "Msg": "ef4eab37181f98423e53e947e7050fd0",
      "Key": "e09eaa5a3f5e56d279d5e7a03373f6ea",
      "e-bit": "0",
      "result": {
        "Tag": "40facf0e2fb51b73a7472681b033d6dc",
        "E_Tag": "207d6787"
      }
    },

    {
      "Test_Id": 3,
      "Msg": "6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c",
      "Key": "2b7e151628aed2a6abf7158809cf4f3c",
      "e-bit": "0",
      "result": {
        "Tag": "c2ccf55ecf86a406d0e83cbbd9711e05",
        "E_Tag": "61667aaf"
      }
    },

    {
      "Test_Id": 4,
      "Msg": "6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411",
      "Key": "2b7e151628aed2a6abf7158809cf4f3c",
      "e-bit": "0",
      "result": {
        "Tag": "dfa66747de9ae63030ca32611497c827",
        "E_Tag": "6fd333a3"
      }
    }
  ]
},
{
  "Category": "AES-CMAC Tag / E=1",
  "comment": "Calculate AES-CMAC Tag and E_Tag for a given message with E_bit=0 ",
  "Test Vectors": [

    {
      "Test_Id": 1,
      "Msg": "",
      "Key": "e34f15c7bd819930fe9d66e0c166e61c",
      "e-bit": "1",
      "result": {
        "CT": "",
        "Tag": "d47afca1d857a5933405b1eb7a5cb7af",
        "E_Tag": "ea3d7e50"
      }
    }
  ]
},
```

```
{
  "Test_Id": 2,
  "Msg": "eaa91273e7",
  "Key": "43151bbaef367277ebfc97509d0aa49c",
  "e-bit": "1",
  "result": {
    "CT": "2248f2872c",
    "Tag": "0069bdbbe97044707df74ee9e98cb42fe",
    "E_Tag": "8034dedf"
  }
},
{
  "Test_Id": 3,
  "Msg": "6bc1bee22e409f96e93d7e117393172a",
  "Key": "2b7e151628aed2a6abf7158809cf4f3c",
  "e-bit": "1",
  "result": {
    "CT": "a3205e16e53f285bd352d4c47575b680",
    "Tag": "939e07cb94c59d8b75aff9b7814cfaeb",
    "E_Tag": "c9cf03e5"
  }
},
{
  "Test_Id": 4,
  "Msg": "6bc1bee22e409f96e93d7e117393172aae2d8a571e03ac9c9eb76fac45af8e5130c81c46a35ce411",
  "Key": "2b7e151628aed2a6abf7158809cf4f3c",
  "e-bit": "1",
  "result": {
    "CT":
"a3205e16e53f285bd352d4c47575b6803ae1231a7a15803bd0d2ce0431b666d7cee0ebd0c5bc3ff5",
    "Tag": "5d0f7987d9aca79e536cafcbdf2e1a9c",
    "E_Tag": "ae87bcc3"
  }
}
],
{
  "Category": "SecOC/E",
  "comment": "Calculate AES-CMAC Tag and E_Tag for a given message with E_bit=0 ",
  "Test Vectors": [
    {
      "Test_Id": 1,
      "Key": "2b7e151628aed2a6abf7158809cf4f3c",
      "Plain_Text": "416c696365426f62",
      "Msg_ID": "19f11afd",
      "fv": "77359400",
      "e-bit": "0",
      "result": {
        "Nonce": "01f11afd77359400",
        "CT": "",
        "Tag": "2138693119472ceea6ab55d7f2dabde8",
        "E_Tag": "109c3498"
      }
    },
    {
      "Test_Id": 2,
      "Key": "2b7e151628aed2a6abf7158809cf4f3c",
      "Plain_Text": "416c696365426f62",
      "Msg_ID": "19f11afd",
      "fv": "77359400",
      "e-bit": "1",
      "result": {
        "Nonce": "81f11afd77359400",
        "CT": "baf013a584b339a4",
        "Tag": "bc430de4d0e63b15e90f143db4566394",
        "E_Tag": "de2186f2"
      }
    }
  ]
}
```

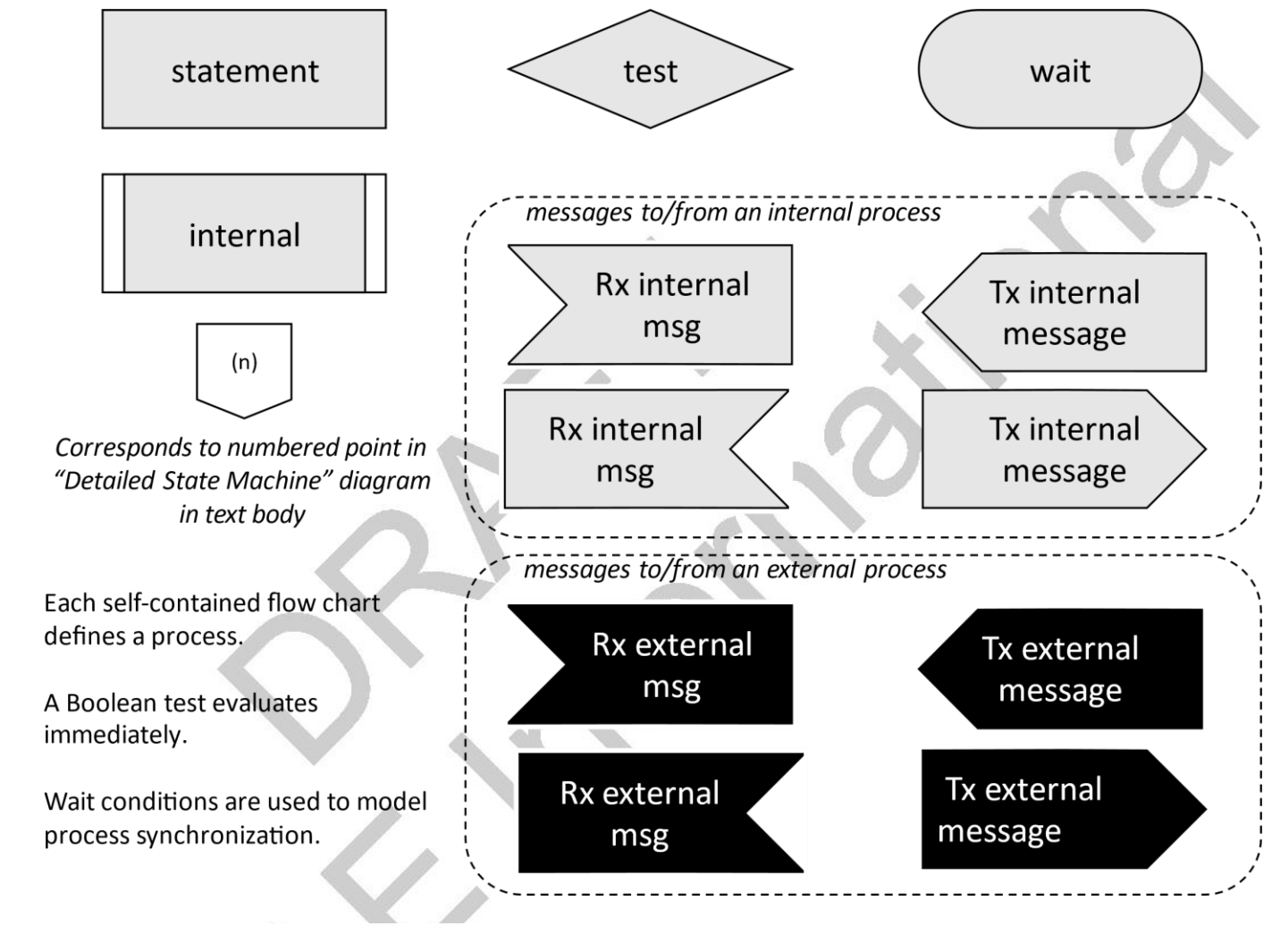
```
},
{
  "Test_Id": 3,
  "Key": "481440298525cc261f8159159aedef62d",
  "Plain_Text": "6123c556c5cc",
  "Msg_ID": "18c12ec1",
  "fv": "25484012",
  "e-bit": "0",
  "result": {
    "Nonce": "00c12ec125484012",
    "CT": "",
    "Tag": "b7073c020fe01525819ee7c118f2d255",
    "E_Tag": "5b839e01 "
  }
},
{
  "Test_Id": 4,
  "Key": "481440298525cc261f8159159aedef62d",
  "Plain_Text": "6123c556c5cc",
  "Msg_ID": "18c12ec1",
  "fv": "25484012",
  "e-bit": "1",
  "result": {
    "Nonce": "80c12ec125484012",
    "CT": "ac0951abd6fa",
    "Tag": "dabb45bb6d277cedbcac09f7895c048a",
    "E_Tag": "ed5da2dd"
  }
}
]
}
```

Figure D1 - Cryptographic Operations Test Vectors

APPENDIX E - MODIFIED SDL DIAGRAMS

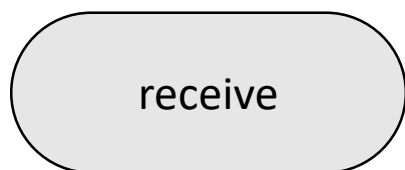
E.1 FLOW CHART SYMBOLS

Figure E1 **Error! Reference source not found.**and Figure E2 describes the symbols used in the behavior models presented in this appendix.



¹ Derivative of flowchart concepts illustrated by Gerard Holzmann in his text *Design and Validation of Computer Protocols*

Figure E1 – SDL Flow Chart Symbol Descriptions



The universal input queue for state machine messages.

It appears in many diagrams, and refers to the same queue.



This location in SDL diagram is identical to the transition on the state diagram.



This location in SDL diagram is where to the transition on the state diagram can map into (it doesn't always. If it did, we use '==' notation.)

Figure E2 – SDL Chart Symbols, continued.

E.2 TIMER VERBS

Figure E3 describes the timer verbs used in this appendix.

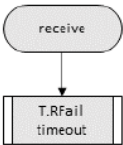
Timer Verb	Action	Note	Graphic
start	starts a timer at its initial value	Used to indicate that the timer should not already be running; if it is then there is likely a logic error.	
restart	same as 'start'	Used to indicate that the timer should be already running; if it is not then there is likely a logic error. The value previous to resetting to the initial value is unimportant.	
(re-)start	same as 'start'	Used to indicate that the timer may or may not already be running. A previous value, if any, is unimportant.	
stop	stops the timer so it cannot timeout	Note that there is no provision for "pausing" a timer; hence after a "stop" the timer value is meaningless.	
timeout	the timer has alarmed	the alarm is detected by matching a guard after a "receive" command, as shown in here:	

Figure E3 – Timer Verb Definitions

E.3 LEGEND

Figure E4 describes states used in this appendix.

States	
IN_E	true when in Exchange Message (either E ₀ or E ₁)
IN_F	true when in Failed
IN_N	true when in Network Formation
IN_O	true when in ECU OFF
IN_R	true when in Rekey
Internal Messages	
do_r	rekey is called for
init_done	initialization of SecOC/E is done (ECU power on is completed)
other_rekey	we need to respond to another ECUs request to rekey
repeat_nonce	we need to repeat the nonce we've already sent
send_msg	request to create and send a secure message

Figure E4 – States for an ECU participating in SecOC/E

As shown in the “detailed state machine” diagram, an ECU can, for example, be in both “Rekey” (IN_R) and exchanging secure messages (IN_E). The Internal Messages are used to signal within an ECU.

E.4 REKEY

This section contains operational model diagrams related to rekey. There are seven process flows in rekey that are illustrated in Figures E5 – E9.

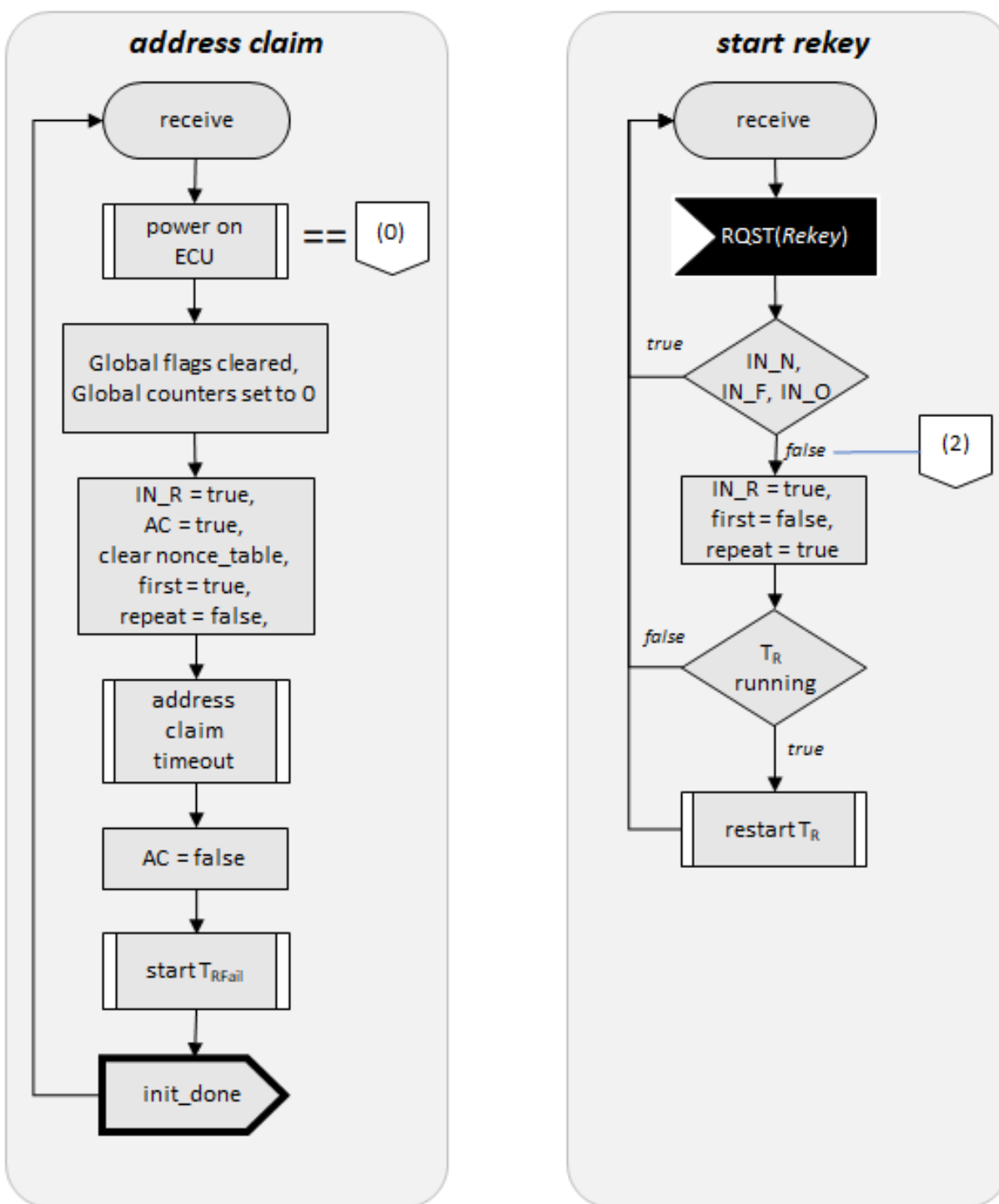


Figure E5 – Rekey: Address Claim and Start Processes

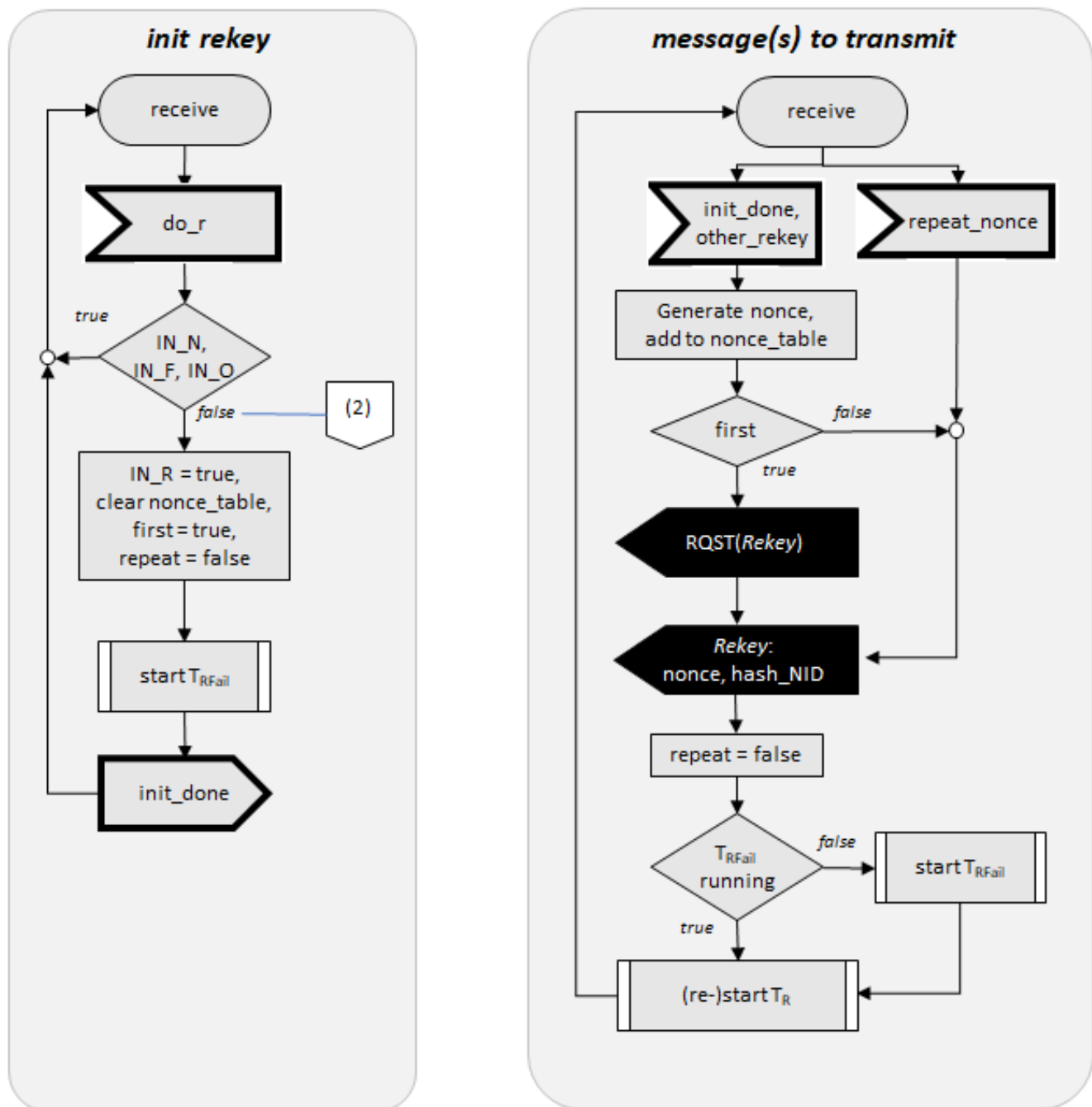


Figure E6 – Rekey: Init and Message Transmit Processes

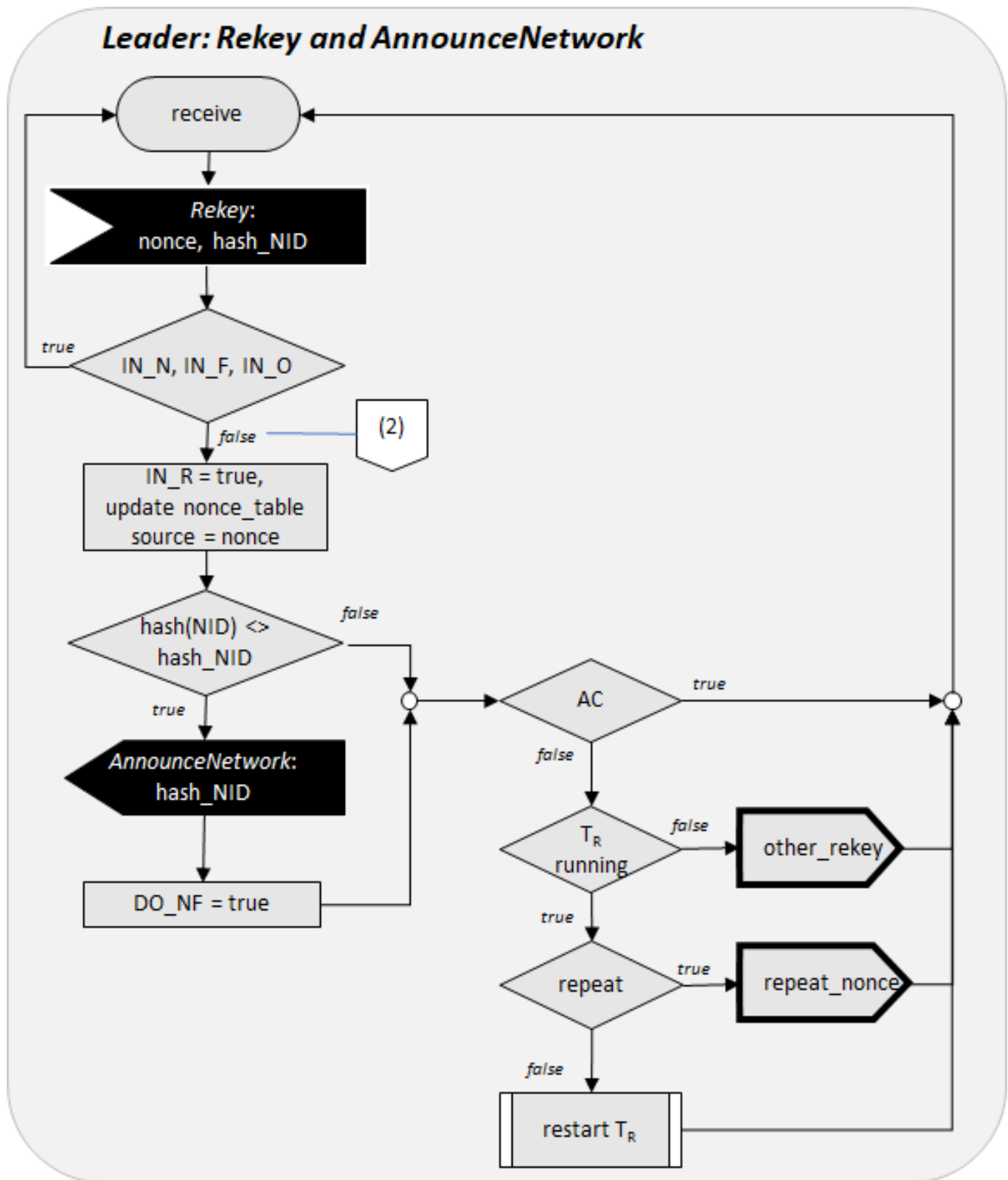


Figure E7 – Leader: Rekey and Announce Network Process

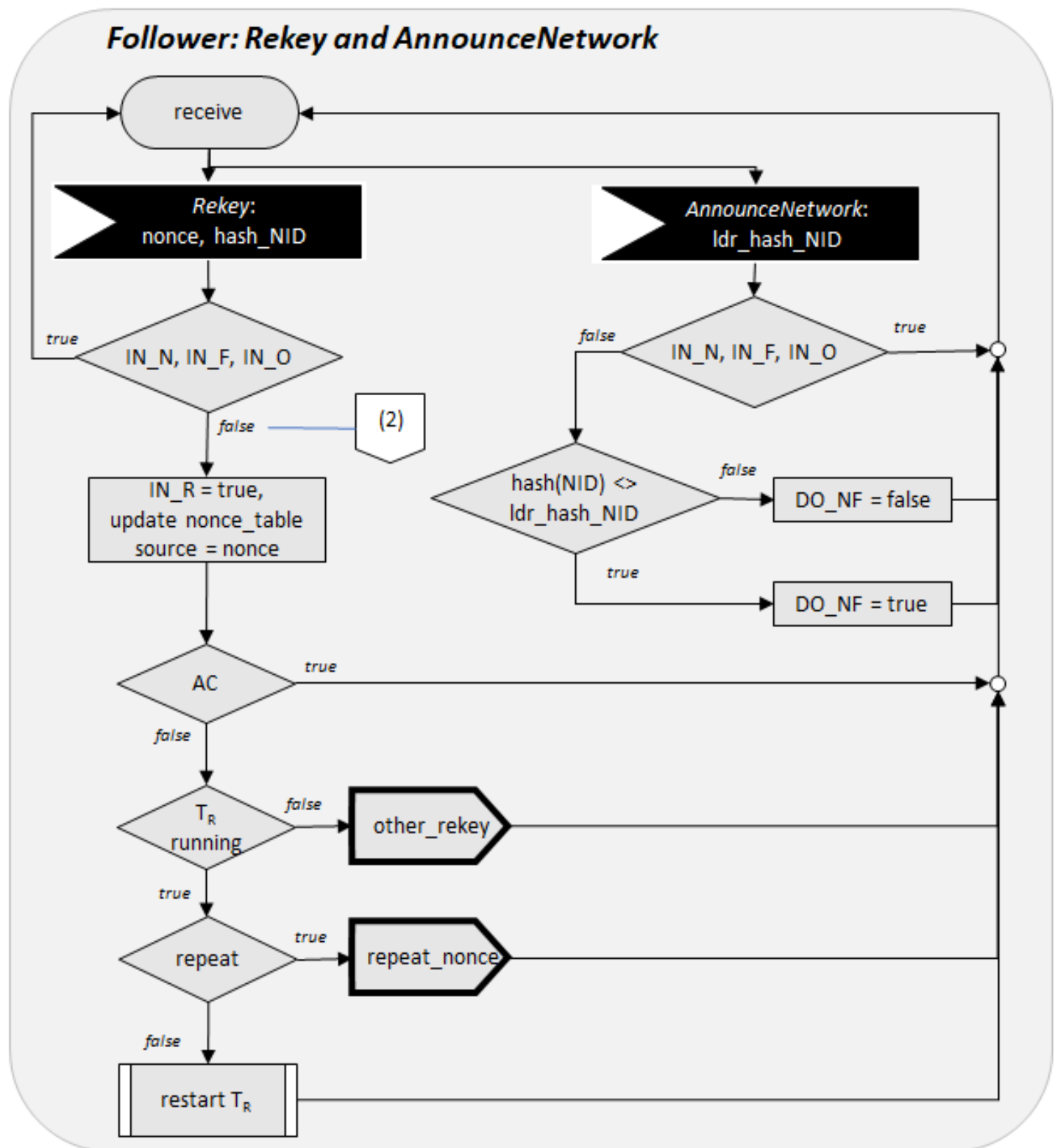


Figure E8 – Follower: Rekey and Announce Network Processes

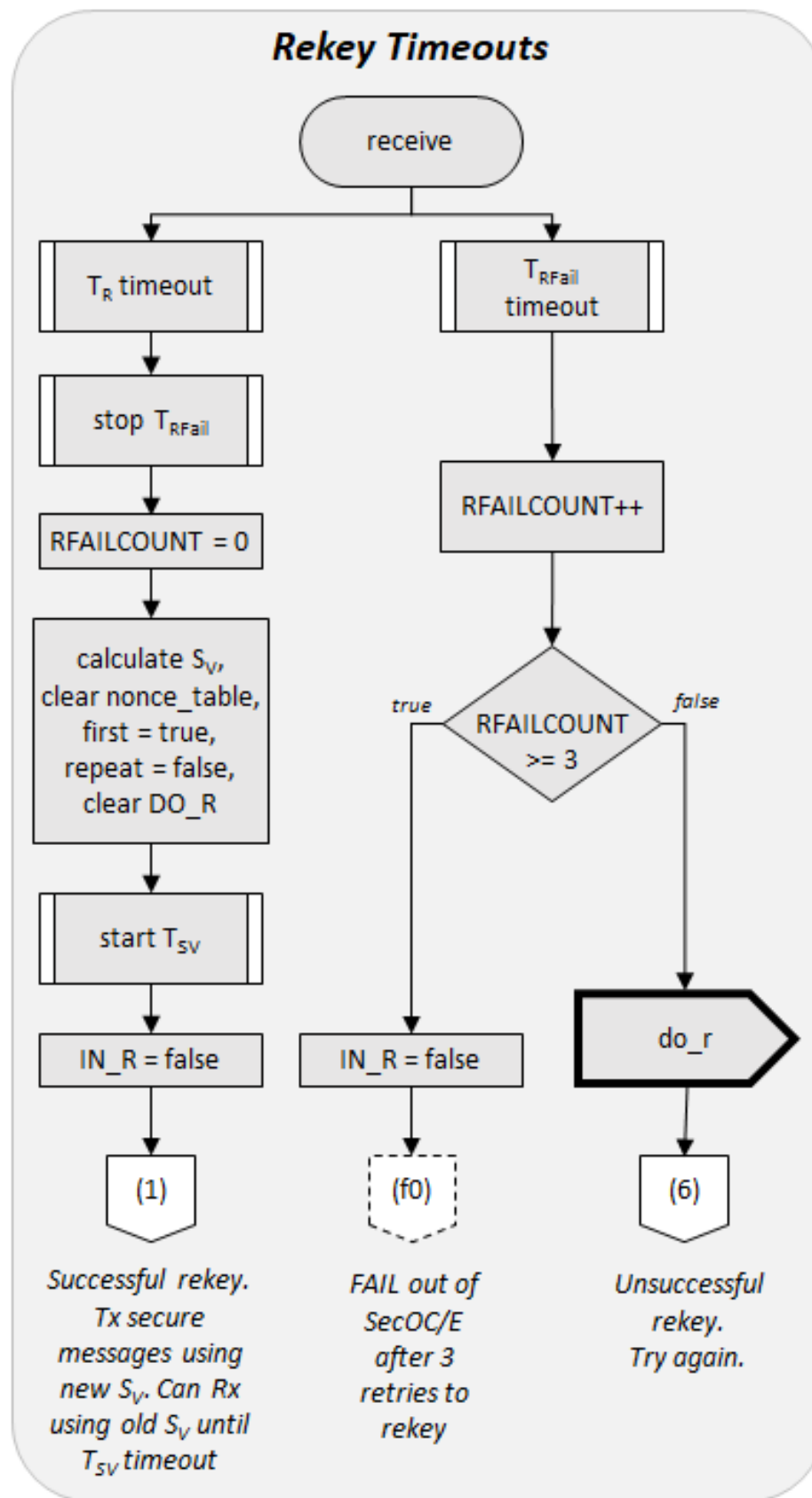


Figure E9 – Rekey Timeout Process

E.5 EXCHANGE MESSAGES

This section contains operational model diagrams related to receiving and sending secure messages. There are three process flows that are illustrated in two figures, Figures E10 – E11.

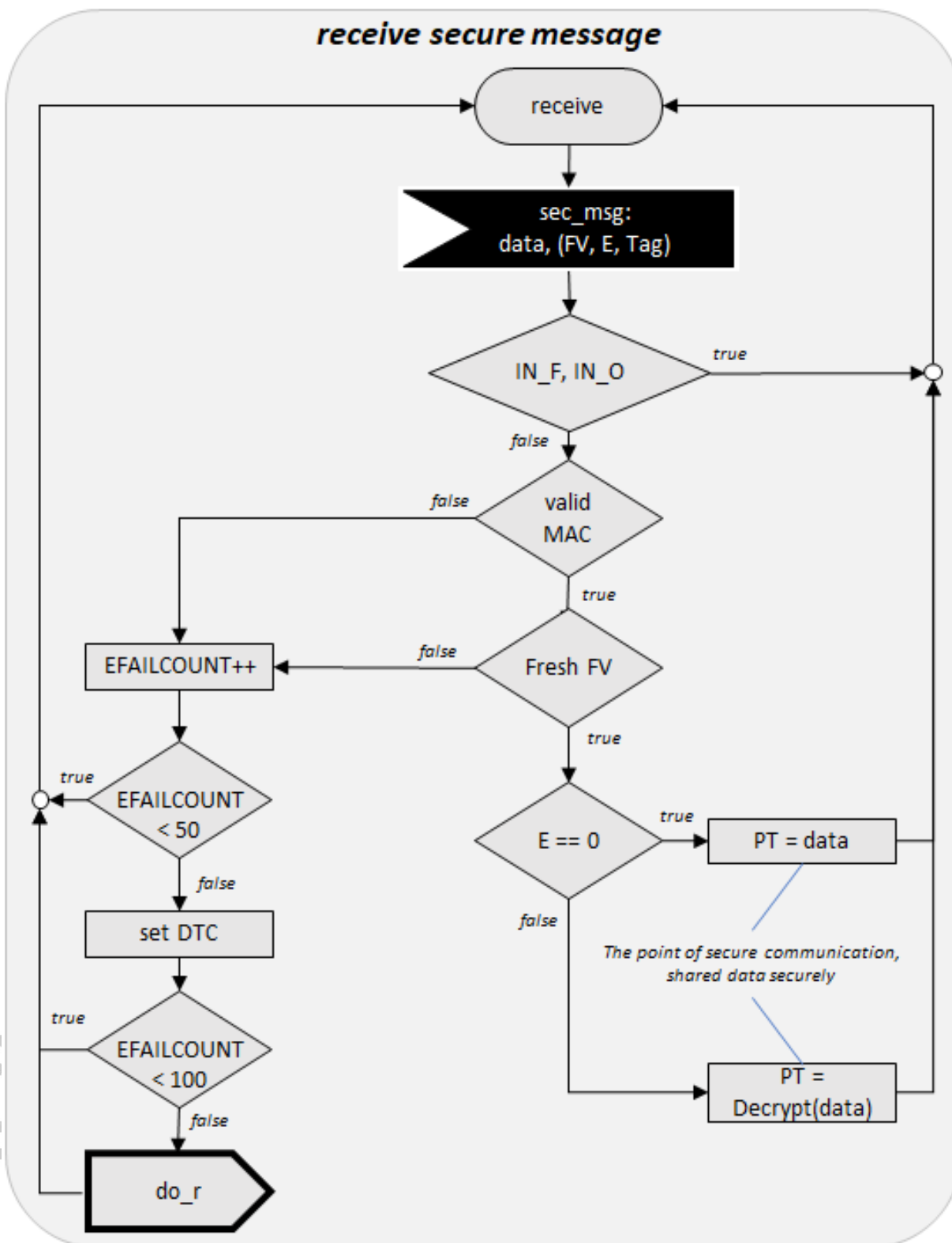


Figure E10 – Receiving Secure Message Process

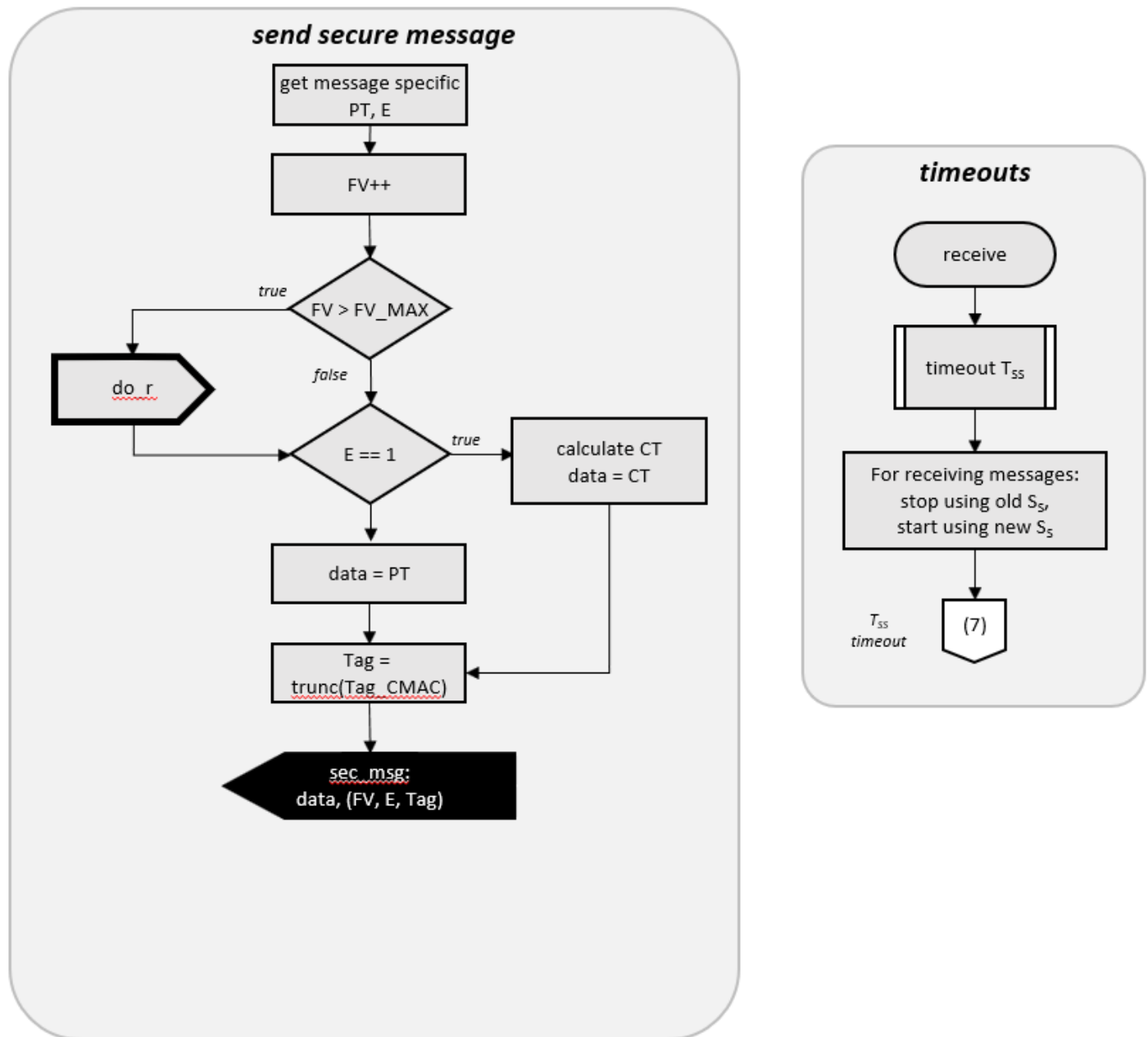


Figure E11 - Send Secure Message and Timeout Processes

E.6 NETWORK FORMATION

This section contains operational model diagrams related to network formation. There are process flows for the Leader that are illustrated in Figure E12. There are process flows for a Follower that are illustrated in two figures, Figure E13 and Figure E14.

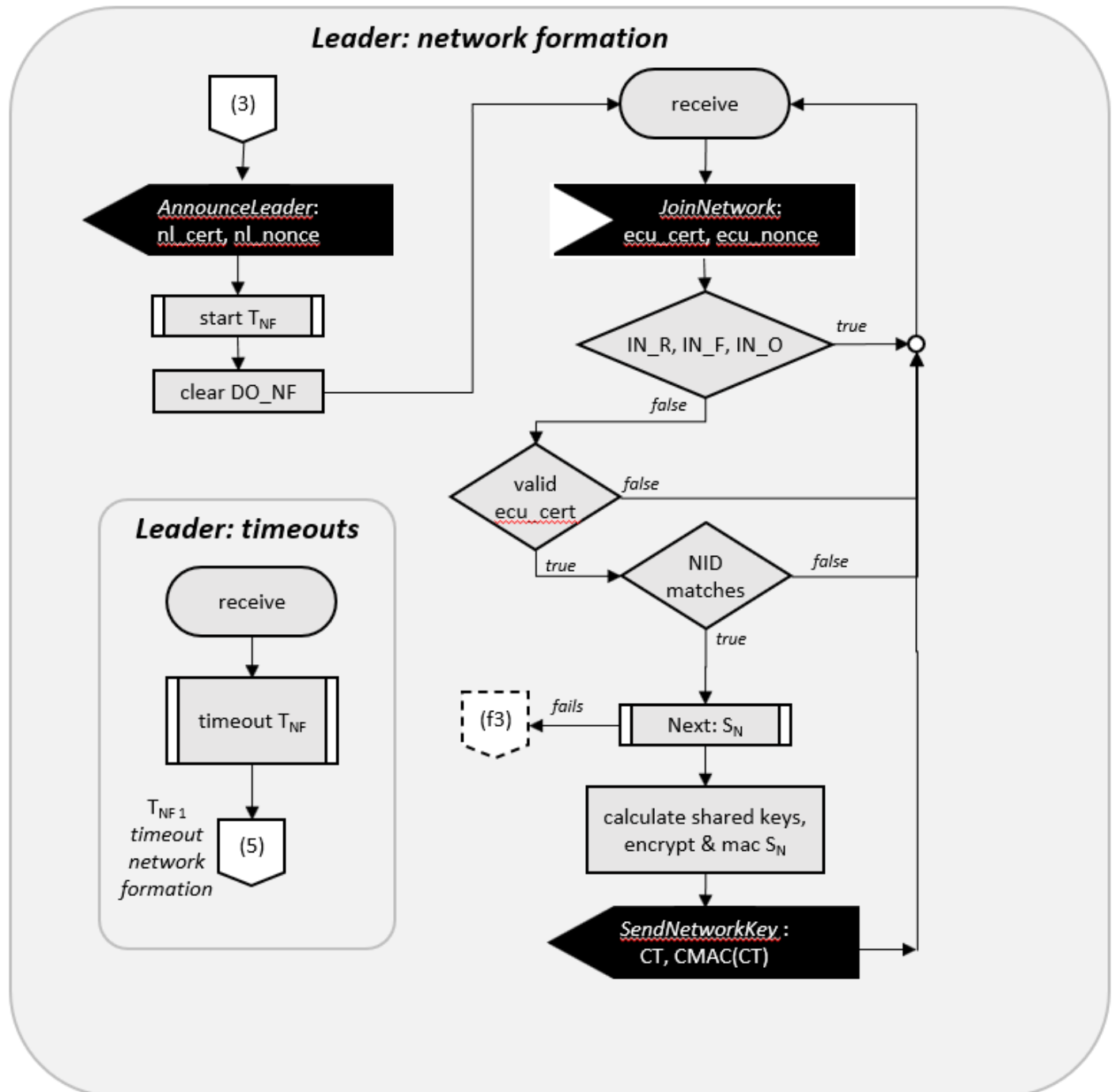


Figure E12 – Leader: Network Formation and Timeout Process

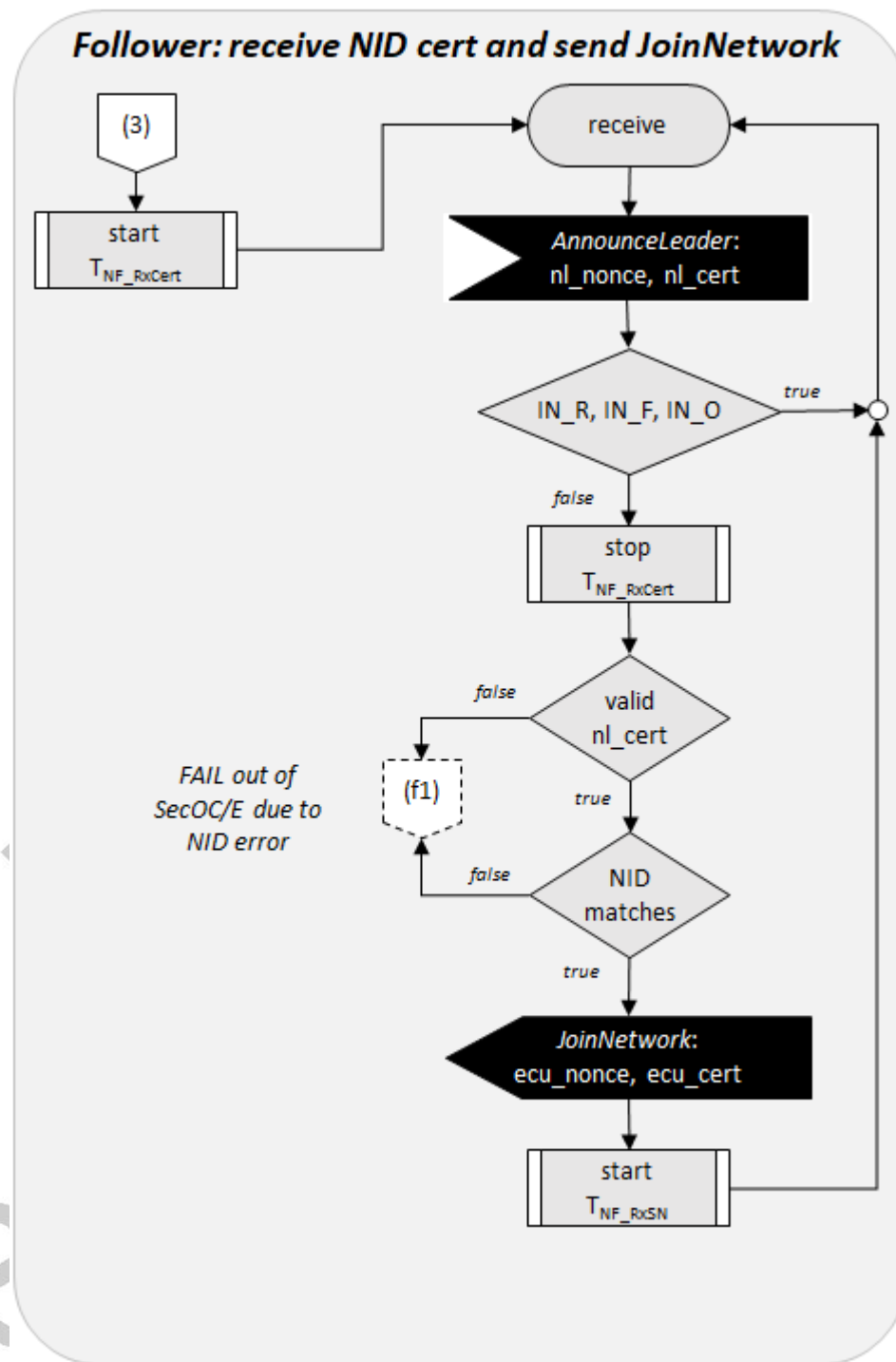


Figure E13 – Follower: Network Formation Process

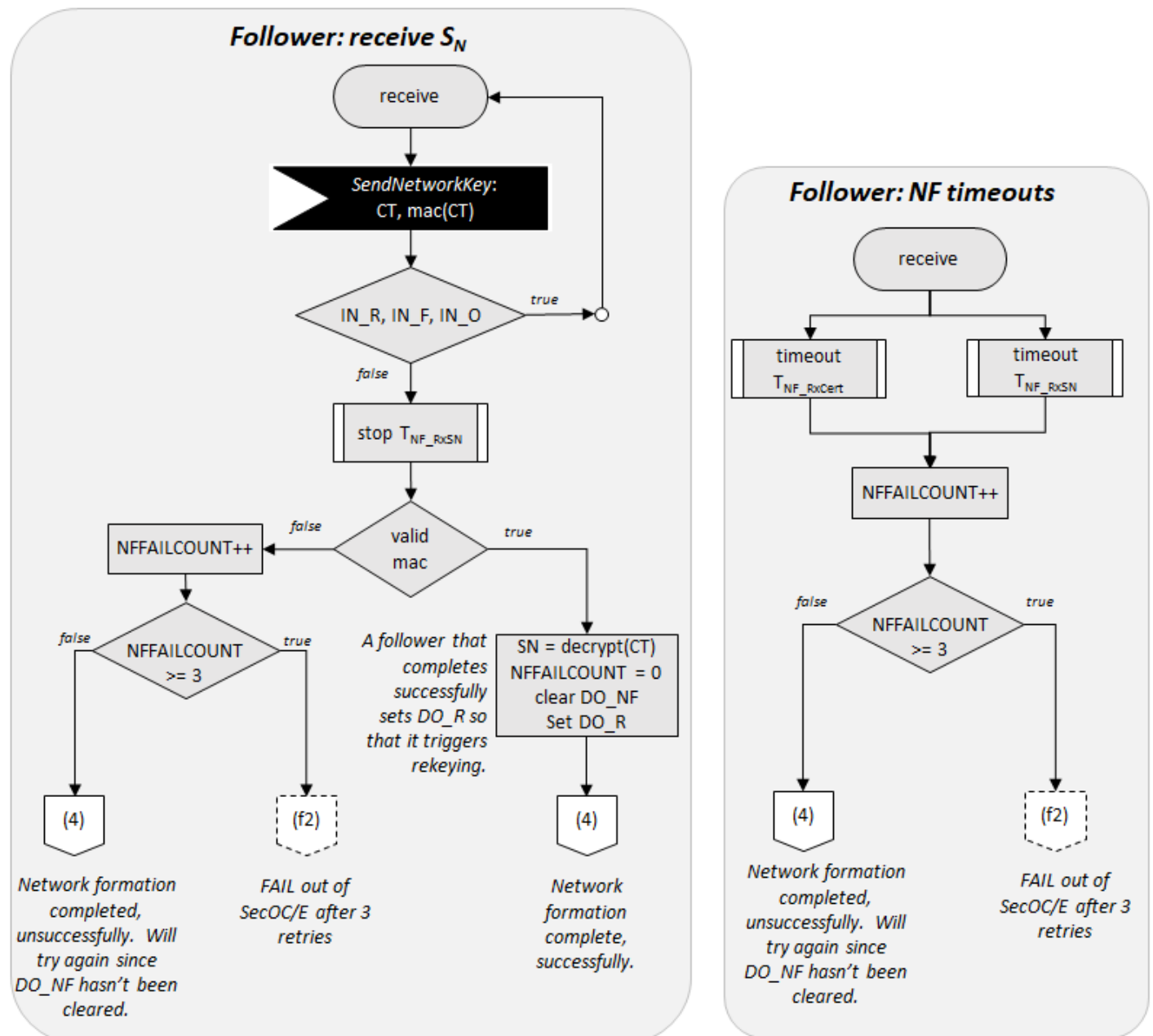


Figure E14 –Follower: Receive Network Key and Timeout Processes

APPENDIX F - SECURE DIAGNOSTICS NETWORK

A secure diagnostics network concept is shown in Figure F1. A service tool creates a secure diagnostics network with the proxy. The proxy decides what, when and how messages are forwarded to and from the service tool. The proxy does not compromise a secure vehicle network – a device attached to the secure diagnostic network does not obtain S_N .

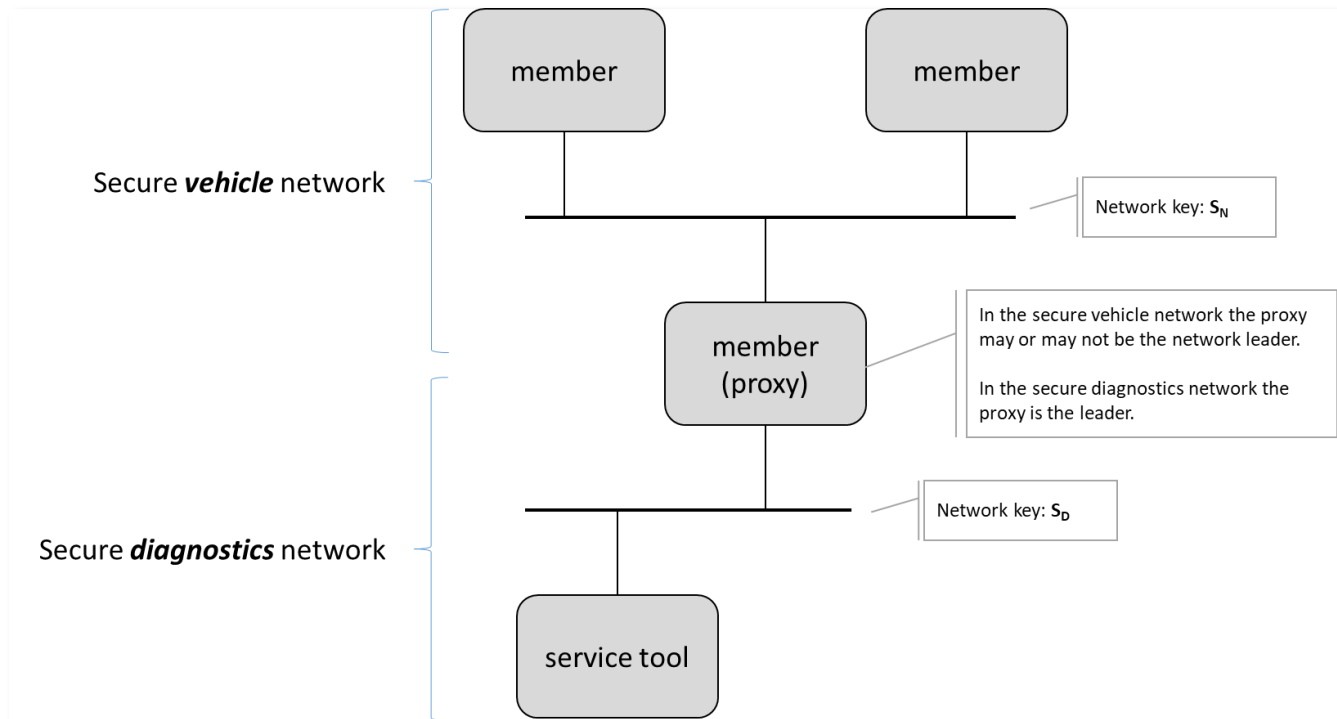


Figure F1 - Secure Diagnostics Network Concept

F.1 PROXY BEHAVIOR

An integrated member ECU should be between the diagnostics connector and the rest of the vehicle network. The integrated member ECU includes the proxy function for the service tool. The proxy will have S_N and will use it, when necessary, to convey secure messages from the service tool to other members of the secure vehicle network. The proxy determines vehicle state (ideally, through the use of secure messages from other members of the secure network) and uses that state when deciding if/when messages from the service tool can be proxied to the vehicle.

In cases where messages between the service tool and the proxy need to be secured, a small (two node) secure network should be created between the proxy and the service tool. This small network is the diagnostic network.

F.2 SECURE DIAGNOSTIC NETWORK REQUIREMENTS

1. The proxy shall be the Network Leader for the tool network.
2. The symmetric key for diagnostic network shall be called S_D (instead of S_N).
3. S_D is not long-lived. It shall be forgotten by the proxy at power down of the proxy.
4. The proxy shall create a new S_D on each power cycle. This means the key will go unused should no service tool be attached to the proxy.
5. Network Formation shall occur the same way as in the vehicle secure network. Through rekeying, the proxy will detect the need to share S_D with the service tool.
6. The service tool shall present a certificate to the proxy.

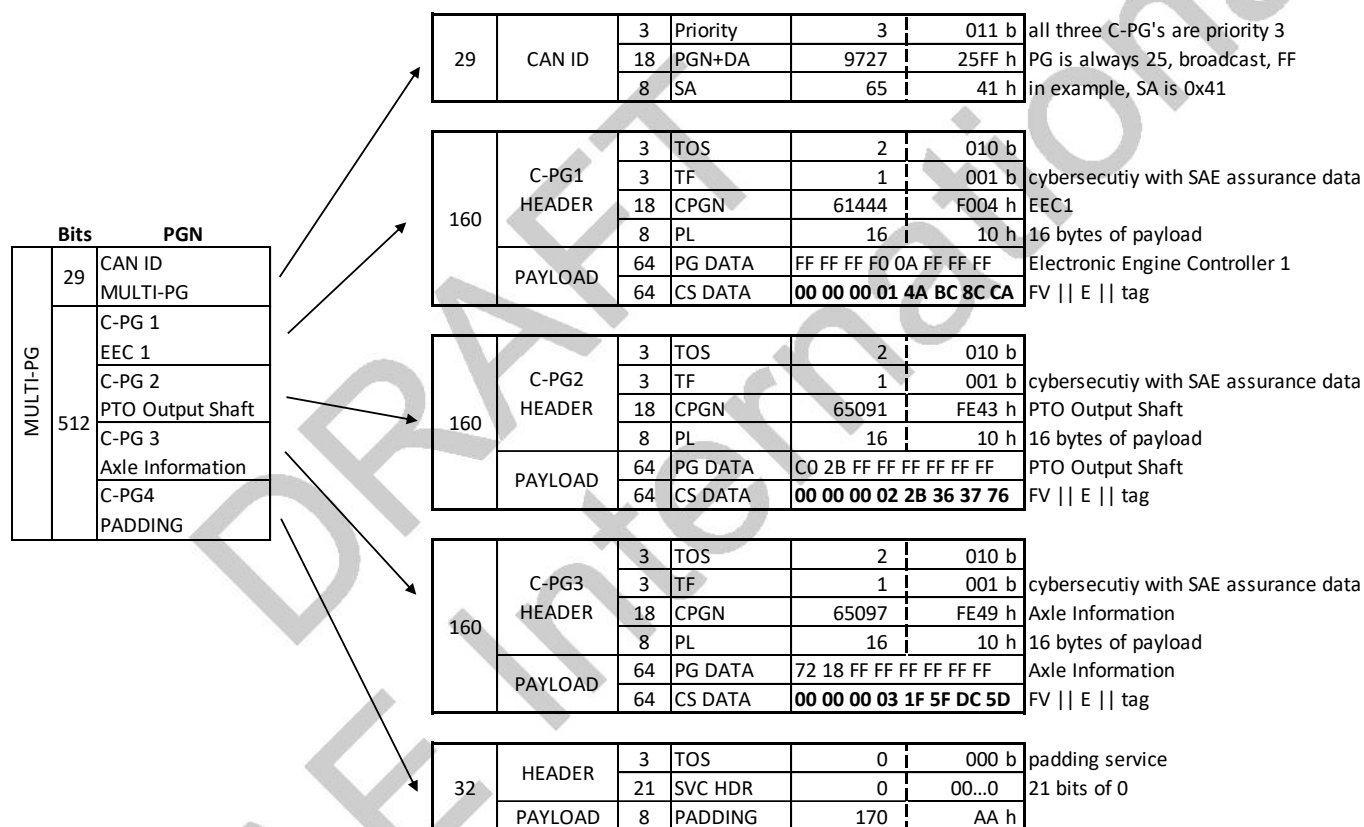
© SAE International
DRAFT

APPENDIX G – SECOC/E EXAMPLE MESSAGE

The example shown in Figure G1 has three contained PGs:

1. Electronic Engine Controller 1 (F004_h) – data = FF FF FF F0 0A FF FF FF
2. PTO Output Shaft (FE43_h) – data = C0 2B FF FF FF FF FF FF
3. Axle Information (FE49_h) – data = 72 18 FF FF FF FF FF FF

Note: the tag values shown in the CS data are calculated values using the CMAC key shown below.



cmac key : 00010203 04050607 08090a0b 0c0d0e0f h

Figure G2 – Example J1939-22 message with SecOC/E Assurance Data