

IEEE Standard for Wireless Access in Vehicular Environments— Security Services for Applications and Management Messages

Amendment 1

IEEE Vehicular Technology Society

Sponsored by the
Intelligent Transportation Systems Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 1609.2a™-2017
(Amendment to
IEEE Std 1609.2™-2016)

IEEE Standard for Wireless Access in Vehicular Environments— Security Services for Applications and Management Messages

Amendment 1

Sponsor

Intelligent Transportation Systems Committee
of the
IEEE Vehicular Technology Society

Approved 28 September 2017

IEEE-SA Standards Board

Abstract: Secure message formats and processing for use by Wireless Access in Vehicular Environments (WAVE) devices, including methods to secure WAVE management messages and methods to secure application messages are defined. Administrative functions necessary to support the core security functions are described.

Keywords: cryptography, IEEE 1609.2™, security, wireless access in vehicular environments (WAVE)

Acknowledgment

The IEEE P1609 Working Group would like to acknowledge significant contributions made to this standard by Drew van Duren.

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2017 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 13 October 2017. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-1-5044-4270-1 STD22746
Print: ISBN 978-1-5044-4271-8 STDPD22746

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE-SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at <http://ieeexplore.ieee.org/> or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at <http://standards.ieee.org>.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this IEEE standard was completed, the Dedicated Short Range Communications Working Group had the following membership:

Thomas M. Kurihara, *Chair*
Justin McNew, Kevin Smith, William Whyte, *Vice Chair*

Mike Brown
Hanbyeog Cho
Hans-Joachim Fischer
Ramez Gerges
Aleksandar Gogic
Shubha Gopalakrishna
Gloria Gwynne
Ronald Hochnadel

Carl Kain
John Kenney
Bill Lattin
Jules Madey
Sean Maschue
Jim Misener
Frank Perry

Randy Roebuck
Richard Roy
Kevin Smith
Jasja Tijink
Michaela Vanderveen
George Vlantis
Jason Wang
Aaron Weinfield

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Mohammad Battah
Aleksandar Gogic
Gloria Gwynne
Ron Hochnadel
Carl Kain
John Kenney

Soyoung Kim
Jules Madey
Sean Maschue
Jim Misener
Randy Roebuck
Richard Roy

Kevin Smith
Jasja Tijink
Michaela Vanderveen
George Vlantis
Jason Wang
Ken Vaughn

When the IEEE-SA Standards Board approved this standard on 28 September 2017, it had the following membership:

Jean-Philippe Faure, *Chair*
Gary Hoffman, *Vice Chair*
John Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Chuck Adams
Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Doug Edwards
J. Travis Griffith
Michael Janezic

Thomas Koshy
Joseph L. Koepfinger*
Kevin Lu
Daleep Mohla
Damir Novosel
Ronald C. Petersen
Annette D. Reilly

Robby Robson
Dorothy Stanley
Adrian Stephens
Mehmet Ulema
Phil Wennblom
Howard Wolfman
Yu Yuan

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1609.2a-2017, IEEE Standard for Wireless Access in Vehicular Environments—Security Services for Applications and Management Messages—Amendment 1.

This amendment addresses multiple needs to enhance and extend IEEE Std 1609.2-2016:

- Since the publication of IEEE Std 1609.2-2016, a number of errors, omissions, and ambiguities have been discovered, which this amendment corrects.
- Industry stakeholders have requested additional functionality, in particular better support for compact expressions of ranges of Service Specific Permissions (SSPs).
- Test vectors are provided to enable implementers to gain confidence in correctness of their implementation before running interoperability tests.
- Additional informative material is provided to assist implementers of the standard and users of the security services in understanding the intended implementation and use.

Contents

3. Definitions, abbreviations, and acronyms	9
3.1 Definitions	9
4. General description.....	10
4.2 Secure data service (SDS)	10
4.3 Security services management entity (SSME).....	11
5. Cryptographic operations and validity.....	11
5.1 Certificate validity	11
5.2 Signed SPDU validity.....	18
5.3 Cryptographic operations.....	29
6. Data structures	32
6.1 Presentation and encoding	32
6.2 ~~Integer~~ Basic types	33
6.3 Secured protocol data units (SPDUs)	33
6.4 Certificates and other security management data structures	43
7. Certificate revocation lists (CRLs) and the CRL Verification Entity	54
7.3 Data structures	54
7.4 CRL: 1609.2 Security envelope.....	55
8. Peer-to-peer certificate distribution (P2PCD).....	58
8.1 General	58
8.2 P2PCD operations.....	58
8.4 Data structures	64
9. Service primitives and functions	65
9.1 General comments and conventions	65
9.4 SSME SAP	71
9.5 SSME-Sec SAP	73
Annex A (informative) Protocol Implementation Conformance Statement (PICS) proforma	74
A.2 PICS proforma—IEEE Std 1609.2	74
Annex B (normative) ASN.1 modules.....	79
B.0a General	79
B.1 1609.2 security services	79
B.2 Certificate revocation list (CRL).....	81
B.3 Peer-to-peer certificate distribution (P2PCD)	83
Annex C (informative) Specifying the use of IEEE Std 1609.2™ by SDEEs.....	84
C.2 IEEE 1609.2 security profiles	84
C.3 IEEE 1609.2 security profile proforma	88
C.4 Service Specific Permissions (SSP).....	90
C.7 Source of encryption keys	91
Annex D (informative) Examples and use cases	93
D.5 Example data structures	93
D.6 Cryptographic test vectors	96

IEEE Standard for Wireless Access in Vehicular Environments— Security Services for Applications and Management Messages

Amendment 1

NOTE—The editing instructions contained in this amendment define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.¹

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~striketrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.

3. Definitions, abbreviations, and acronyms

3.1 Definitions

Delete the following definitions:

¹ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

(permitted) depth of chain (in a certificate authority [CA] certificate's permissions): The length that a certificate chain is permitted to be, starting from the certificate with which the depth value is associated and ending with an end-entity certificate.

permitted depth of chain: *See:* **depth of chain.**

Insert the following definitions in alphanumeric order:

end-entity: An entity that is requesting certificates or signing Protocol Data Units. *Contrast:* **certificate authority.** Note that an entity may act as an end-entity in one context and as a Certificate Authority in another context.

length of certificate chain (In a certificate chain from any CA certificate to a final certificate, with each certificate in the chain issuing the one after it): The number of certificates in the chain, except for the first one; in other words, the number of certificates in the chain, minus 1.

whole-certificate hash algorithm: The algorithm used to calculate the hash of a certificate for purposes of identifying that certificate.

4. General description

4.2 Secure data service (SDS)

4.2.1 Secured protocol data units (SPDUs)

Change the last paragraph of 4.2.1 as follows:

SPDUs may be of type *unsecured*, *signed*, ~~or encrypted~~, or signed certificate request. A SPDU may contain another SPDU of the same or different type.

4.2.2 Secure data service

4.2.2.3 Processing received SPDUs

4.2.2.3.1 Preprocessing

Change the last two paragraphs of 4.2.2.3.1 as follows:

The SDS is requested to preprocess a received SPDU via Sec-SecureDataPreprocessing.request. The SDS returns the result of the request to the requesting SDEE via Sec-SecureDataPreprocessing.confirm. The result is:

- The type of the SPDU
- If the SPDU was of type signed:
 - The Service Specific Permissions of the signer as defined in 5.2.3.3.3
 - The geographic validity region of the signer's certificate as defined in 6.4.17

- The assurance level of the signer’s certificate as defined in 6.4.27
- The earliest Next CRL time of any certificate in the chain as defined in 5.1.3.6

When providing this service, the SDS extracts security management information and passes it to the SSME to support P2PCD and digest-form SignerIdentifier structures.

4.2.5 1609.2 security profile

Change the first paragraph of 4.2.5 as follows:

The information elements used by the SDS operations are specified in 9.2.2 as (sometimes optional) parameters to primitives; the SPDU data structures and their encodings are specified in Clause 6. The *IEEE 1609.2 security profile* (occasionally referred to in this document simply as the “security profile”) specified in Annex C is a format suggested for use by the specifier of a SDEE as a compact way to specify which SDS parameters are used and which values they should take for that particular SDEE. Additionally, the security profile allows the SDEE specifier to specify other aspects of the security behavior of the SDEE; see Annex C for more information.

4.3 Security services management entity (SSME)

4.3.1 General

Delete the final sentence of 4.3.1:

Any certificate information added to an instance of a SSME is made available to all SDEEs and functional entities that have access to that SSME.

5. Cryptographic operations and validity

5.1 Certificate validity

5.1.1 Certificate contents

Change the third bulleted list in 5.1.1 as follows:

Certificate issuance permissions, i.e., the permissions that govern what certificates a CA is authorized to issue, are expressed using the following information elements (see 6.4.32 for a full specification):

- One or more PSIDs.
- For each PSID, the SSP Range, which identifies the SSPs associated with that PSID for which the CA is permitted to grant permissions.
- The permissible length(s) of the certificate chain from the certificate containing these issuance permissions to the certificate that signs the PDU. ~~This length is referred to as the permitted depth.~~ Certificate chain length is defined in 5.1.2.1.

- The end-entity type permissions, which indicate whether the ultimate end-entity certificate permits application operations, certificate request operations, or both.

Change the second paragraph following the third bulleted list in 5.1.1 as follows:

Certificate request permissions are expressed using the same information elements as certificate issuance permissions. When a certificate is being used to request certificates it is referred to as an enrollment ~~enrolment~~ certificate.

Add a footnote to the second item in the fourth bulleted list in 5.1.1 as follows:

If the verification key is not explicitly given in the certificate, but is obtained from a *reconstruction value* in the certificate and the issuer's public key via the reconstruction function specified in 5.3.2, the certificate is an *implicit certificate* and the corresponding verification key is referred to as the *associated public key*.¹ In this case the cryptographic demonstration that the issuer authorized the linkage is provided by the fact that a signature verifies correctly with the verification key that was so derived.

¹ Elliptic Curve Qu-Vanstone (ECQV) or "implicit" certificates were proposed in Brown, Gallant, and Vanstone [B3] and Pintsov and Vanstone [B18], and modifications to protect against attacks were proposed in Brown, Campagna, and Vanstone [B4].

5.1.2 Certificate chain

5.1.2.1 Certificate chain construction

Change the first five paragraphs of 5.1.2.1 and insert Figure 4a and Figure 4b as follows:

A *certificate chain* is a set of certificates ordered from "top" to "bottom", (equivalently, "first" to "last" or "beginning" to "end") such that each certificate in the chain, except the last one, is the *issuing certificate* for one below (or "after") it and each certificate, except the first one, is the *subordinate certificate* of the certificate above (or "before") it.

One certificate is the *issuing certificate* for a second one if the certificate holder of the first certificate used the private key of the first certificate to create the final form of the second certificate, either by signing it (in the case of an explicit certificate) or by carrying out cryptographic operations to create a reconstruction value (in the case of an implicit certificate). The counterpart of an issuing certificate is a *subordinate certificate*. If certificate B is the issuing certificate for certificate A, for compactness this ~~certificate standard~~ uses the terminology "B issues A" even though it would be more correct to use the terminology "B's holder issues A" or "the private key associated with B issues A".

A *trust anchor* is any certificate that is established to be trustworthy by itself, e.g., by preconfiguration or independent provisioning; in other words, not by reference to any other certificate. A necessary (but not sufficient) condition for a certificate to be valid is that it is possible to construct a certificate chain from the certificate to a trust anchor. The SSME stores information about which certificates are trust anchors.

A *root certificate* is an explicit certificate that is verified with the public key included directly in the certificate, in contrast to other certificates that are verified using the verification key of the issuing certificate. There is no distinct issuing certificate for a root certificate. All trusted root certificates are by definition trust anchors. A certificate chain that starts with a root certificate is referred to in this standard as a full certificate chain.

The ~~length of the a certificate chain~~ length of a certificate chain is defined as the number of certificates in the chain apart from the certificate under consideration to the topmost one, or equivalently as the root, inclusive ~~number of both (so if the root were to sign a SPDU, the associated intra-certificate chain length would be 1). This is true even if the~~

local trust anchor used by a particular implementation is not the root but some intermediate CA. gaps in the chain. Figure 4a illustrates these two definitions. Figure 4b illustrates that the definition also applies to a “subchain” within a longer chain, i.e., that the definition does not assume that the topmost certificate is a root certificate or that the bottom certificate is an end-entity certificate.

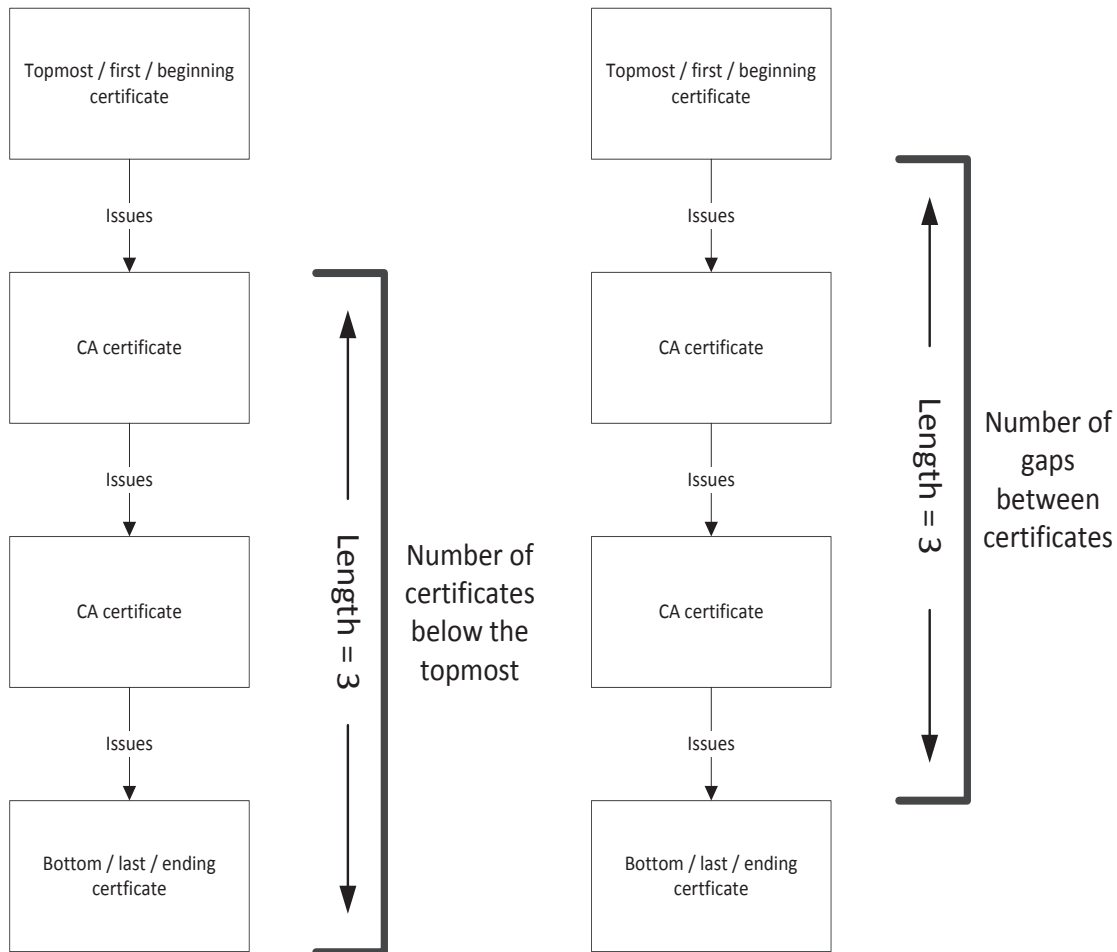


Figure 4a—A certificate chain of length 3

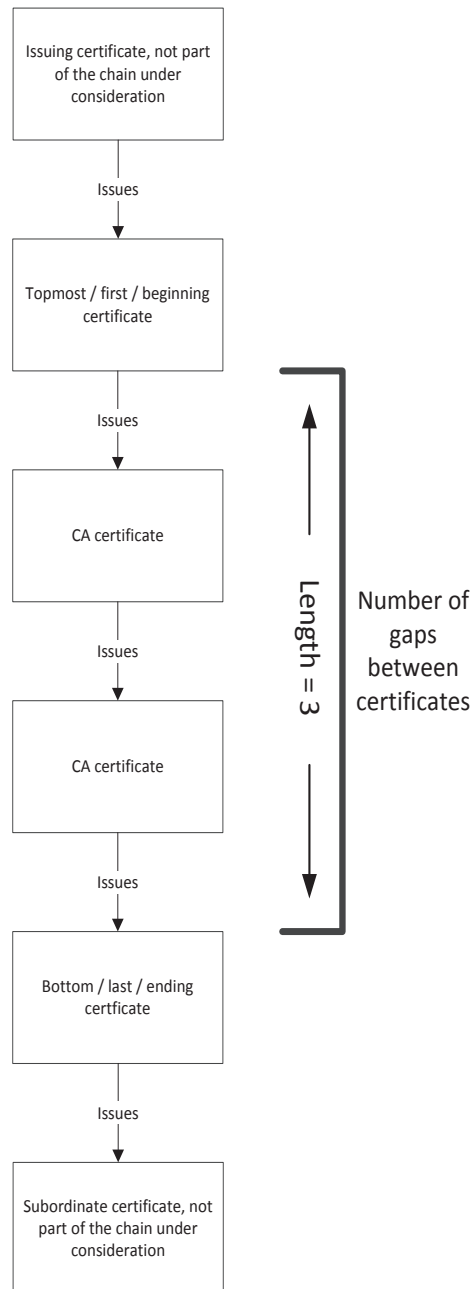


Figure 4b—A subchain of length 3 within a longer chain

5.1.2.2 Maximum supported certificate chain length

Change the text of 5.1.2.2 as follows:

An implementation of WAVE Security Services may have a maximum length of full certificate chain that it can support. A conformant implementation shall support a maximum length of at least eight-two, i.e., a maximum total number of certificates in the chain of at least three. The Protocol Implementation

Conformance Statement (PICS) proforma given in Annex A allows the vendor of an implementation of WAVE Security Services to state the maximum length of certificate chain that the implementation supports.

5.1.2.3 Cryptographic validity of a chain

Change the captions of Figure 5 and Figure 6 as follows:

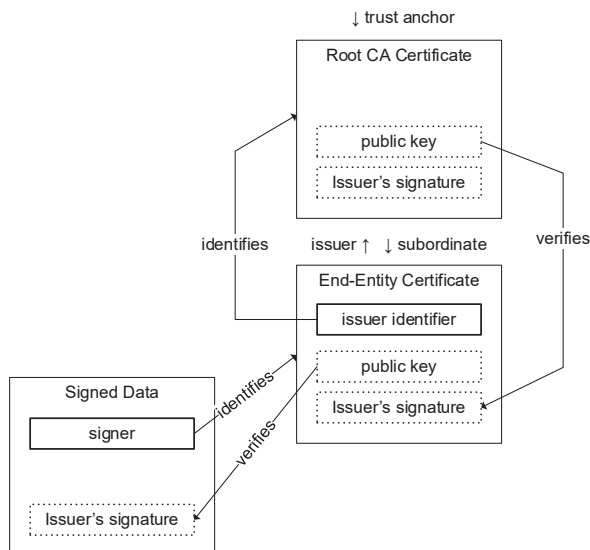


Figure 5—Cryptographic verification with a signed SPDU and with a full certificate chain, using explicit certificates

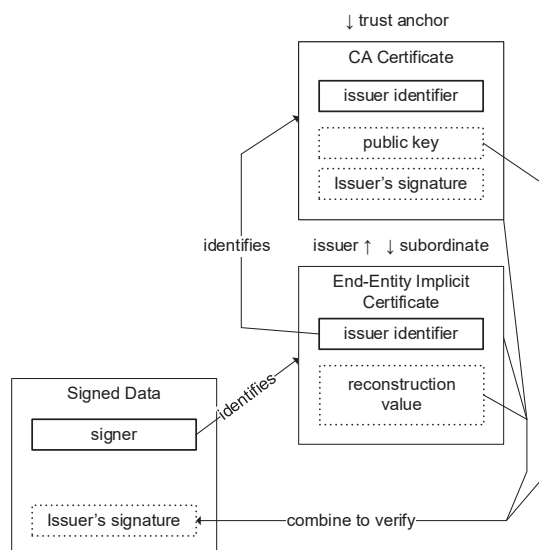


Figure 6—Cryptographic verification of a signed SPDU and with a (non-full) certificate chain with implicit certificates and a non-root CA as trust anchor end-entity certificate

5.1.2.4 Consistency of permissions within a certificate chain

Change the indicated text in 5.1.2.4 as follows:

A self-signed certificate (i.e., a root CA certificate) is consistent with itself by definition. A subordinate certificate is consistent with its issuing certificate if the following conditions hold:

- **Consistency of application/issuance permissions:** The subordinate certificate's application or certificate issuance permissions are consistent with the issuing certificate's certificate issuance permissions, i.e., for every (PSID, SSP) entry in the subordinate certificate's application or certificate issuance permissions (the "subordinate entry") there is an entry in the issuing certificate's certificate issuance permissions (the "issuing entry") such that:
 - *[other bullet points omitted]*
 - If the subordinate entry is for certificate issuance ~~or request permissions~~, the permitted ~~depth~~length of the chain in the subordinate entry is consistent with the permitted ~~depth~~length of the chain in the issuing entry. Specifically, if ~~minChainDepth~~minChainLength and ~~chainDepthRange~~chainLengthRange in the subordinate certificate and issuing certificates have the values ~~medmcr_s~~, ~~edrclr_s~~, ~~medmcr_i~~, ~~edrclr_i~~, respectively, then ~~medmcr_i~~ ≤ ~~medmcr_s~~+1 and (~~medmcr_i~~ + ~~edrclr_i~~) ≥ (~~medmcr_s~~ + ~~edrclr_s~~ +1). (In the case where the subordinate certificate is an end-entity certificate, ~~medmcr_s~~ and ~~edrclr_s~~ are set equal to zero (0) in these formulas.)

Insert 5.1.2.4a after 5.1.2.4:

5.1.2.4a Trustworthiness of a certificate chain

A certificate chain is trustworthy if the following hold:

- Each subordinate certificate is consistent with its issuing certificate.
- The chain begins with a trust anchor.
- None of the certificates in the chain have been revoked, as discussed in 5.1.3.
- (Optional) none of the certificates in the chain have expired at the time of chain verification, as discussed in 5.2.4.2.1 and 5.2.4.2.7.

Whether the expiry test is applied is specified as part of the SDEE specification, as discussed in 5.2.4.2.7. It is strongly recommended that chains with expired certificates are treated as untrustworthy.

5.1.3 Revocation and expiry

5.1.3.1 General

Change the first four paragraphs of 5.1.3.1 as follows:

A certificate is said to be revoked if an appropriately authorized entity states that that certificate is known not to be trustworthy. If a certificate is revoked, the SDS shall consider all SPDUs signed by that certificate and received after the issue date of the revocation statement to be invalid even if their stated generation time is before the issue date of the revocation statement.

If a CA certificate is revoked, the SSME shall indicate that any certificates issued by that CA certificate and first received after the issue date of the revocation statement are revoked, even if their stated ~~generation~~

~~time~~start of validity period is before the issue date of the revocation statement. This applies to any certificate that chains back to the revoked CA certificate.

Information about revoked certificates is stored by the SSME via the 9.4.9.1 SSME-AddRevocation-Info.request and 9.4.9.6 SSME-AddRevocationInfo.confirm primitives. The SSME provides the revocation status of certificates via the 9.4.1.1 SSME-CertificateInfo.request and 9.4.1.6 SSME-Certificate-Info.confirm primitives. Revocation information consists of a series of individual data items and information allowing the SSME to associate the revocation information with specific certificates.

For any certificate *C*, there is at most one authority with authorization to issue revocation information for that certificate. Each such authority might be entitled to issue and keep up to date more than one set of revocation information ~~*R*~~, but there is one specific set that is identified as the one that will ~~indicate that~~ contain revocation information about *C* if it is revoked. ~~If *R* does not indicate that *C* is revoked, then *C* is not revoked.~~

~~Within the SSME~~Therefore, the process of determining whether or not a certificate is revoked involves two steps:

- a) Determine which set of revocation information applies to the certificate.
- b) Determine whether any individual data item within the relevant revocation information indicates that the certificate is revoked.

5.1.3.2 Determining which revocation information applies to a given certificate

Change the start of 5.1.3.2 as follows:

Revocation information applies to a given certificate if it:

- Indicates ~~the same~~ that certificate's *Certificate Revocation Authorizing CA* (CRACA) certificate, and
- Indicates ~~the same~~ that certificate's *CRL Series* value, and
- Is of the appropriate type (linkage-based or hash ID-based)

~~The~~A CRACA is a CA that has authority to authorize the issuance of revocation information for a particular group of other certificates. The CRACA certificate for a certificate *C* is only valid if it is ~~one~~ one of the certificates in *C*'s ~~issuing~~full chain. Likewise, when the revocation information is transported in the form of a signed CRL, the CRACA certificate is only valid if it either signed the CRL itself, or issued the certificate that signed the CRL.

5.1.3.3 Identification of CRACA certificate

Change the text of 5.1.3.3 as follows:

A certificate contains a *cracald* field as specified in 6.4.8. This is an octet string of length 3. The relevant CRACA certificate is the certificate in the full chain for which the low-order three bytes of its SHA-256 hash are equal to the *cracald*. The hash of the certificate is obtained as specified in 6.4.3.

A *cracald* of all 0s and a *CrlSeries* value of 0 indicates that the certificate will not be revoked, i.e., that there is no revocation list that it will appear on. This may be because ~~the certificate~~ the certificate has a very short lifetime or for some other reason.

A *cracald* of all 0s and a non-zero *CrlSeries* value indicates that the certificate will appear on a CRL signed by itself.

If a certificate has a non-zero *cracald*, and the *cracald* is not matched by a unique certificate in the full chain (i.e., either it is not matched at all, or it is matched by more than one certificate), then the number of certificates in the chain that match the *cracald* is anything other than 1, the certificate is invalid.

5.1.3.6 Dubious certificates

Change the text of 5.1.3.6 as follows:

For each known (CRACA, CRL Series) pair, the SSME maintains an *expected update* time, i.e., the time when the revocation information issuer has indicated that revocation information is going to be updated. This value is set to “undefined” if the SSME has never received revocation information for that (CRACA, CRL Series) pair. The expected update time for revocation information contained in a CRL is given in the *nextCrl* field.

A certificate is considered by the SSME to be a *dubious certificate* if either no revocation information is available for that certificate, or the expected update time for that revocation information is in the past.

If queried about the revocation status of a dubious certificate via SSME-CertificateInfo.request, the SSME indicates that the certificate is dubious via SSME-CertificateInfo.confirm.

Any certificate in the full chain associated with a signed SPDU might potentially be dubious. The primitive Sec-SecureDataPreprocessing.confirm indicates the earliest ~~nextCrlTime~~ time associated with any certificate in the full chain associated with a signed SPDU. If that time is in the past, the certificate is considered dubious.

The standard provides the following mechanisms to handle the case where the SDS determines that a SPDU signed with a dubious certificate would be valid if the certificate was known not to be revoked, i.e., it passes all checks except that its revocation status is undetermined:

- **Use Overdue CRL Tolerance within SDS:** The SDS may be passed a parameter Overdue CRL Tolerance via Sec-SignedDataVerification.request. In this case, if the earliest *nextCrl* time for any certificate in the full chain is in the past by more than Overdue CRL Tolerance, the SDS indicates that the signed SPDU is invalid. If the parameter is not passed, the SDS indicate as valid a signed SPDU that meets all other validity conditions, regardless of the *nextCrl* time values.
- **SDEE-specific processing:** The SDEE may alternatively obtain the earliest *nextCrl* time for any certificate in the full chain via Sec-SecureDataPreprocessing.request, Sec-SecureDataPreprocessing.confirm. How a SDEE handles dubious certificates is SDEE-specific.

5.2 Signed SPDU validity

5.2.1 General

Change the text of 5.2.1 as follows, and insert the indicated figures:

A signed SPDU for a SDEE is valid for use by a receiving SDEE if all of the following hold:

The SDS supports two forms of signed SPDU: SPDUs signed with a certificate, and self-signed SPDUs. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state which form or forms are permitted for that SDEE. It is strongly recommended that SDEE specifications permit only SPDUs signed with a certificate.

A signed SPDU is valid for use by a receiving SDEE if all of the following hold:

- The SPDU meets a set of conditions that depend only on information sent by the sender, referred to as *consistency conditions*. These are discussed in 5.2.3.
- The SPDU meets other criteria, referred to as *relevance conditions*, which take into account the local time, location, and other state of the receiving SDEE. These are discussed in 5.2.4.
- The SPDU contains no unsupported *critical information fields*. Critical information fields are information fields that are necessary to determine whether a SPDU is valid. This is discussed in 5.2.5.

Consistency conditions make use of the claimed generation time and location of the signed SPDU. Relevance conditions make use of the claimed generation time and location of the signed SPDU, and the current time and location of the receiving SDEE. Time and location measurement requirements for the SDS are discussed in 5.2.2.

Figure 10a illustrates the different validity conditions used to determine the validity of a signed SPDU that is signed with a certificate, and the input information used to check against those validity conditions. Figure 10b shows the information fields that go into creating a signed SPDU that is signed with a certificate.

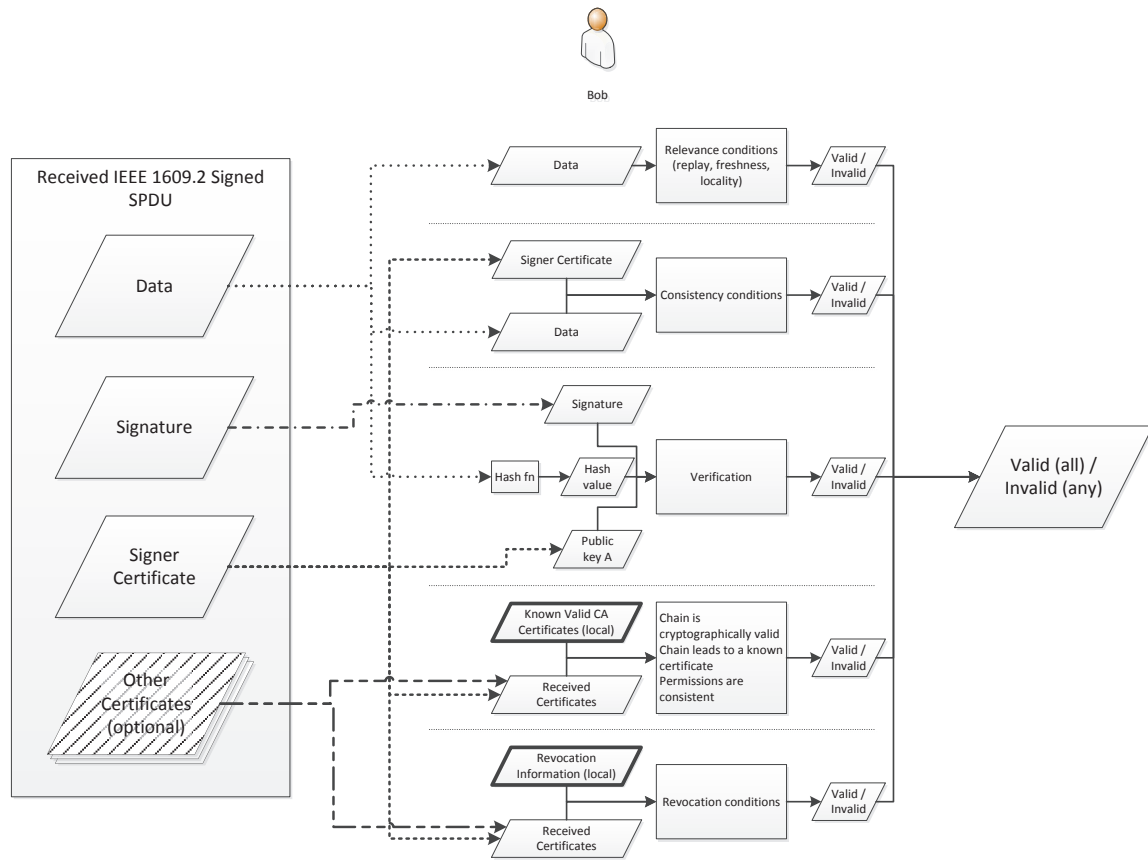


Figure 10a—Validity conditions for a signed SPDU

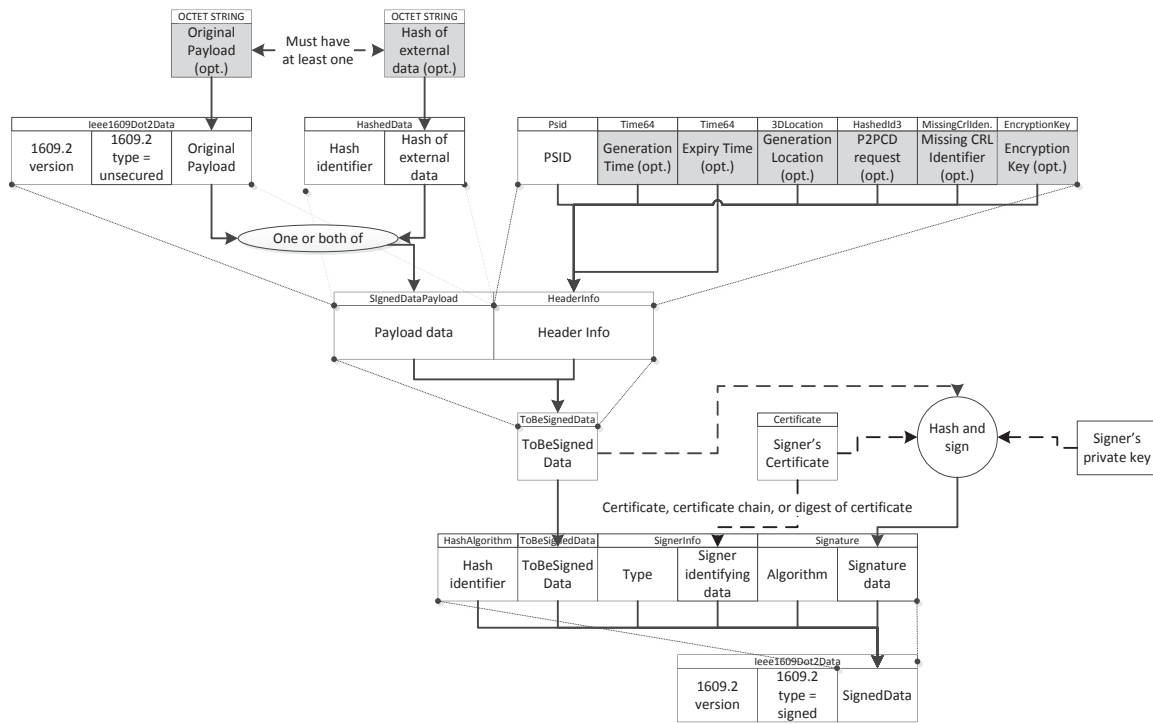


Figure 10b—Decomposition of a signed SPDU

5.2.3 Consistency conditions

5.2.3.1 General

There are two types of consistency conditions; *global consistency conditions* which do not depend on the specific SDEE that consumes the SPDU, and *SDEE-specific consistency conditions* which do depend on the receiving SDEE but not on its local conditions.

5.2.3.2 Global consistency conditions

5.2.3.2.1 General

Change the text of 5.2.3.2.1 as follows:

Global consistency conditions are:

- The SPDU is correctly formed using the data structures of Clause 6.
- The signature on the SPDU verifies with the appropriate certificate or public key, as specified in 5.2.3.2.1a.
- The SPDU is internally consistent, as specified in 5.2.3.2.3.
- EITHER the SPDU is self-signed and the SDEE specification permits self-signed SPDUs;

- OR the SPDU is signed with a certificate and all of the following conditions hold:
 - There is a certificate chain that leads from the signing certificate to a known trust anchor, constructed as specified in 5.1.2.1, such that:
 - All of the certificates in the chain are correctly formed using the data structures of Clause 6.
 - All certificates in the chain pass cryptographic verification with the appropriate public keys as specified in 5.1.2.3.
 - The certificate chain is internally consistent as specified in 5.1.2.4.
 - None of the certificates in the chain have been revoked as specified in 5.1.3.
 - The PDU is consistent with the signing certificate:
 - The permissions indicated by the security envelope are consistent with the permissions in the signing certificate, and the security envelope is consistent with itself, as specified in 5.2.3.2.2.
 - The PDU cryptographically verifies with the appropriate public keys. The cryptographic operations used for signing and verification are specified in 5.3.1. The encoding of data structures for input to those cryptographic operations is defined in Clause 6.
- ~~The PDU is internally consistent, as specified in 5.2.3.2.3.~~

~~If a signed SPDU does not meet all of these conditions it is invalid.~~

If a signed SPDU does not meet the first three conditions and either the fourth (if it is self-signed) or the fifth (if it is signed with a certificate), it is invalid.

Insert 5.2.3.2.1a as follows:

5.2.3.2.1a Signature verification

If the signature on a signed SPDU does not pass cryptographic verification, the SPDU is invalid. Signature generation and verification is specified in 5.3.1.

In the case of a signed SPDU signed with a certificate, the certificate to use to verify the signature is indicated using the SignerIdentifier structure within the SignedData as specified in Clause 6.

In the case of a self-signed SPDU, the public key is not transported in the IEEE 1609.2 security envelope. In this case, the means by which the receiving SDEE obtains the public key are part of the SDEE specification.

5.2.3.2.2 Consistency between signed SPDU and signing certificate

Change the opening text of 5.2.3.2.2 as follows:

A signed SPDU that is signed with a certificate contains the following information elements that are used when determining validity:

- Required:
 - Identifier of signing certificate

- Associated PSID
- One of: Encapsulated payload or hash of external payload
- Optional:
 - Generation location (see 5.2.2)
 - Generation time (see 5.2.2)
 - Expiry time

Change the second set of bullet points in 5.2.3.2.2 as follows (only the changed bullet points are provided, all others are unchanged):

A signed SPDU is consistent with the signing certificate if all the following hold:

- [...]
- The stated generation location, if present, is consistent with the geographic validity region indicated in the certificate, i.e., one of the following conditions holds:
 - [...]
- ~~The stated generation time~~ generation time is available (either from the headers of the signed SPDU, or obtained by other means such as from the SPDU payload by the SDEE and provided to the SDS) and is within the validity period of the certificate.

5.2.3.3 SDEE-specific consistency conditions

5.2.3.3.1 General

Change 5.2.3.3.1 as follows:

SDEE-specific consistency conditions are:

- The Provider Service ID (PSID) in the SPDU is consistent with any other PSID that the SDEE associates with the received PDU as specified in 5.2.3.3.2. This condition is checked by Sec-SignedDataVerification.request if the SDEE so requests and provides the appropriate PSID in that request.
- (If signed with a certificate) The payload of the PDU is consistent with the permissions (PSID, SSP, assurance level) in the signing certificate as specified in 5.2.3.3.3. This condition cannot be verified by the SDS and is intended to be verified by the receiving SDEE.
- Any external data included in the calculation of the signature has the correct hash value as specified in 5.2.3.3.4. This condition cannot be verified by the SDS and is intended to be verified by the receiving SDEE.
- ~~(Optionally)~~ If signed with a certificate The number of certificates in the full chain, from end entity ending in the SPDU-signing certificate to root certificate inclusive, is less than some SDEE-specific limit (see Annex C). This condition can be verified by the SDS.
- (If signed with a certificate) If the signed SPDU is making a statement about a geographic region other than a single point, that region is contained within the validity region of the certificate as specified in 5.2.3.3.4a. This condition is not verified by the SDS and is intended to be verified by the receiving SDEE.

5.2.3.3.3 Consistency between SPDU payload and permissions: Service Specific Permissions

Change the opening sentence of 5.2.3.3.3 as follows:

A valid signed SPDU that is signed with a certificate satisfies the following conditions that address consistency of the PDU payload with the sender's permissions.

Insert 5.2.3.3.4a and 5.2.3.3a after 5.2.3.3.4:

5.2.3.3.4a Consistency between SPDU payload and permissions: Relevance region

If a signed SPDU was signed with a certificate, the generation location consistency conditions specified in 5.2.3.2.2 can be used to determine that the signed SPDU was generated in a location where the generating SDEE is permitted to operate. However, depending on the application use case, generation location might not need to be the subject of a consistency check, and there also might be other geographic information for which it is appropriate to require authorization. An example of generation location not needing to be checked is certificate revocation list (CRL) generation activity following the specification in Clause 7; in this case, the location at which the CRL is generated is not germane to whether or not the CRL is valid, and the CRL could in fact be generated in a physical location that is outside any region that the revoked certificates would have been valid in. An example of other geographic information for which it is appropriate to require authorization is given by the Signal Phase and Timing (SPaT) message defined in SAE J2735 [B20]. In this case, the SPaT message can include information about signal phase and timing at multiple intersections, and it is appropriate to require that all the locations about which the message makes statements are permitted by the certificate.

Consistency between relevance areas in the payload and the certificate is SDEE-specific and outside the scope of this standard. The Sec-SecureDataPreprocessing.request returns the geographic region associated with a certificate, and the SDEE is expected to carry out any consistency checks necessary to determine that the payload is consistent with that geographic region. The 1609.2 security profile specified in Annex C can be used to note that additional geographic consistency checks are to be carried out; however, the details of these geographic consistency checks should be defined as part of the specification of the SDEE.

5.2.3.3a Identified regions

In a valid signed SPDU signed with a certificate, both of the conditions below hold:

- The geographic validity regions in each subordinate certificate are consistent within the chain, meaning that each validity region in a subordinate certificate is wholly contained in the validity region of its issuing certificate.
- If so specified by the SDEE specification, the generation location or relevance region of the SPDU is consistent with the validity region of the SDEE's certificate, meaning that the generation location or relevance region is respectively inside or wholly contained within that validity region.

This standard allows multiple approaches to indicate a validity region in a certificate. One of these approaches is to include an identifier for the region in the certificate, such that the SDS maps from the region identifier to a representation of the region which may be used for validity checking. The accuracy of this representation is addressed below in this subclause.

The IdentifiedRegion identifier may be drawn from one of a number of dictionaries. The permitted dictionaries are specified in 6.4.22 and the subclauses immediately thereafter.

The Protocol Implementation Conformance Statement (PICS) proforma given in Annex A allows the vendor of an implementation of WAVE Security Services to state whether any identified regions are supported, and to indicate which particular regions are supported in the sense that the WAVE Security Services have access to a map from that identifier to a region representation.

In claiming support of a particular region identifier *RId*, contained in one of the supported dictionaries and representing a region *R*, the PICS for an implementation is indicating that the following conditions hold:

- The region representation for *R* enables all consistency conditions with respect to identifiers in the same dictionary to be carried out with respect to *R*, i.e.:
 - In addition to supporting *RId* in this sense, the implementation supports all identifiers in the dictionary that identify a region that fully contains *R*.
 - For each region that fully contains *R*, the representation of the containing region fully contains the representation of *R*.
 - For each region that does not fully contain *R*, the representation of that region does not fully contain the representation of *R*.

The 1609.2 security profile is provided to enable SDEE specifiers to specify whether the SDEE should use the identified region type, and if so the representation accuracy requirements that apply (see Annex C).

5.2.4 Relevance conditions

5.2.4.2 SDS-verified relevance conditions

5.2.4.2.1 General

Change the opening paragraph and bulleted list in 5.2.4.2.1 as follows:

~~The following relevance conditions depend on the local state of the receiving SDEE and can be checked by the SDS. Whether or not any or all of these relevance conditions apply—and if they apply what parameters are used with them—is SDEE specific and is intended to be part of the SDEE specification. The 1609.2 security profile is provided to enable SDEE specifiers to specify the relevance conditions that apply (see Annex C). The relevance conditions are specified in more detail in subsequent subclauses. A signed SPDU is valid only if it is valid with respect to each specified relevance condition as defined in the appropriate subclause below.~~

- ~~**Future generation:** The signature generation time is not in the future.~~
- ~~**Freshness:** The signature generation time is not too long ago (for some PSID-specific definition of “not too long ago”).~~
- ~~**Expiry:** The signed data has not expired.~~
- ~~**Location:** The generation location is within a location tolerance *tl* of the receiver’s location.~~
- ~~**Replay:** The PDU is not a duplicate of a PDU acted upon by that SDEE in the recent past (for some PSID-specific definition of “the recent past”).~~
- ~~**Certificate expiry:** None of the certificates in the chain leading to the signed data have expired.~~

The following relevance conditions depend on the local state of the receiving SDEE and can be checked by the SDS. Whether or not any or all of these relevance conditions apply—and if they apply what parameters are used with them—is SDEE-specific and is intended to be part of the SDEE specification. The 1609.2

security profile is provided to enable SDEE specifiers to specify the relevance conditions that apply (see Annex C). The relevance conditions are specified in more detail in subsequent subclauses. A signed SPDU received by a given SDEE is valid only if it is valid with respect to each of the relevance conditions appropriate to that SDEE. The possible relevance conditions are as follows.

- **Freshness:** The signature generation time is not too far in the past, relative to the receiving SDS's estimate of the current time, for some definition of "too far in the past" given in the SDEE specification.
- **Future generation:** The signature generation time is not (too far) in the future, relative to the receiving SDS's estimate of the current time, for some definition of "too far in the future" given in the SDEE specification.
- **Expiry:** The signed data has not expired relative to the receiving SDS's estimate of the current time. Whether or not to carry out this check is specified in the SDEE specification.
- **Location:** The generation location is not too far away from the receiving SDS's estimate of its location, for some definition of "too far away" given in the SDEE specification.
- **Replay:** The PDU is not a replay of a PDU acted upon by that SDEE in the recent past, for some PSID-specific definition of "the recent past". Whether or not to carry out this check is specified in the SDEE specification.
- **Certificate expiry:** For an SPDU signed with a certificate, none of the certificates in the full chain ending with the certificate that signed the signed SPDU have expired relative to the receiving SDS's estimate of the current time. Whether or not to carry out this check is specified in the SDEE specification.

5.2.4.2.2 Generation time too far in the past

Change 5.2.4.2.2 as follows:

~~The following algorithm is defined to determine whether the generation time of a signed SPDU is too far in the past, for some SDEE specific (and possibly local condition specific) definition of "too far".~~

The SDS provides the service of checking whether a signed SPDU received by an SDEE has a generation time too far in the past. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state whether this service is used by that SDEE and, if so, to provide a definition of "too far in the past".

The following algorithm is defined to determine whether the generation time of a signed SPDU is too far in the past.

The difference between the local estimate of time at which the SPDU was received and the estimated generation time contained in that SPDU is calculated. If that difference exceeds V , the validity period associated with PDUs of the same type as that received, the PDU is invalid. Otherwise, the PDU is valid with respect to this relevance condition.

5.2.4.2.3 Generation time in the future

Change 5.2.4.2.3 as follows:

~~The following algorithm is defined to determine whether the generation time of a signed SPDU is too far in the future, for some SDEE specific (and possibly local condition specific) definition of "too far".~~

The SDS provides the service of checking whether a signed SPDU received by an SDEE has a generation time too far in the future. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state whether this service is used by that SDEE and, if so, to provide a definition of “too far in the future”.

The following algorithm is defined to determine whether the generation time of a signed SPDU is too far in the future.

The difference between the local estimate of time at which the SPDU was received and the estimated generation time contained in the SPDU is calculated. If that difference is less than zero, the PDU is invalid. Otherwise, the PDU is valid with respect to this relevance condition.

5.2.4.2.4 Expiry time

Change 5.2.4.2.4 as follows:

The SDS provides the service of checking whether a signed SPDU received by an SDEE has passed some expiry time stated in the SPDU. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state whether this service is used by that SDEE.

The following algorithm is defined to determine whether a signed SPDU should be considered expired.

The difference between the local estimate of time at which the SPDU was received and the expiry time contained in that SPDU is calculated. If that difference is greater than zero, the PDU is invalid. Otherwise, the PDU is valid with respect to this relevance condition.

5.2.4.2.5 Generation location too distant

Change 5.2.4.2.5 as follows:

The SDS provides the service of checking whether a signed SPDU received by an SDEE was generated too far away. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state whether this service is used by that SDEE and, if so, to provide the definition of “too far away”.

The following algorithm is defined to determine whether the estimated generation location of the message is too far away.

The distance between the estimated generation location of the signed SPDU and the receiver’s estimated location is calculated. If this distance is greater than D, the rejection threshold distance, the PDU is invalid. Otherwise, the PDU is valid with respect to this relevance condition.

5.2.4.2.6 Replay

Change the contents of 5.2.4.2.6 as follows:

~~The SDS can be requested to detect whether a signed SPDU is an exact duplicate of one previously processed by the SDS for that SDEE. The replay detection service is provided by SSME Sec ReplayDetection.request, SSME Sec ReplayDetection.confirm. The replay detection service indicates that a signed SPDU is a replay if the entire encoded signed SPDU, including signature and other fields such as generation time inserted by the SDS, is identical to a recently received SPDU. The definition of “recently received” is SDEE-specific, but it is a logically consistent choice for this value to be the same as the value used to determine whether a SPDU has a generation time too far in the past (see 5.2.4.2.2), and the~~

~~interfaces defined in this standard enforce that the two values are the same. Other replay detection techniques, such as ones based on the payload only or on the same data encoded in different ways, are out of scope of this standard.~~

~~Whether or not the SDS carries out replay detection, and the length of the interval for replay detection, is part of the SDEE specification. The 1609.2 security profile (see Annex C) is provided to specify replay detection behavior by the security services. Annex C also provides discussion of how to establish whether replay detection is important for a particular SDEE.~~

~~The SDS provides the service of checking whether a signed SPDU received by an SDEE is a replay, i.e. whether it is a duplicate of a signed SPDU recently processed by the SDS for that SDEE. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state whether this service is used by that SDEE. The definition of “recently processed” is SDEE-specific, but it is a logically consistent choice for this value to be the same as the value used to determine whether a SPDU has a generation time too far in the past (see 5.2.4.2.2), and the interfaces defined in this standard enforce that the two values are the same.~~

~~The replay detection service is provided by SSME-Sec-ReplayDetection.request, SSME-Sec-ReplayDetection.confirm. The replay detection service indicates that a signed SPDU is a replay if BOTH the COER encoding of the tbsData field canonicalized according to the encoding considerations given in IEEE 1609.2 subclause 6.3.6, AND the COER encoding of the Certificate that is to be used to verify the SPDU, canonicalized according to the encoding considerations given in subclause 6.4.3, are identical to those information elements for another recently received SPDU.~~

~~Other replay detection techniques, such as ones based on the payload only or on the same data encoded in different ways, are out of scope of this standard.~~

5.2.4.2.7 Certificate expiry

~~For an SPDU signed with a certificate, the SDS provides the service of checking whether any certificate in the chain of that signed SPDU has expired at the time the SPDU is verified. The 1609.2 security profile (see Annex C) is provided to enable an SDEE specification to state whether this service is used by that SDEE. Annex C also provides discussion of how to establish whether certificate expiry detection is important for a particular SDEE.~~

~~The following algorithm is defined to determine whether the certificates in the chain of a signed SPDU should be considered expired.~~

~~The pairwise difference between the local estimate of time at which the SPDU was received and the expiry time in each certificate in the full chain that signed that SPDU is calculated. If any of those differences is greater than zero, the PDU is invalid. Otherwise, the PDU is valid with respect to this relevance condition.~~

~~NOTE—Certificates that have expired are risky to trust. It is strongly recommended that a signed SPDU received after the expiry time of any certificate in its full chain be considered as invalid.~~

~~Certificates that have expired are risky to trust, since they are not guaranteed to appear on a CRL. It is strongly recommended that a signed SPDU received after the expiry time of any certificate in its chain be rejected.~~

~~The 1609.2 security profile (see Annex C) is provided to specify whether certificate expiry is checked by the security services. Annex C also provides discussion of how to establish whether certificate expiry detection is important for a particular SDEE.~~

5.2.5 Supported critical information fields

Change the text of 5.2.5 as follows:

Critical information fields are any fields necessary to establish the validity of a signed SPDU. An implementation of WAVE Security Services that cannot ~~parse~~interpret critical information fields in a signed SPDU or a certificate shall consider that signed SPDU or certificate to be invalid.

An implementation of WAVE Security Services might not be able to ~~parse~~interpret critical information fields for a number of reasons, including:

- The fields are too long.
- An array contains too many entries.
- A recursive structure contains too many recursions.
- A structure that uses identifiers includes an identifier that the implementation does not recognize.

For each data type defined in Clause 6 that may be of arbitrary length (in octets or number of entries) ~~or depth~~, the definition in Clause 6 specifies the circumstances under which it is a critical information field, and a minimum size to be supported by any conformant implementation of WAVE Security Services. The Protocol Implementation Conformance Statement (PICS) provided in Annex A allows an implementation to state any size it supports beyond the minimum required for conformance.

5.3 Cryptographic operations

5.3.1 Signature algorithms

Change the second paragraph of 5.3.1 as follows:

~~Two~~Three elliptic curves are specified for use with ECDSA: NIST P-256 as specified in FIPS 186-4, and (~~brainpoolP256r1~~, brainpoolP384r1) as specified in RFC 5639. Data structures and encoding rules for data objects associated with ECDSA are specified in Clause 6 of this standard and include an indication of which curve is applicable. A conformant implementation that supports signing or verification shall support at least one of these curves and may support more.

Change list entry c)2) in 5.3.1 as follows:

- 2) *Signer identifier input* depends on the verification type of the message
 - i) If the verification type is *certificate*, *signer identifier input* shall be the certificate with which the message is to be verified, canonicalized as specified in 6.4.3.
 - ii) If the verification type is *self-signed*, *signer identifier input* shall be the empty string, i.e., a string of length 0.

Delete the second-last paragraph of 5.3.1:

~~The encoding of the signature is specified in 6.3.28.~~

5.3.2 Implicit certificates

Change 5.3.2 as follows:

~~Implicit certificates were proposed in Brown, Gallant, and Vanstone [B3] and Pintsov and Vanstone [B18], and modifications to protect against attacks were proposed in Brown, Campagna, and Vanstone [B4].~~ In this standard, implicit certificates are processed as specified in Standards for Efficient Cryptography (SEC) 4 ~~except for~~ with the exceptions noted in this subclause.

- a) In this standard, an implicit certificate is ~~encoded as~~ an ImplicitCertificate, as defined in 6.4.5, encoded with the Canonical Octet Encoding Rules (COER). All references to “the certificate CertU” in SEC 4 should be taken as referring to the encoded ImplicitCertificate except ~~when in the instance~~ when in the instance the implicit certificate is hashed to an integer ~~modulus~~ modulo n ; this case is addressed in item b) below.
- b) When an implicit certificate is hashed to an integer modulo n , the input is not simply the implicit certificate CertU but the information specified below. This affects the following steps in SEC 4:
 - 1) Section 3.4, Action 7
 - 2) Section 3.5, Action 4
 - 3) Section 3.6, Action 2
 - 4) Section 3.7, Action 4
 - 5) Section 3.8, Action 4

The encoded data input to the hash function is Hash (ToBeSignedCertificate from the subordinate certificate as specified in 6.4.8, canonicalized as specified in 6.4.3) || Hash (Entirety of issuer certificate, canonicalized as specified in 6.4.3).

- c) SHA-256 shall be used as the Hash algorithm H.
- d) Within the integer hash function H_n , the output of the hash function H is not converted to an integer mod n using the mechanism specified in SEC 4, section 2.3. Instead, the hash function is converted to an integer by taking the 256-bit output from SHA-256, converting that bit string to an octet string using the Bit String To Octet String Conversion Primitive of SEC 1, and then converting that octet string to an integer using the Octet String To Integer Conversion Primitive of SEC 1.

This standard defines implicit certificates over the curves NIST P-256 and brainpoolP256r1. This standard does not define certificates over the curve brainpoolP384r1.

SHA-256 shall be used as the Hash algorithm H ~~within~~ used by the integer hash H_n specified in SEC 4, section 2.3.

The private key is judged as valid or invalid relative to an implicit certificate using the techniques of SEC 4 section 3.6.

Change the title and contents of 5.3.3 as follows:

5.3.3 Hash algorithms: SHA-256, SHA-384

The ~~only hash algorithm~~ algorithms approved for use in this standard ~~is~~ are SHA-256 and SHA-384 as specified in the Federal Information Processing Standard (FIPS) 180-4. In this standard, the phrase “the SHA-256 (resp. SHA-384) hash of [an octet string]” is used to mean “the hash of [that octet string] obtained using SHA-256 (resp. SHA-384) as specified in FIPS 180-4”.

5.3.5 Public key encryption algorithms: ECIES

Change the text of 5.3.5 as follows:

The only asymmetric encryption algorithm specified in this standard is the Elliptic Curve Integrated Encryption Scheme (ECIES) as specified in IEEE Std 1363a. This standard supports the use of ECIES to encrypt ephemeral data encryption keys as specified in 5.3.4.1 and does not support the use of ECIES to encrypt data directly.

Two elliptic curves are specified for use with ECIES: NIST P-256 as specified in FIPS 186-4, and brainpoolP256r1 as specified in RFC 5639. ~~Data structures and encoding rules for data objects associated with ECIES are specified in Clause 6 of this standard and include an indication of which curve is applicable.~~

~~In this standard, ECIES is used to encrypt symmetric keys. Data is encrypted with AES CCM (see 5.3.8). References to the “data encryption method” below are for consistency with IEEE Std 1363a.~~

When encrypting with ECIES, the following constraints on the specification in IEEE Std 1363a shall be applied.

NOTE—IEEE Std 1363a specifies the use of ECIES to encrypt data; in this standard, as noted above, ECIES is used only to encrypt symmetric keys. In the bulleted list below the word *data* is used to describe the plaintext input to encryption for consistency with IEEE Std 1363a, even though in the case of this standard the input is in fact a key.

- ~~— The public key V shall be freshly generated for each encryption operation.~~
- ~~— a) The secret value derivation primitive shall be Elliptic Curve Secret Value Derivation Primitive–Diffie-Hellman version with cofactor multiplication (ECSVDP-DHC).~~
- b) Compatibility with the corresponding -DH primitive shall not be desired.
- ~~— c) The data encryption method shall be a stream cipher based on Key Derivation Function 2 (KDF2) which shall be parameterized by the choice:~~
 - ~~— Hash = SHA-256~~
 - ~~— PI: recipient information, see below~~
- ~~— d) The data authentication code shall be MAC1 which shall be parameterized by the choices:~~
 - ~~— Input key length = 256 bits~~
 - ~~— Hash = SHA-256~~
 - ~~— $tBits$ = 128~~
 - ~~— P2 = the empty string~~
- ~~— e) Encryption shall use non-DHAES mode. This means that the elliptic curve points shall be converted to octet strings using LSB compressed representation.~~
- f) Data structures and encoding rules for data objects associated with ECIES are specified in Clause 6 of this standard and include an indication of which curve is used.

The ephemeral public key V shall be freshly generated for each encryption operation, i.e., an encryption operation shall not reuse an ephemeral public key V .

The parameter PI is a hash of the information that was bound to the ECIES key used for the encryption:

- If the encryption key was obtained from a certificate c , $P1$ is SHA-256 (c). ~~The), where c is the COER encoding of the certificate shall be put in, canonicalized form when hashing; see per 6.4.3.~~
- If the encryption key was obtained from a SignedData within an Ieee1609Dot2Data d (i.e., the encryption key is `d.signedData.tbsData.headerInfo.encryptionKey.public`), $P1$ is SHA-256 (d). ~~where d is the COER encoding of the Ieee1609Dot2Data, canonicalized per 6.3.4.~~
- If the encryption key was obtained from a different source, $P1$ is SHA-256 (“”, the empty string).

How a SDEE obtains encryption keys, and which form the parameter $P1$ takes, is SDEE-specific. See Annex C.7 for guidance on when different approaches to obtaining the encryption key may be appropriate. The data structures in Clause 6 allow the sender of an encrypted message to indicate the source of the encryption key to the recipient.

The output of this encryption is a triple (V , C , tag), where:

- V is an octet string representing the sender’s ephemeral public key.
- C is the encrypted symmetric key.
- tag is the authentication tag.

~~NOTE— The case where $P1$ is the hash of the empty string is defined only for use in responses to anonymous certificate requests and is not recommended for use in any other case. It should only be used if the SignedData is not available as it potentially allows misbinding attacks.~~

Example test vectors for ECIES are provided in Annex D.6.2.

Example test vectors for MAC1 are provided in Annex D.6.3.

Example test vectors for KDF2 are provided in Annex D.6.4.

5.3.8 Symmetric algorithms: AES-CCM

Insert the following text at the end of 5.3.8:

Example AES-CCM test vectors are provided in D.6.1.

6. Data structures

6.1 Presentation and encoding

Change the third and fourth paragraphs of 6.1 as follows:

There are some data structures in this standard for which a “canonical encoding” is defined. This is the encoding to be used whenever the structures are to be encoded for processing by a cryptographic hash function. In general, these are structures that include the output of some cryptographic operation, for which the generator of the structure may choose either to include additional information to speed up receive-side processing, or to omit that additional information and reduce the transmitted packet size. ~~The structures for which a canonical encoding is defined are SignedData, ToBeSignedData, HeaderInfo, HashedId8EcdsaP256Signature, EcdsaP384Signature, CertificateBase, and~~

~~CertificateBaseToBeSignedCertificate~~. Any structure for which encoding is subject to canonicalization has a paragraph entitled **Encoding considerations** in its description in Clause 6.

~~Clause 6 specifies and describes the data structures one at a time.~~ The complete IEEE 1609.2 ASN.1 modules are given in Annex B. In the event of a conflict between Annex B and this clause, this clause takes precedence.

Change the title of 6.2 as follows:

6.2 Integer Basic types

6.3 Secured protocol data units (SPDUs)

6.3.4 SignedData

Change the text of 6.3.4 as follows:

```
SignedData ::= SEQUENCE {  
    hashId          HashAlgorithm,  
    tbsData          ToBeSignedData,  
    signer          SignerIdentifier,  
    signature        Signature  
}
```

In this structure:

- hashId indicates the hash algorithm to be used to generate the hash of the message for signing and verification.
- tbsData contains the data that is hashed as input to the signature.
- signer determines the keying material and hash algorithm used to sign the data
- signature contains the digital signature itself, calculated as specified in 5.3.1, ~~with:~~
 - If signer indicates the choice self, then the signature calculation is parameterized as follows:
 - Data input is equal to the COER encoding of the tbsData field canonicalized according to the encoding considerations given in 6.3.6.
 - Verification type is equal to self.
 - Signer identifier input is equal to the empty string.
 - If signer indicates certificate or digest, then the signature calculation is parameterized as follows:
 - Data input is equal to the COER encoding of the tbsData field canonicalized according to the encoding considerations given in 6.3.6.
 - Verification type is equal to certificate.

- *Signer identifier input* is equal to the COER encoding of the Certificate that is to be used to verify the SPDU, canonicalized according to the encoding considerations given in 6.4.3.

6.3.5 HashAlgorithm

Change the ASN.1 code and first paragraph of 6.3.5 as follows:

```
HashAlgorithm ::= ENUMERATED {
    sha256,
    ...,
    sha384
}
```

This structure identifies a hash algorithm. The ~~only value currently supported is sha256, indicating~~ indicates SHA-256 as specified in 5.3.3. The value sha384 indicates SHA-384 as specified in 5.3.3.

Critical information fields: This is a critical information field as defined in 5.2.5. An implementation that does not recognize the enumerated value of this type in a signed SPDU when verifying a signed SPDU shall indicate that the signed SPDU is invalid.

6.3.9 HeaderInfo

Change the ASN.1 code, the indicated bullet points, and the “encoding considerations” paragraph in 6.3.9 as follows:

```
HeaderInfo ::= SEQUENCE {
    psid                      Psid,
    generationTime            Time64 OPTIONAL,
    expiryTime                Time64 OPTIONAL,
    generationLocation        ThreeDLocation OPTIONAL,
    p2pcdLearningRequest      HashedId3 OPTIONAL,
    missingCrlIdentifier       MissingCrlIdentifier OPTIONAL,
    encryptionKey             EncryptionKey OPTIONAL,
    ...,
    inlineP2pcdRequest        SequenceOfHashedId3 OPTIONAL,
    requestedCertificate       Certificate OPTIONAL,
}
```

- p2pcdLearningRequest, if present, is used by the SDS to request certificates for which it has seen identifiers but ~~does~~ does not know the entire certificate. A specification of this peer-to-peer certificate distribution (P2PCD) mechanism is given in Clause 8. This field is used for the out-of-band flavor of P2PCD and shall only be present if inlineP2pcdRequest is not present. The HashedId3 is calculated with the whole-certificate hash algorithm, determined as described in 6.4.3.
- missingCrlIdentifier, if present, is used by the SDS to request CRLs which it knows to have been issued but ~~have~~ has not received. This is provided for future use and the associated mechanism is not defined in this version of this standard.
- encryptionKey, if present, is used to indicate that a further ~~communication~~ communication should be encrypted with the indicated key. One possible use of this key to encrypt a response is specified in 6.3.33, 6.3.34, and 6.3.36. An encryptionKey field of type symmetric should only be used if the SignedData containing this field is securely encrypted by some means.

- inlineP2pcdRequest, if present, is used by the SDS to request unknown certificates per the inline peer-to-peer certificate distribution mechanism is given in Clause 8. This field shall only be present if p2pcdLearningRequest is not present. The HashedId3 is calculated with the whole-certificate hash algorithm, determined as described in 6.4.3.
- requestedCertificate, if present, is used by the SDS to provide certificates per the “inline” version of the peer-to-peer certificate distribution mechanism given in Clause 8.

Encoding considerations: When the structure is encoded in order to be digested to generate or check a signature, if `encryptionKey` is present, and indicates the choice public, and contains a `BasePublicEncryptionKey` that is an elliptic curve point (i.e., of type `EccP256CurvePoint` or `EccP384CurvePoint`), then the elliptic curve point is encoded in compressed form, i.e., such that the choice indicated within the `EccP256CurvePoint`/`EccP384CurvePoint` is `compressed-y-0` or `compressed-y-1`.

~~**Critical information fields:** This structure contains no critical information fields in the sense defined in 5.2.5~~

6.3.16 MissingCrlIdentifier

Change the text of 6.3.16 as follows:

```
MissingCrlIdentifier ::= SEQUENCE {  
    cracaId          HashedId3,  
    crlSeries        CrlSeries,  
    ...  
}
```

This structure may be used to request a CRL that the SSME knows to have been issued but has not yet received. It is provided for future use and its use is not defined in this version of this standard.

- `cracaId` is the HashedId3 of the CRACA, as defined in 5.1.3. The HashedId3 is calculated with the whole-certificate hash algorithm, determined as described in 6.4.3.
- `crlSeries` is the requested CRL Series value. See 5.1.3 for more information.

6.3.19 SymmetricEncryptionKey

Delete the following paragraph:

Critical information fields: If present, this is a critical information field as defined in 5.2.5. An implementation that does not recognize the indicated CHOICE for this type when verifying a signed SPDU shall indicate that the signed SPDU is invalid.

6.3.21 SymmAlgorithm

Delete the following paragraph:

Critical information fields: If present, this is a critical information field as defined in 5.2.5. An implementation that does not recognize the indicated CHOICE for this type when verifying a signed SPDU shall indicate that the signed SPDU is invalid.

6.3.22 BasePublicEncryptionKey

Delete the following paragraph:

Critical information fields: If present, this is a critical information field as defined in 5.2.5. An implementation that does not recognize the indicated CHOICE for this type when verifying a signed SPDU shall indicate that the signed SPDU is invalid.

6.3.23 EccP256CurvePoint

Change 6.3.23 as follows:

```
EccP256CurvePoint ::= CHOICE {  
    x-only          OCTET STRING (SIZE (32)),  
    fill            NULL, -- consistency w 1363 / X9.62  
    compressed-y-0  OCTET STRING (SIZE (32)),  
    compressed-y-1  OCTET STRING (SIZE (32)),  
    uncompressedP256 SEQUENCE {  
        x OCTET STRING (SIZE (32)),  
        y OCTET STRING (SIZE (32))  
    }  
}
```

This structure specifies a point on an elliptic curve in Weierstrass form defined over a 256-bit prime number. This encompasses both NIST p256 as defined in FIPS 186-4 and Brainpool p256r1 as defined in RFC 5639. The fields in this structure are OCTET STRINGS produced with the elliptic curve point encoding and decoding methods defined in subclause 5.5.6 of IEEE Std 1363-2000. The x-coordinate is encoded as an unsigned integer of length 32 octets in network byte order for all values of the CHOICE; the encoding of the y-coordinate y depends on whether the point is x-only, compressed, or uncompressed. If the point is x-only, y is omitted. If the point is compressed, the value of type depends on the LSB-least significant bit of y: if the LSB-least significant bit of y is 0, type takes the value compressed-y-0, and if the LSB-least significant bit of y is 1, type takes the value compressed-y-1. If the point is uncompressed, y is encoded explicitly as an unsigned integer of length 32 octets in network byte order.

Insert new 6.3.23a:

6.3.23a EccP384CurvePoint

```
EccP384CurvePoint ::= CHOICE {  
    x-only          OCTET STRING (SIZE (48)),  
    fill            NULL, -- consistency w 1363 / X9.62  
    compressed-y-0  OCTET STRING (SIZE (48)),  
    compressed-y-1  OCTET STRING (SIZE (48)),  
    uncompressedP384 SEQUENCE {  
        x OCTET STRING (SIZE (48)),  
        y OCTET STRING (SIZE (48))  
    }  
}
```

This structure specifies a point on an elliptic curve in Weierstrass form defined over a 384-bit prime number. The only supported such curve in this standard is Brainpool p384r1 as defined in RFC 5639. The fields in this structure are OCTET STRINGS produced with the elliptic curve point encoding and decoding methods defined in subclause 5.5.6 of IEEE Std 1363-2000. The x-coordinate is encoded as an unsigned