

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

IEEE Vehicular Technology Society

Sponsored by the
Intelligent Transportation Systems Committee

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

IEEE Std 1609.2™-2013
(Revision of
IEEE Std 1609.2-2006)

26 April 2013

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

Sponsor

Intelligent Transportation Systems Committee
of the
IEEE Vehicular Technology Society

Approved 6 February 2013

IEEE-SA Standards Board

Approved 27 October 2014

American National Standards Institute

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 in this standard. It also describes administrative functions necessary to support the core security functions.

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

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

Copyright © 2013 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 26 April 2013 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 978-0-7381-8287-2 STD98169
Print: ISBN 978-0-7381-8288-9 STDPD98169

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.

Notice and Disclaimer of Liability Concerning the Use of IEEE Documents: IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. 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.

Use of an IEEE Standard is wholly voluntary. IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon any IEEE Standard document.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied "AS IS."

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. 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 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.

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 the official position of IEEE or any of its committees and shall not be considered to be, nor 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 to ensure 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. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE working group at <http://standards.ieee.org/develop/wg/>.

Comments on standards should be submitted to the following address:

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

Photocopies: Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, 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.

Notice to users

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

This document is copyrighted by the IEEE. It is made available 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 this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE 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. 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://standards.ieee.org/index.html> or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit IEEE-SA Website at <http://standards.ieee.org/index.html>.

Errata

Errata, if any, for this and all other standards can be accessed 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 standard was completed, the Dedicated Short Range Communication Working Group had the following membership:

Thomas M. Kurihara, *Chair*
William Whyte, *Vice Chair*

Scott Andrews
Lee Armstrong
Roger Berg
William Brownlow
Susan Dickey
Hans-Joachim Fischer
Wayne Fisher
Jon Friedman
Ramez Gerges
Paul Gray
Gloria Gwynne
Ron Hochnadel
Stanley Hsu

Carl Kain
Doug Kavner
David Kelley
John Kenney
Jeremy Landt
Michael Li
Chih-Che (Mike) Lin
Julius Madey
Alastair Malarky
Justin McNew
John Moring
Satoshi Oyama

Gary Pruitt
James D. Randall
Steve Randall
Güner Refi-Tugrul
Randal Roebuck
Richard Roy
Steve Sill
François Simon
Hsieh Tien-Yuan
George Vlantis
Timothy Weil
Andre Weimerskirch
Aaron Weinfield

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

Iwan Adhicandra
Lee Armstrong
H. Stephen Berger
Mark Bushnell
William Byrd
Kai T. Chen
Keith Chow
Michael Coop
Patrick Diamond
Sourav Dutta
Richard Edgar
Wayne Fisher
Andre Fournier
Ramez Gerges
Paul Gray
Ron Greenthaler
Randall Groves
Tugrul Guener
Gloria Gwynne
Ronald Hochnadel
Werner Hoelzl
Chung-Hsien Hsu

David Hunter
Noriyuki Ikeuchi
Piotr Karocki
John Kenney
Stuart Kerry
Max Kicherer
Thomas M. Kurihara
Paul Lambert
Richard Lancaster
Jeremy Landt
Hsia-Hsin Li
William Lumpkins
Greg Luri
Julius Madey
Alastair Malarky
Sean Maschue
Edward McCall
Justin McNew
John Moring
Michael S. Newman
Satoshi Obara

Satoshi Oyama
Venkatesha Prasad
Markus Riederer
Robert Robinson
Randal Roebuck
Richard Roy
Randall Safier
Bartien Sayogo
Gil Shultz
Di Dieter Smely
Walter Struppler
Dale Sumida
Jasja Tijink
Thomas Tullia
Dmitri Varsanofiev
John Vergis
George Vlantis
Stephen Webb
Hung-Yu Wei
Andre Weimerskirch
William Whyte
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 6 February 2013, it had the following membership:

John Kulick, *Chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Masayuki Ariyoshi
Peter Balma
Farooq Bari
Ted Burse
Wael William Diab
Stephen Dukes
Jean-Phillippe Faure
Alexander Gelman

Mark Halpin
Gary Hoffman
Paul Houzé
Jim Hughes
Michael Janezic
Joseph L. Keopfinger*
David J. Law
Oleg Logvinov

Ron Peterson
Gary Robinson
Jon Walter Rosdahl
Adrian Stephens
Peter Sutherland
Yatin Trivedi
Phil Winston
Yu Yuan

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Michelle Turner
IEEE Standards Program Manager, Document Development

Michael Kipness
IEEE Standards Program Manager, Technical Program Development

Introduction

This introduction is not part of IEEE Std 1609.2-2013, IEEE Standard for Wireless Access in Vehicular Environments—Security Services for Applications and Management Messages.

5.9 GHz Dedicated Short Range Communications for Wireless Access in Vehicular Environments (DSRC/WAVE, hereafter simply WAVE), as specified in a range of standards including those generated by the IEEE P1609 working group, enables vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) wireless communications. This connectivity makes possible a range of applications that rely on communications between road users, including vehicle safety, public service, commercial fleet management, tolling, and other operations.

With improved communications come increased risks, and the safety-critical nature of many WAVE applications makes it vital that services be specified that can be used to protect messages from attacks such as eavesdropping, spoofing, alteration, and replay. Additionally, the fact that the wireless technology will be deployed in personal vehicles, whose owners have a right to privacy, means that inasmuch as possible the security services should respect that right and not leak personal, identifying, or linkable information to unauthorized parties.

With this in mind, at the time that IEEE P1609 was established to develop the standards for the WAVE wireless networking protocols, the IEEE also established IEEE P1556™ (later renumbered as IEEE 1609.2) to develop standards for the security techniques that will be used to protect the services that use these protocols. These applications face unique constraints. Many of them, particularly safety applications, are time-critical: the processing and bandwidth overhead due to security must be kept to a minimum, to improve responsiveness and decrease the likelihood of packet loss. For many applications, the potential audience consists of all vehicles on the road in North America; therefore, the mechanism used to authenticate messages must be as flexible and scalable as possible, and must accommodate the smooth removal of compromised WAVE devices from the system. Additionally, as mentioned above, the privacy of privately owned and operated vehicles must be respected as far as technically and administratively feasible.

This document specifies a range of security services for use by WAVE devices. Mechanisms are provided to authenticate WAVE management messages, to authenticate messages from non-anonymous users, and to encrypt messages to a known recipient. Mechanisms to provide anonymous authentication, particularly anonymous broadcast, will be provided in a separate document.

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Document organization	2
1.4 Document conventions	2
1.5 Note to implementers	2
2. Normative references	2
3. Definitions, abbreviations, and acronyms	3
3.1 Definitions	3
3.2 Abbreviations and acronyms	8
4. General description	10
4.1 WAVE protocol stack overview	10
4.2 Generic security services	12
4.3 Security processing services	12
4.4 Cryptomaterial	14
4.5 Security management services	16
5. Security services	18
5.1 General	18
5.2 Preconditions for secure processing	18
5.3 Secure data exchange	23
5.4 Signed WSAs	27
5.5 Validity of signed communications	29
5.6 Processing for security management	43
5.7 Certificate Management Entity	51
5.8 Cryptographic operations	53
6. Data structures for secure communication	55
6.1 Presentation language	55
6.2 Structures for secure communications	65
6.3 Certificates and other security management data structures	78
7. Service primitives and functions	102
7.1 General comments and conventions	102
7.2 Sec SAP	105
7.3 WME-Sec SAP	150
7.4 PSSME SAP	159
7.5 CME SAP	166
7.6 PSSME-Sec SAP	179
7.7 CME-Sec SAP	183
7.8 Internal functions	184
Annex A (normative) Protocol Implementation Conformance Statement (PICS) proforma	200
A.1 Instructions for completing the PICS proforma	200
A.2 PICS proforma—IEEE Std 1609.2	202
Annex B (informative) IEEE 1609.2 security profiles	212
B.1 General	212

B.2 Secure data exchange	213
B.3 IEEE 1609.2 security profile proforma	217
Annex C (normative) IEEE 1609.2 security profile for specific use cases	220
C.1 SAE J2735 Basic Safety Message	220
C.2 WSA	222
Annex D (informative) Example and Use Cases	224
D.1 Examples of encoded data structures	224
D.2 Secure data reception	228
D.3 Certificate request	231
D.4 Signed WSA: full example with certificate request and WSA processing	242
D.5 Processing CRLs	253
D.6 Constructing a certificate chain	255
Annex E (informative) Rationale and FAQ	260
E.1 Introduction	260
E.2 General philosophy	260
E.3 System assumptions made in this standard	263
E.4 Cryptography	264
E.5 Secure data exchange	267
E.6 Signed WSAs	269
E.7 Certificate request	272
E.8 CRL use	273
E.9 Security mechanisms not included in this standard	273
Annex F (informative) Copyright statement for 6.1	276
Annex G (informative) Bibliography	277

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

IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

1.1 Scope

This standard defines 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. It also describes administrative functions necessary to support the core security functions.

1.2 Purpose

The safety-critical nature of many WAVE applications makes it vital that services be specified that can be used to protect messages from attacks such as eavesdropping, spoofing, alteration, and replay. Additionally, the fact that the wireless technology will be deployed in communication devices in personal vehicles as well as other portable devices, whose owners have an expectation of privacy, means that in as much as possible the security services must be designed to respect privacy and not leak personal, identifying, or linkable information to unauthorized parties. This standard describes security services for WAVE management messages and application messages designed to meet these goals.

1.3 Document organization

Clause 1 provides an overview of the document. Clause 2 contains the normative references. Clause 3 contains definitions and abbreviations. Clause 4 provides a general description of WAVE Security Services and their use. Clause 5 specifies how WAVE Security Services may be used by calling entities to execute specific security use cases. Clause 6 specifies the encoding and structure of messages generated and consumed by WAVE Security Services. Clause 7 defines the primitives used to communicate between WAVE Security Services and other functional entities.

Annex A provides a PICS proforma. Annex B provides a description of the IEEE 1609.2 security profile and a proforma that may be used by developers of applications (or other entities that invoke WAVE Security Services) to specify options for how those applications are to interact with WAVE Security Services. Annex C provides an IEEE 1609.2 security profile for the Basic Safety Message (BSM) specified by the Society of Automotive Engineers (SAE). Annex D provides examples, including examples of encoded protocol data unit (PDU) and sample process flows from the point of view of an entity invoking WAVE Security Services rather than from the point of view of WAVE Security services. Annex E provides an informative rationale and FAQ. Annex F is a copyright statement covering the text from Clause 6 that was excerpted from other sources. Annex G provides an informative bibliography.

1.4 Document conventions

Unless otherwise stated, conventions follow those in IEEE Std 802.11™ [B10]¹, including conventions for the ordering of information elements within data streams.

Numbers are decimal unless otherwise noted. Numbers preceded by 0x are to be read as hexadecimal, so that 0xFF is equivalent to “FF hexadecimal.” Occasionally, this standard includes representations of octet strings in hexadecimal form; these strings are indicated as hexadecimal on a case-by-case basis.

Figures are used for illustration and are informative, unless otherwise noted.

1.5 Note to implementers

Future versions of this standard may differ significantly from the current version. In particular, international harmonization efforts, advances in cryptanalysis, and other considerations make it likely that future versions of this standard will not preserve backwards compatibility with the current version. Changes of data structures in future versions of the standards will be reflected by changes in the version numbers within those data structures.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard, October 2008.²

Federal Information Processing Standard (FIPS) 186-3, Digital Signature Standard.

¹ The numbers in brackets correspond to those of the bibliography in Annex G.

² FIPS publications are available from the National Technical Information Service (NTIS), U. S. Dept. of Commerce, 5285 Port Royal Rd., Springfield, VA 22161 (<http://www.ntis.org/>).

IEEE Std 1363™-2000, IEEE Standard Specifications for Public Key Cryptography.^{3, 4}

IEEE Std 1363a™-2004, IEEE Standard Specifications for Public Key Cryptography: Additional Techniques.

IEEE P1609.0™ /D1.0, April 2011, Draft Guide for Wireless Access in Vehicular Environments (WAVE) – Architecture.⁵

IEEE Std 1609.3™-2010, Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services.

IEEE Std 1609.12™-2012, Standard for Wireless Access in Vehicular Environments (WAVE) – Identifier Allocations.

IETF Request for Comments: 3629, UTF-8, A Transformation Format of ISO 10646.⁶

NIMA Technical Report TR8350.2, “Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems.”⁷

NIST Special Publication SP 800-38C, Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality.⁸

Standards for Efficient Cryptography Group, “SEC 1: Elliptic Curve Cryptography version 2.0.”⁹

Standards for Efficient Cryptography Group, “SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV),” Working Draft Version 0.97, March 2011.

3. Definitions, abbreviations, and acronyms

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.¹⁰

Advanced Encryption Standard (AES): A U.S. government standard specifying a symmetric block cipher; also, the block cipher specified in that standard.

asymmetric cryptographic algorithm: A cryptographic algorithm that uses two related keys, a public key and a private key, such that the public key is derived from the private key but, given only the public key, it is computationally infeasible to derive the private key.

³ IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08855-1331, USA (<http://standards.ieee.org/>).

⁴ The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

⁵ This IEEE standards project was not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining a draft, contact the IEEE.

⁶ IETF publications are available from <http://www.ietf.org>.

⁷ This technical report is available from http://earth-info.nga.mil/GandG/tr8350/tr8350_2.html.

⁸ NIST special publications are available from <http://csrc.nist.gov/publications/nistpubs/>.

⁹ SECG publications are available from <http://www.secg.org>

¹⁰ *IEEE Standards Dictionary Online* subscription is available at:

http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

authenticated channel: A logical communications channel such that the receiver has assurance that the sender is who they claim to be, and that modifications to the data can be detected. A channel may be authenticated by applying cryptographic mechanisms to the channel itself, or by checking the transmitted protocol data units by some out-of-band mechanism.

authentication: A cryptographic service that provides assurance that the sender of a communication is who they claim to be.

Basic Safety Message (BSM): A message, described in SAE J2735, used to announce the status of an object such as a vehicle, pedestrian, or roadside worker.

block cipher: A symmetric encryption algorithm that processes data in blocks, typically of 8 or 16 octets.

calling entity: A functional entity that invokes WAVE Security Services.

certificate: *See digital certificate.*

certificate authority (CA): An entity that issues certificates. A correctly functioning certificate authority has procedures in place to ensure that certificates are only issued to entities that are entitled to them.

certificate authority certificate (CA certificate): A certificate that is used to verify other certificates; a type of *security management certificate*.

certificate chain: An ordered list of certificates such that each certificate in the list (except for the first) was issued by the certificate before it in the list. *See also: full certificate chain; partial certificate chain.*

certificate holder: The entity authorized to use a particular digital certificate to establish trust. The certificate holder can carry out operations using the private key corresponding to the certificate's public key.

Certificate Management Entity (CME): The management entity responsible for managing the certificate information and corresponding key information on a device.

Certificate Revocation List (CRL): A list identifying certificates that have been revoked. *See: revocation.*

Certificate Revocation List distribution center (CRL distribution center): A networked entity that stores and distributes certificate revocation lists (CRLs).

Certificate Revocation List series (CRL series): An integer used to assign different certificates issued by the same certificate authority to distinct sets, such that the certificates in different sets appear on different revocation lists if revoked.

Certificate Signing Request (CSR): A communication from an entity to a certificate authority, requesting that the certificate authority issue a certificate on behalf of the entity.

communications certificate: A certificate used to sign either secure data or a *WAVE Service Advertisement*.

computationally infeasible: An operation that requires an exceptionally large amount of computing power to complete. The threshold for computational infeasibility will rise over time as computing power at a given price point increases.

Counter Mode with Cipher Block Chaining Message Authentication Code (CCM): A mode of operation of a block cipher in which the data is encrypted with a keystream, which in turn is generated by encrypting an incrementing counter, and in turn authenticated with a message authentication code calculated using cipher block chaining mode.

cryptographic verification: The process of determining whether a signature on a signed communication is consistent with the communication and the private key.

cryptographically secure hash function: A function that maps an arbitrary-length input into a fixed-length output (the hash value) such that (a) it is computationally infeasible to find an input that maps to a specific hash value and (b) it is computationally infeasible to find two inputs that map to the same hash value. All hash functions used in this document are cryptographically secure hash functions.

cryptomaterial: A private key, a public key, or a certificate.

Cryptomaterial Handle: A reference to a private key and the associated public key or certificate, stored within the security processing services.

decrypt: To convert unreadable, encrypted data to readable, decrypted data using a decryption algorithm and a key. *Contrast:* **decode, encrypt.**

decryption algorithm: An algorithm that takes as input ciphertext and a key and (if the correct key is provided) produces the original plaintext.

delta Certificate Revocation List (delta CRL): A certificate revocation list that carries information only about certificates that were revoked within a certain time period. *Contrast:* **full Certificate Revocation List.**

digital certificate: A digitally signed document binding a public key to an identity and/or a set of permissions.

dubious certificate: A certificate for which the most recent certificate revocation list has not yet been received.

Elliptic Curve Cryptography (ECC): A form of public-key cryptography based on the problem of finding discrete logarithms in a group defined over elliptic curves.

Elliptic Curve Digital Signature Algorithm (ECDSA): A specific digital signature mechanism based on the elliptic curve discrete logarithm problem.

Elliptic Curve Integrated Encryption Scheme (ECIES): A public-key encryption mechanism based on the elliptic curve discrete logarithm problem.

encode: To convert a data structure into an array of octets. *Contrast:* **decode, encrypt.**

encrypt: To convert readable data to unreadable, encrypted data using an encryption algorithm and a key. *Contrast:* **decrypt, encode.**

encryption algorithm: An algorithm that takes as input plaintext and a key and produces ciphertext.

encryption certificate: A certificate that contains an encryption key.

end-entity certificate: Any certificate other than one used by a CA for certificate management operations.

enrolment certificate: A certificate used to sign a Certificate Signing Request. A type of *security management certificate*. *Contrast:* **operational certificate**.

explicit certificate: A certificate that contains a public key and the certificate authority's signature

full certificate chain: A certificate chain in which the first certificate is a root certificate and the last certificate is an end-entity certificate.

full Certificate Revocation List (full CRL): A certificate revocation list that carries information about certificates that were revoked and have not expired, whenever the revocation took place. *Contrast:* **delta Certificate Revocation List**.

hash function: *See:* **cryptographically secure hash function**.

hash value: The output of a hash function.

IEEE 1609.2™ security profile: A specification of the options and parameters provided to the 1609.2 security services by a particular consumer of those services.

implicit certificate: A digital certificate that does not explicitly contain the certificate holder's public key, but instead allows the public key to be reconstructed from a reconstruction value and the certificate authority's public key

inherited permissions: Permissions within a subordinate certificate that are communicated by reference to the issuing certificate, rather than stated explicitly within the subordinate certificate.

issuing certificate: If the public key from certificate A can be used to verify the signature on certificate B, then A is the *issuing certificate* for B. *Contrast:* **subordinate certificate**.

Local Service Index for Security (LSI-S): An identifier used to identify a secure communications entity to the security services.

operational certificate: Any certificate other than an enrolment certificate.

operational permissions: The actions a certificate holder is allowed to take as stated in their certificate. Expressed in this standard using provider service identifiers.

partial certificate chain: A certificate chain that is not a full certificate chain (i.e., the first certificate is not a root certificate or the last certificate is not an end-entity certificate).

plaintext: Unencrypted data.

properly formed certificate: A certificate that can be parsed correctly according to the data structures and encoding defined in this standard.

provider: Advertiser of a WAVE service. *See also:* **user**; **WAVE Service Advertisement**.

Provider Service Identifier (PSID): An octet string that identifies a service provided by a higher layer entity.

public-key digital signature: A cryptographically secure checksum that is generated using a private key and verified using a public key.

reconstruction value: A value in an implicit certificate that allows the associated public key to be recovered.

replay attack: An attack in which an attacker retransmits or delays data that was originally validly transmitted.

revocation: The publication by a relevant authority of the information that a particular certificate is no longer to be trusted.

root certificate: A self-signed certificate that can be used as a trust anchor to verify other certificates.

secure communications entity: A functional entity that uses IEEE 1609.2 services to secure communications for any purpose other than security management.

secure data exchange certificate: A certificate that is used to verify the signature on data other than a *WAVE service advertisement*. *Contrast:* **WSA certificate, communications certificate.**

Secure Data Exchange Entity (SDEE): A secure communications entity that uses IEEE 1609.2 services to secure any communications other than *WAVE Service Advertisements*.

secure provider service: A higher layer entity that requests that its entry in a WAVE Service Advertisement be secured.

security management: Operations that support acquiring or establishing the validity of 1609.2 certificates.

security management certificate: A certificate used to authenticate security management protocol data units. *Enrolment certificates* and *certificate authority certificates* are security management certificates. *Contrast:* **communications.**

security management message: A protocol data unit used to manage certificates or information about certificates.

self signed certificate: A certificate whose signature can be verified with the public key in the certificate.

Service Specific Permissions (SSP): A field that encodes permissions relevant to a particular certificate holder.

ServiceInfo: A field in an IEEE Std 1609.3 *WAVE service advertisement* (WSA) that announces the availability of a service. A WSA may contain multiple ServiceInfo fields.

signature: See **public key digital signature**.

store: A collection of data objects (typically certificates and certificate revocation lists in this standard) such that each object stays in the store until it expires, is superseded, or is revoked.

symmetric cryptographic algorithm: A cryptographic algorithm that uses a single key. Knowledge of a symmetric encryption key allows both encryption and decryption. Knowledge of a symmetric authentication key allows generation and verification of message authentication codes.

symmetric key: *See:* **symmetric cryptographic algorithm**.

Transport Layer Security (TLS): An Internet Engineering Task Force (IETF) standard providing for secure communications over Transmission Control Protocol/Internet Protocol (TCP/IP).

trust anchor: A certificate whose validity does not depend on the validity of other certificates.

user: Consumer of a WAVE service. See also: **provider; WAVE Service Advertisement**.

valid certificate: A certificate that is correctly formed, that has not been revoked or expired, and for which a certificate chain to a trust anchor can be constructed.

vehicle-to-infrastructure: Wireless communication between vehicles and roadside units in either direction

Wireless Access in Vehicular Environments (WAVE) device: A device that is compliant to IEEE Std 1609.3™, IEEE Std 1609.4™ and IEEE Std 802.11™, operating outside the context of a basic service set (BSS). (See IEEE Std 802.11 specification of “STA transmission of data frames outside the context of a BSS.”)

Wireless Access in Vehicular Environments (WAVE) Management Entity (WME): A set of management functions, defined in IEEE Std 1609.3, required to provide WAVE Networking Services.

Wireless Access in Vehicular Environments (WAVE) Service Advertisement (WSA): A collection of data indicating the services that a given WAVE device is offering.

Wireless Access in Vehicular Environments (WAVE) Service Advertisement (WSA) certificate: A certificate used to verify a wireless access in *WAVE Service Advertisement*.

Wireless Access in Vehicular Environments (WAVE) Short Message (WSM): A packet consisting of a data payload and a *WAVE Short Message Protocol* header.

Wireless Access in Vehicular Environments (WAVE) Short Message Protocol (WSMP): A low-overhead protocol for exchange of messages in a rapidly varying radio frequency environment where low latency is also an important objective.

3.2 Abbreviations and acronyms

AES	Advanced Encryption Standard
ASN.1	Abstract Syntax Notation 1
BSM	Basic Safety Message
CA	certificate authority
CCM	counter with cipher block chaining message authentication code
CME	Certificate Management Entity
CMH	Cryptomaterial Handle
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DSRC	Dedicated Short Range Communications
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme

FIPS	Federal Information Processing Standard
GPS	Global Positioning System
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol version 6
ITS	Intelligent Transportation Systems
LSI-S	Local Service Index for Security
MAC	medium access control, or message authentication code
NIST	National Institute for Standards and Technology
OBU	on-board unit
PDF	probability distribution function
PDU	protocol data unit
PHY	Physical Layer
PSC	Provider Service Context
PSID	Provider Service Identifier
PSSME	Provider Service Security Management Entity
RFC	Request for Comments
RSU	roadside unit
SAE	Society of Automotive Engineers
SAP	Service Access Point
SDEE	Secure Data Exchange Entity
SP	special publication
SSP	Service Specific Permissions
TAI	International Atomic Time
TLS	Transport Layer Security
TTP	Trusted Third Party
UTC	Coordinated Universal Time
UTF	Unicode Transformation Format

V2I	vehicle-to-infrastructure
V2V	vehicle-to-vehicle
WAVE	Wireless Access in Vehicular Environments
WGS	World Geodetic System
WME	WAVE Management Entity
WSA	WAVE Service Advertisement
WSM	WAVE Short Message
WSMP	WAVE Short Message Protocol

4. General description

4.1 WAVE protocol stack overview

Wireless Access in Vehicular Environments (WAVE) provides a communication protocol stack optimized for the vehicular environment, employing both customized and general-purpose elements as shown in Figure 1. See IEEE P1609.0¹¹ for an overview of the WAVE architecture and services, including a description of the use of the security services described in this document.

¹¹ Information on references can be found in Clause 2.

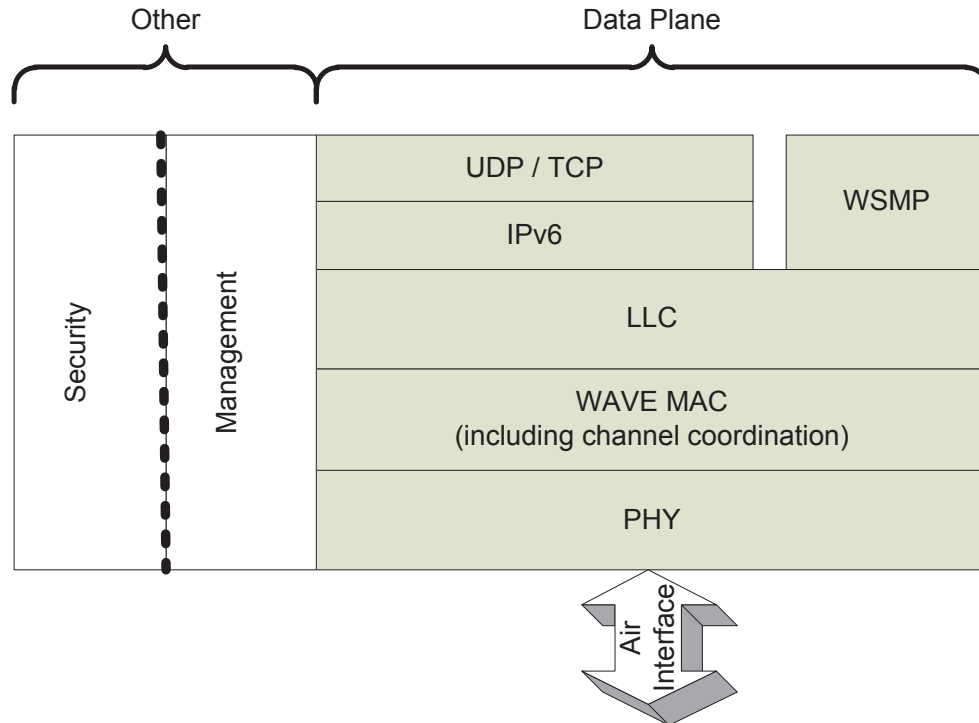


Figure 1—WAVE Protocol Stack

This standard specifies a set of management plane WAVE Security Services available to applications and processes running in the various layers of the WAVE protocol stack as shown in Figure 2. The WAVE Security Services consist of:

- *security processing services:*
 - provide processing that is performed to enable *secure communications that comprise* secure data and secure WAVE Service Advertisements (WSAs); and
- *security management services:*
 - provide certificate management services that are provided by the Certificate Management Entity (CME) and that manage information related to the validity of all certificates; and
 - provide service security management services that are provided by the Provider Service Security Management Entity (PSSME) and that manage information related to certificates and private keys that are used to send secured WSAs.

The services and entities within the WAVE Security Services are shown in Figure 2, which also shows Service Access Points (SAPs) that support communications between WAVE Security Services entities and other entities. This standard specifies the security processing via primitives defined at these SAPs. Horizontal boundaries within the WAVE Security Services in Figure 2 are abstract and do not correspond to horizontal boundaries within the Data Plane.

Data passed across the SAPs in this standard is assumed to be secure and trustworthy. This standard does not provide mechanisms to ensure the trustworthiness of this data.

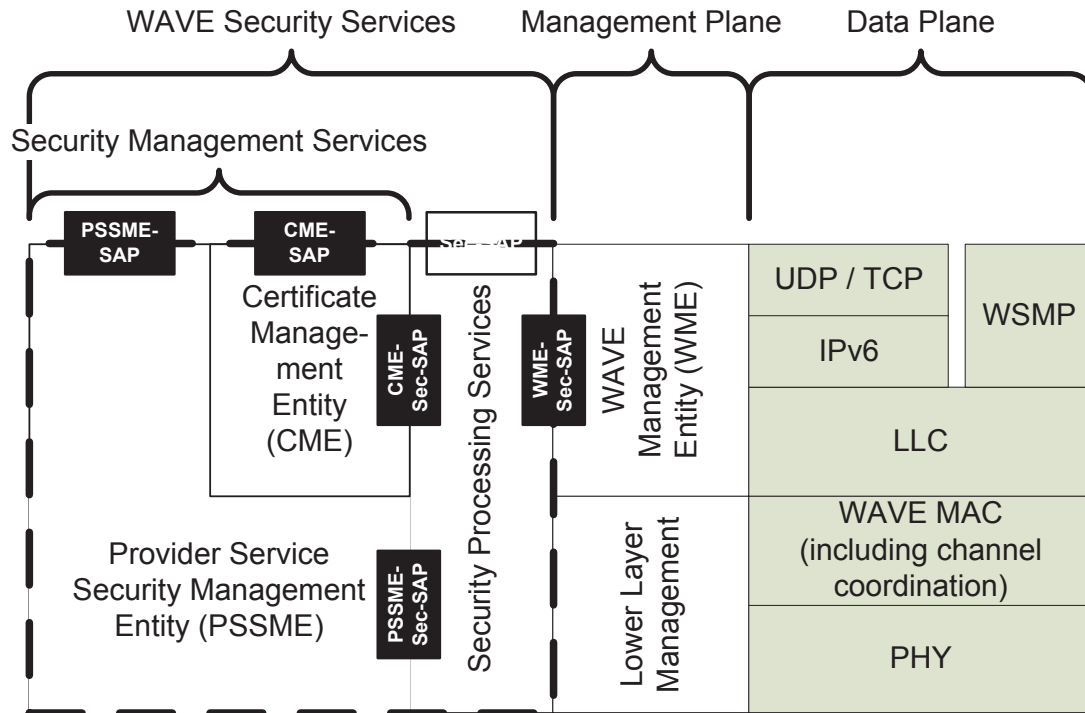


Figure 2—WAVE Protocol Stack showing detail of the security services

4.2 Generic security services

IEEE 1609 provides WAVE Security Services that in turn support the following generic security services:

- Confidentiality: Supported by the security processing services *Generate encrypted data* and *Decrypt encrypted data*.
- Authentication: Supported by the security processing services *Generate signed data* and *Verify signed data*.
- Authorization: Supported by the security processing services *Generate signed data* and *Verify signed data*.
- Integrity: Supported by the security processing services *Generate signed data* and *Verify signed data*.

4.3 Security processing services

4.3.1 General

The security processing services accept input requests from calling entities. The security processing services respond to each input request with a corresponding output confirmation containing the results of the request.

The security processing services consist of the following:

- Transforming unsecured Protocol Data Units (PDUs) into secured PDUs to be transferred using data-plane operations, and processing secured PDUs on reception, including transforming secured PDUs into unsecured PDUs. These processes together are referred to as “Secure data exchange.” The security services specified in this document support the signing and/or encryption of PDUs prior to their transmission, and the decryption and verification (as appropriate) of PDUs on reception.
- Generating secured WSAs and processing signed WSAs on reception (hereafter, “Signed WSAs”). WSAs are defined in IEEE Std 1609.3. They originate at the WAVE Management Entity (WME) in the transmitting device, and are consumed by the WME in the receiving device. The WME is also defined in IEEE Std 1609.3. The security services specified in this document support the signing of WSAs prior to their transmission and the verification of WSAs on reception.
- Security processing for security management: ensuring access to cryptomaterial (private keys, public keys and certificates) as described in 4.4; generating certificate requests and processing responses; and validating Certificate Revocation Lists (CRLs).
- Storing private keys and their associated certificates.

An implementation of WAVE security services shall include at least one of the following security processing services:

- Generate signed data (see 5.3.2)
- Generate encrypted data (see 5.3.3)
- Verify signed data (see 5.3.6)
- Decrypt encrypted data (see 5.3.5)
- Generate signed WSA (see 5.4.1)
- Verify signed WSA on reception (see 5.4.2)
- Generate certificate request (see 5.6.1)
- Verify response to certificate request (see 5.6.2)
- Verify certificate revocation list (see 5.6.4)

4.3.2 Secure data exchange

The secure data exchange operations are:

- Generate signed data (see 5.3.2)
- Generate encrypted data (see 5.3.3)
- Verify signed data (see 5.3.6)
- Decrypt encrypted data (see 5.3.5)

The information elements used by the secure data exchange operations are specified in 7.2 as parameters to primitives. The specifier of an application may use the *IEEE 1609.2 security profile* as a compact way to communicate secure data exchange parameters to multiple implementers. The IEEE 1609.2 security profile is a set of fixed parameters or parameter ranges to be supplied to the secure data exchange operations when applying security to PDUs for a specific application or service. IEEE 1609.2 security profiles are described in Annex B and may be specified using the proforma given in B.3. Annex C provides a normative example of an IEEE 1609.2 security profile for a specific calling entity, namely one that sends or receives the Basic Safety Message (BSM) as specified in the Society of Automotive Engineers (SAE) J2735 standard.

Communication of IEEE 1609.2 security profiles is outside the scope of this standard and is discussed in Annex B.

4.3.3 Signed WSAs

The Signed WSA operations are:

- Generate signed WSA (see 5.4.1)
- Verify signed WSA on reception (see 5.4.2)

The information elements used by the signed WSA operations are specified in 7.3.

4.3.4 Processing for security management

The processing for security management operations are:

- Generate certificate request (see 5.6.1)
- Verify response to certificate request (see 5.6.2)
- Verify certificate revocation list (see 5.6.4)

The information elements used by these operations are specified in 7.2.

4.4 Cryptomaterial

4.4.1 General

Cryptographic operations within the security processing services make use of data of the following types:

- Private keys and associated public keys.
- Private keys and associated certificates.
- Digital certificates held by peer entities, for which no private key is stored by the security processing services.

This data is collectively referred to as *cryptomaterial*.

Private keys and associated certificates are used by the security processing services in the following operations:

- Generate signed data
- Decrypt encrypted data on reception
- Generate signed WSA

WAVE security services shall reject a request to carry out an operation that uses a private key and certificate if the private key and certificate provided to the operation do not form a valid pair as defined in 5.8.4.

Private keys and associated public keys are used during certificate request as described in 5.6.1.1.

WAVE security services shall reject a request to carry out an operation that uses a private key and a public key if the private key and public key provided to the operation do not form a valid pair as defined in 5.8.4.

Certificates held by peer entities are used in the following operations:

- Verify signed data on reception
- Generate encrypted data
- Verify signed WSA on reception

Private keys are held by the security processing services and managed through cryptomaterial handles, specified in 4.4.2. Certificates held by peer entities are managed by the CME, specified in 4.5.1.

The cryptographic mechanisms supported by this standard are the Elliptic Curve Digital Signature Algorithm (ECDSA) and the Elliptic Curve Integrated Encryption Scheme (ECIES). Cryptographic mechanisms are specified in 5.8.

4.4.2 Private key storage and cryptomaterial handles

Within this standard, private keys and the associated public keys and certificates are stored within the security processing services and referenced using a Cryptomaterial Handle (CMH). A CMH is an index used to reference storage within the security processing services. A CMH references a private key and a public key, or a private key and a certificate. Mechanisms for initializing a CMH and storing keys and certificates using a CMH are specified in 5.2.3. These mechanisms ensure that a private key and a public key, or a private key and a certificate, referenced by a CMH always form a valid pair.

Each secure data exchange entity (SDEE) involved in a secure data exchange operation directly references its key and certificate using a CMH. This is illustrated in Figure 3, where the dashed lines indicate the functionality specified in this standard.

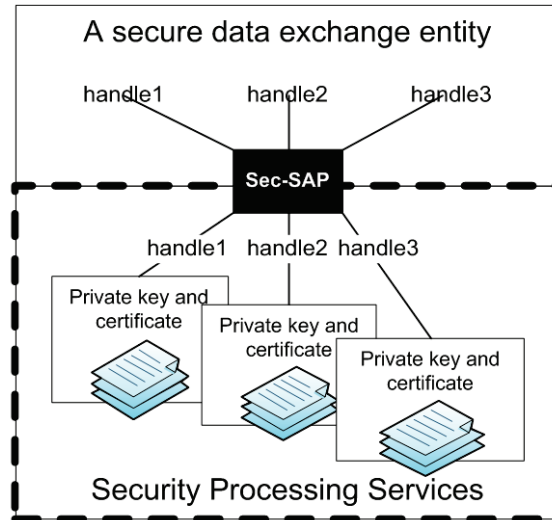


Figure 3—Secure data exchange entity and cryptomaterial handle

Private keys and certificates necessary to sign WSAs are stored and managed by the PSSME as described in 5.2.3. This is illustrated in Figure 4, where the dashed lines indicate the functionality specified in this standard. The PSSME references the private keys and certificates by means of CMHs.

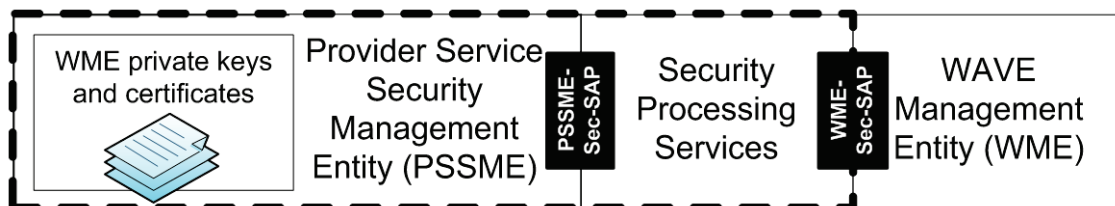


Figure 4—Relationship among WME, WSA private keys, and certificates, and the WAVE Security Services

NOTE—This standard does not provide a mechanism to remove CMHs or the keys associated with them if they are no longer needed. Implementations may provide this mechanism, for example to save storage space on the WAVE device, or to prevent the accidental use of expired or otherwise invalid cryptomaterial.

4.5 Security management services

4.5.1 Certificate Management Entity

The Certificate Management Entity (CME) manages the following information that allows the security processing services to determine the trustworthiness of certificates and received data:

- Revocation and other status information about certificates for which the corresponding private key is stored by the security processing services.
- Revocation and information about certificates for which the corresponding private key is not stored by the security processing services (i.e., those belonging to peer WAVE security services and to CAs).

- Certificates that may be used as a trust anchor.

Additionally, the CME manages information to allow it to determine whether or not a received secure PDU is an exact duplicate of a previously received PDU.

As illustrated in Figure 2, the CME has two Service Access Points (SAPs) through which other entities may communicate with it to obtain and update security management information: one for use by the security processing services (CME-Sec-SAP), and one for use by other entities (CME-SAP).

Certificate management information is generated by CAs and may be obtained by a number of different methods by entities that communicate with WAVE Security Services (see 5.6.4). This standard refers to certificates whose management information is available within the CME as certificates that are “known to” the CME.

NOTE—This standard specifies mechanisms for providing certificate management information to the CME but not for removing information that is no longer relevant. Implementations may implement such mechanisms.¹²

4.5.2 Provider Service Security Management Entity

The Provider Service Security Management Entity (PSSME) enables the sharing of security management information, including cryptomaterial, between the WME and higher layer entities that request to be advertised as secure provider services. The process of requesting to be advertised is specified in IEEE Std 1609.3. The PSSME provides the following functionality:

- Managing the assignment of a Local Service Index for Security (LSI-S) to allow the security processing services to distinguish between higher layer entities.
- Registration of permissions for secure provider services, to allow processes outside the PSSME to apply for WSA signing certificates.
- Communication of certificates and private keys to the security processing services, to allow the security processing services to sign WSAs on behalf of the WME.
- For each known WSA signing certificate, the PSSME may store the most recently accepted WSA signed by that certificate, to allow for fast acceptance of repeated WSAs.

PSSME operations are specified in 5.4 and 5.6.1. The PSSME’s relationship with the security processing services and with the WME is illustrated in Figure 4.

¹² Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

5. Security services

5.1 General

Figure 5 shows the general model for security processing. Security services are invoked by a secure communications entity (SCE) and return their output to the same SCE.

The sending entity (which may be an application, another higher layer entity, the WME or any other entity) invokes the security services to perform sender-side security processing. The results of the processing are returned to the sending entity, which then transmits the resulting application protocol data unit (APDU).

The receiving entity receives the secure APDU. It then invokes the security services to perform security processing on the contents of the APDU. The security services return the result to the receiving entity for further processing which may include multiple invocations of the security services if necessary.

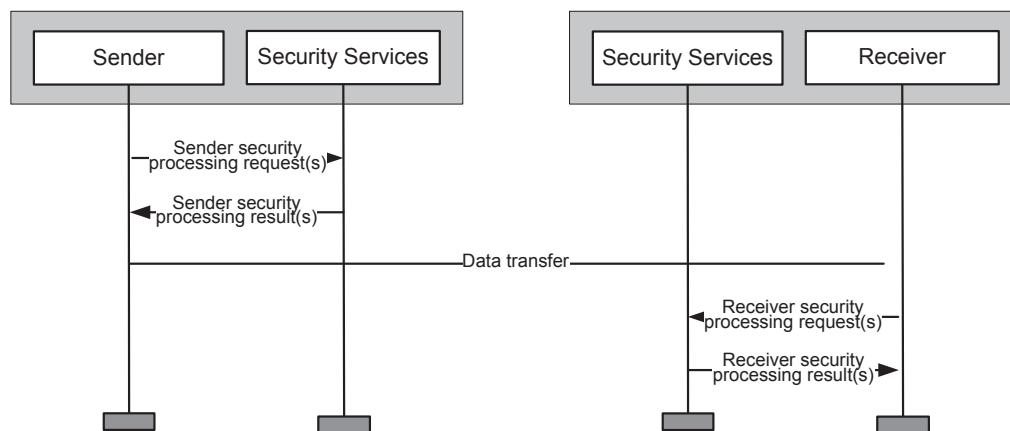


Figure 5—Process flow for use of 1609.2 security services

5.2 Preconditions for secure processing

5.2.1 Secure data exchange

The security processing services uniquely identify each SDEE using a Local Service Identifier for Security (LSI-S) which is an integer generated by the security processing services. The security services ensure that each LSI-S value is only allocated once. Implementations may choose a different mechanism to support unique identification of entities, as long as that mechanism supports all of the use cases for the LSI-S described in this standard.

NOTE—The LSI-S is only used by SDEEs that are going to request that the security processing services use replay protection on incoming messages. See E.5.8 for more discussion of replay protection. An SDEE that does not use replay protection on incoming messages need not request an LSI-S.

The process for obtaining an LSI-S is shown in Figure 6 where the dashed lines indicate functionality defined in this standard. Primitive names in the figure are abbreviated for compactness. The calling entity obtains its LSI-S via Sec-LocalServiceIndexForSecurity.request (Sec-LSIS.req in the diagram) and the LSI-S is returned via Sec-LocalServiceIndexForSecurity.confirm (Sec-LSIS.cfm).

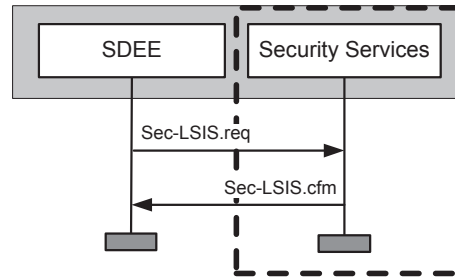


Figure 6—Process flow for obtaining a Local Service Identifier for Security (LSI-S) for a secure data exchange entity (SDEE)

5.2.2 Secure WSAs

5.2.2.1 Local service index for security

A higher layer entity that requests that its entry in a WSA is secured, is referred to as a *secure provider service*. The security processing services uniquely identify each secure provider service using a LSI-S which is a locally unique integer generated by the security processing services.

The calling entity obtains its LSI-S via PSSME-LocalServiceIndexForSecurity.request (PSSME-LSIS.req in the diagram) and the LSI-S is returned via PSSME-LocalServiceIndexForSecurity.confirm (PSSME-LSIS.cfm).

The process for obtaining an LSI-S is shown in Figure 7 where the dashed lines indicate functionality defined in this standard. Primitive names in the figure are abbreviated for compactness.

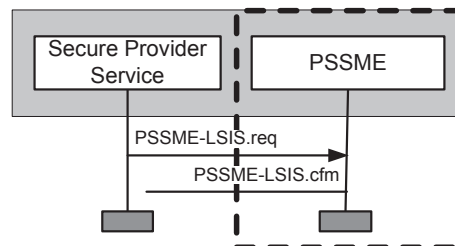


Figure 7—Process flow for obtaining a Local Service Identifier for Security (LSI-S) for a secure provider service

NOTE 1—The LSI-S for secure provider services and the LSI-S for secure data exchange are always used in different contexts and so in principle, a secure provider services LSI-S and a secure data exchange LSI-S may have the same value.

NOTE 2—The purpose of the LSI-S is to enable the WME to reference the secure provider service when requesting that the security processing services sign a WSA. This allows the security processing services to determine whether or not a valid WSA certificate exists for the combination of secure provider services in the WSA.

NOTE 3—The secure provider service relays the LSI-S to the WME via WME-ProviderService.request as defined in IEEE Std 1609.3.

NOTE 4—A sample process flow covering the inclusion of a secure provider service in a signed WSA, including certificate provision, is given in D.4.

5.2.2.2 Registering secure provider service permissions with PSSME

Each secure provider service has specified permissions associated with it. These are denoted by a Provider Service Identifier (PSID), a maximum allowable priority, and (optionally) a set of Service Specific Permissions (SSP). The mapping between permissions and certificates is managed by the PSSME.

The PSSME is requested to register secure provider service permissions via PSSME-SecuredProviderService.request. The information elements used by the PSSME are specified in 7.4.3.

The PSSME acknowledges the registration request via PSSME-SecuredProviderService.confirm.

The process for registering permissions is shown in Figure 8 where the dashed lines indicate functionality defined in this standard. Primitive names in the figure are abbreviated for compactness.

Sample process flows to obtain and use certificates for WSA signing are described in D.3.2, D.4.

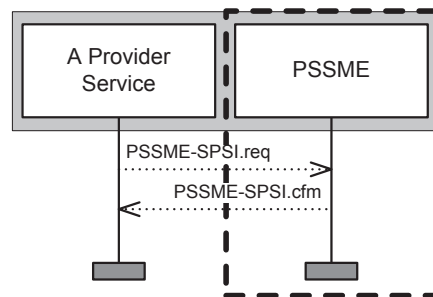


Figure 8—Process flow for registering secure provider service permissions

5.2.3 Cryptomaterial

5.2.3.1 General

In this standard, private keys and their associated public keys or certificates are stored internally to the security processing services. An invoking entity references a particular stored private key for use in a particular operation by passing a Cryptomaterial Handle (CMH) to the security processing services. The security processing services manage the state of the CMH in response to requests from external entities.

There are three states defined for a CMH, as follows:

- *Initialized*: A CMH in *Initialized* state does not reference any cryptomaterial.
- *Key Pair Only*: A CMH in *Key Pair Only* state references a private key and the corresponding public key.
- *Key and Certificate*: A CMH in *Key and Certificate* state references a private key and the corresponding certificate.

During the course of operations, the security processing services may directly access the certificate referenced by a CMH in *Key and Certificate* state or the public key referenced by a CMH in *Key Pair Only* state.

Figure 9 shows the state diagram for Cryptomaterial Handles, including an indication of the primitives that may be used to transition from one state to the next.

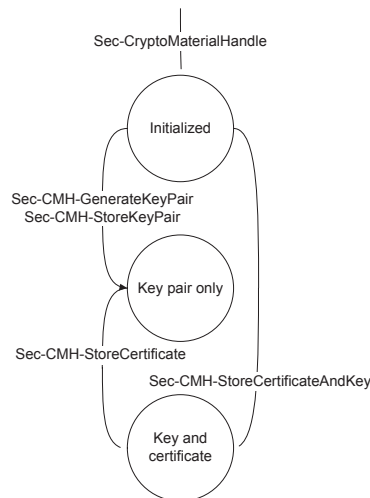


Figure 9—State diagram for Cryptomaterial Handles

5.2.3.2 Initialization

A CMH is created in the *Initialized* state as shown in Figure 10, where the dashed lines indicate functionality defined in this standard. Primitive names in the figure are abbreviated for compactness. The calling entity creates the CMH via Sec-CryptomaterialHandle.request (Sec-CMH.req in the diagram) and the CMH is returned via Sec-CryptomaterialHandle.confirm (Sec-CMH.cfm).

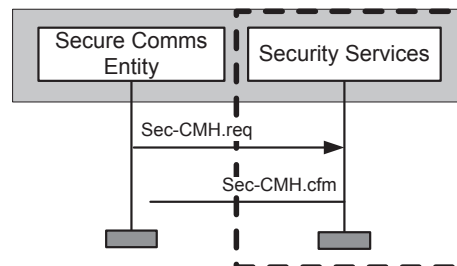


Figure 10—Process flow for obtaining a new Cryptomaterial Handle

5.2.3.3 Transition to Key Pair Only state

On invocation of Sec-CryptomaterialHandle-StoreKeyPair.request referencing a CMH that is in the *Initialized* state, the security processing services shall store the enclosed private key and place the CMH in the *Key Pair Only* state. The security processing services confirm the operation back to the invoking Secure Communications Entity via Sec-CryptomaterialHandle-StoreKeyPair.confirm. This is illustrated in Figure 11.

On invocation of Sec-CryptomaterialHandle-GenerateKeyPair.request for a CMH that is in the *Initialized* state, the security processing services shall generate a private key and public key pair, store them with the CMH and confirm the operation back to the invoking Secure Communications Entity in a Sec-CryptomaterialHandle-GenerateKeyPair.confirm, thus causing the CMH to enter the *Key Pair Only* state. This is illustrated in Figure 12.

A CMH shall only be placed in the *Key Pair Only* state if the private and public keys referenced by the CMH form a valid key pair for the given cryptographic algorithm. Key pair validity is established for ECDSA and ECIES as specified in 5.8.

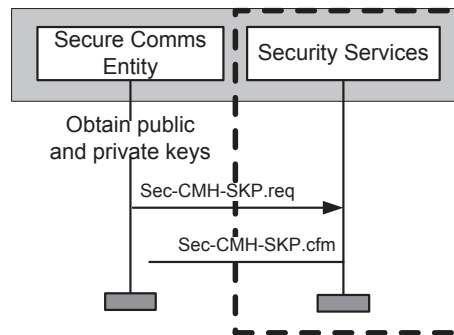


Figure 11—Transitioning a CMH from *Initialized* to *Key Pair Only* state with externally generated keys

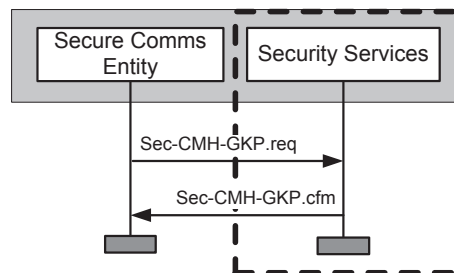


Figure 12—Transitioning a CMH from *Initialized* to *Key Pair Only* state with keys generated by the security services

5.2.3.4 Transition to *Key and Certificate* state

On invocation of Sec-CryptomaterialHandle-StoreCertificateAndKey.request for a CMH that is in the *Initialized* state, the security processing services shall store the enclosed private key and certificate and confirm the operation back to the invoking Secure Communications Entity via Sec-CryptomaterialHandle-StoreCertificateAndKey.confirm, thus causing the CMH to enter the *Key and Certificate* state. This is illustrated in Figure 13, where the primitive names are abbreviated to Sec-CMH-SCAK.req and Sec-CMH-SCAK.cfm respectively.

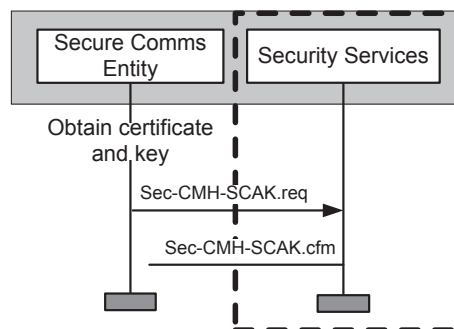


Figure 13—Transitioning a CMH directly from *Initialized* to *Key and Certificate* state

On invocation of a Sec-CryptomaterialHandle-StoreCertificate.request for a CMH that is in the *Key Pair Only* state, the security processing services shall store the enclosed certificate with the CMH and confirm the operation back to the invoking SCE in a Sec-CryptomaterialHandle-StoreCertificate.confirm, thus causing the CMH to enter the *Key and Certificate* state. This is illustrated in Figure 14.

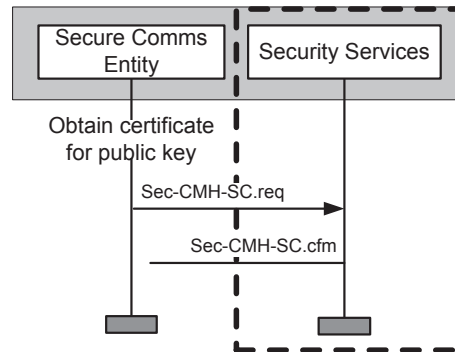


Figure 14 — Transitioning a CMH from *Initialized* to *Key and Certificate* state via *Key Pair Only* state with externally generated keys

A CMH shall only be placed in the *Key and Certificate* state if the private key and the public key indicated by the certificate form a valid key pair for the given cryptographic algorithm and if the certificate is part of a valid certificate chain terminating in a trust anchor. Key pair validity for ECDSA and ECIES may be assured using the processing specified for Sec-CryptomaterialHandle-StoreCertificate.request (7.2.9) and Sec-CryptomaterialHandle-StoreCertificateAndKey.request (7.2.11). The validity of the certificate chain shall be established according to the criteria given in 5.5.2. This may also be assured using the processing specified for Sec-CryptomaterialHandle-StoreCertificate.request and Sec-CryptomaterialHandle-StoreCertificateAndKey.request.

NOTE 1—This standard intentionally does not provide primitives that may be used to extract the private key from a CMH.

NOTE 2—This standard does not provide a primitive that allows a private key to be imported to a CMH in encrypted form, but implementations may provide such a primitive.

5.3 Secure data exchange

5.3.1 General

This section specifies the process flow for the following specific uses of the security processing services for secure data exchange:

- Sign data
- Encrypt data
- Sign and encrypt data
- Process received secure data

All of these uses follow the same template:

- A calling entity invokes the security services via a request primitive to request processing of specific data.
- The security services carry out the requested processing and return the result to the calling entity via a confirm primitive. This returned values include a result code indicating whether the request succeeded or failed.

NOTE—This standard uses the term “reject a request” as shorthand for “return an error code in the confirm primitive corresponding to the request.”

Actions taken by the calling entity before or after it invokes the security services are out of scope of this standard.

5.3.2 Sign data

The security processing services are requested to sign data via Sec-SignedData.request. The information elements used by the security processing services to generate signed data are specified in 7.2.13. These information elements include a CMH in *Key and Certificate* state.

The security processing services return the result of the request (either an octet string containing the signed data or an error code) to the SDEE via Sec-SignedData.confirm. On success, the octet string returned shall be a valid encoding of a 1609Dot2Data as defined in Clause 6.

The security processing services shall reject a request to sign data if the certificate is not currently valid or consistent with the data according to the criteria specified in 5.5 or if the CMH is not valid according to the criteria specified in 5.2.3.4, or if the CMH is not in *Key and Certificate* state. An implementation of WAVE security services may have an upper bound on full certificate chain length and may reject a request to sign data if the chain from the certificate to a trust anchor, as described in 5.5.2, is longer than that upper bound.¹³

The signing process is illustrated in Figure 15, where the dashed lines indicate entities and processing that are defined in this standard. Primitive names in the figure are abbreviated for compactness.

NOTE—Any changes made to the signed data after signing before sending it invalidate the digital signature, resulting in invalid signed data.

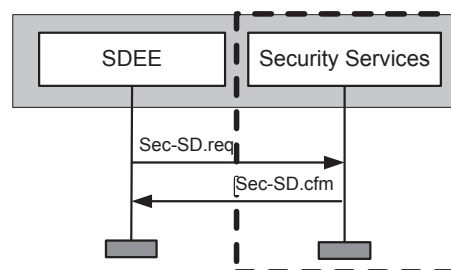


Figure 15—Process flow for signing data

¹³ The Protocol Conformance Implementation Statement (PICS) in Annex A allows an implementation to state the maximum certificate chain length that it supports.

5.3.3 Encrypt data

The security processing services are requested to encrypt data via Sec-EncryptedData.request. The information elements used by the security processing services to generate encrypted data are specified in 7.2.15.

The security processing services return the result of the request (either an octet string containing the encrypted data, or an error code) via Sec-EncryptedData.confirm. On success, the octet string returned shall be a valid encoding of a 1609Dot2Data as defined in Clause 6.

The security processing services shall reject a request to encrypt data to a certificate that is not valid according to the criteria specified in 5.5. WAVE security services may reject a request to encrypt data if the number of requested recipients is greater than 6.

A single input PDU may be encrypted for multiple recipients, resulting in a single encrypted PDU that may be decrypted by any of those recipients.

The encryption process is illustrated in Figure 16, where the dashed lines indicate entities and processing that are defined in this standard.

NOTE—Encrypted data created by the security processing services includes an integrity check value. Changes made to the encrypted data before sending it may invalidate this check value resulting in the recipient rejecting the encrypted data.

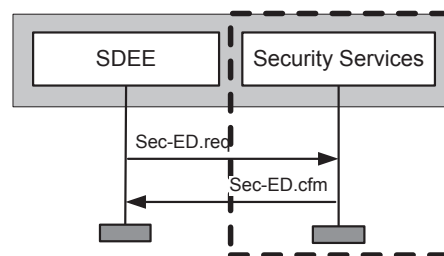


Figure 16—Process flow for encrypting data

5.3.4 Sign and encrypt data

The security processing services may be requested to create signed and encrypted data by successive invocations of Sec-SignedData.request and Sec-EncryptedData.request.

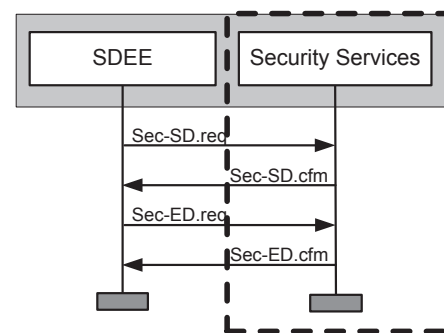


Figure 17 — Process flow for generating signed and encrypted data

5.3.5 Decrypt data

The security processing services are requested to decrypt data via `Sec-SecureDataContentExtraction.request`. The information elements used by the security processing services for this primitive are specified in 7.2.17. These information elements include a CMH in *Key and Certificate* state. The security services take as input encrypted data that is a valid encoding of a `1609Dot2Data` as defined in Clause 6.

The security processing services return the result of the request via `Sec-SecureDataContentExtraction.confirm`. If all operations succeed, the result shall be the type of the data, the plaintext contents of the data and additional information elements that were included by the sending security services. If any of the operations fail, the result shall be an appropriate error code.

The security processing services may reject a request to decrypt data if the number of requested recipients is greater than 6.

While performing operations in response to `Sec-SecureDataContentExtraction.request`, the security processing services shall also extract and return the information elements specified in the specification of `Sec-SecureDataContentExtraction.confirm`.

The decryption process is illustrated in Figure 18, where the dashed lines indicate entities and processing that are defined in this standard. See D.2 for a sample process flow for receiving secure data.

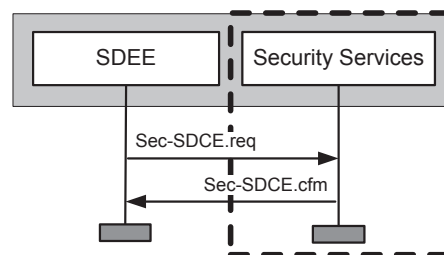


Figure 18—Process flow for decrypting data

5.3.6 Verify signed data

The security processing services are requested to verify signed data via `Sec-SignedDataVerification.request`. The information elements used by the security processing services for this primitive are specified in 7.2.19. The security services take as input signed data that is a valid encoding of a `SignedData` as defined in Clause 6.

The security processing services return the result of the request (either an indication of success with copies of relevant certificates or an indication of the reasons for failure) via `Sec-SignedDataVerification.confirm`.

The security processing services shall reject signed data as failing verification if the data and associated certificates fail the validity and consistency tests specified in 5.5. If requested, WAVE security services shall reject signed data as failing verification if the signed data fails any of the relevance tests specified in 5.5.5. The security processing services may reject signed data as failing verification if the full certificate chain from the certificate to a trust anchor, as described in 5.5.2, is longer than some threshold specified by the invoking SDEE.

While performing operations in response to `Sec-SignedDataVerification.request` the security processing services shall also extract information about the certificates associated with the message as described in the specification of `Sec-SignedDataVerification.request`. The security services return this information via `Sec-SignedDataVerification.confirm`.

The verification process is illustrated in Figure 19, where the dashed lines indicate entities and processing that are defined in this standard. Primitive names in the figure are abbreviated for compactness. See D.2 for a sample process flow for receiving secure data.

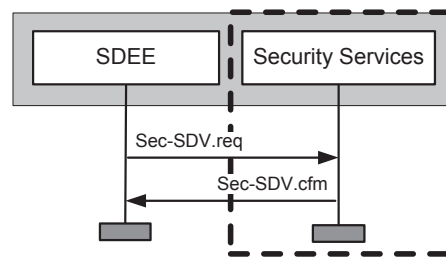


Figure 19—Process flow for verifying data

5.4 Signed WSAs

5.4.1 Sign WSA

The security processing services are requested to sign a WSA via WME-Sec-SignedWsa.request. The information elements that the security processing services require the WME to provide are specified in 7.3.2. The CMH in *Key and Certificate* state that is required to generate the signature shall be obtained from the PSSME via PSSME-Sec-CryptomaterialHandle.request. A full process flow covering the inclusion of a secure provider service in a signed WSA is illustrated in D.4.

The security processing services return the result of the request (either an octet string containing the signed WSA, or an error code) to the WME via in a WME-Sec-SignedWsa.confirm. On success, the octet string returned shall be a valid encoding of a 1609Dot2Data as defined in Clause 6. WME behavior following reception of WME-Sec-SignedWsa.confirm is specified in IEEE Std 1609.3.

WAVE security services shall reject a request to sign a WSA if it does not have access to a certificate that is currently valid according to the criteria specified in 5.5, if the CMH is not valid according to the criteria specified in 5.2.3.4, or if the certificate referenced by the CMH is not an explicit certificate (see 5.5.2). WAVE security services may reject a request to sign data if the full certificate chain from the certificate to a trust anchor, as described in 5.5.2, is longer than 5 certificates. WAVE security services may reject a request to sign a WSA if it contains more than 32 ServiceInfo entries (see IEEE Std 1609.3 for the definition of WSAs and ServiceInfo entries).

The WSA signing process is illustrated in Figure 20 where the dashed lines indicate entities and processing that are defined in this standard. Primitive names in the figure are abbreviated for compactness. Sample process flows to obtain and use certificates for WSA signing are described in D.3.2 and D.4.

NOTE—Any changes made to the signed WSA before sending it invalidate the digital signature, resulting in the recipient rejecting the signed WSA.

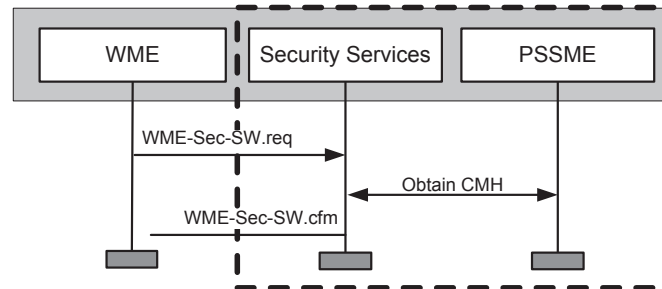


Figure 20—Process flow to sign WSA

5.4.2 Verify signed WSA

The security processing services are requested to verify a signed WSA via WME-Sec-SignedWsaVerification.request¹⁴. The information elements used by the security processing services for this primitive are specified in 7.3.4.

The security processing services shall then invoke WME-Sec-SignedWsaVerification.confirm to return the result to the SDEE. This result shall be either a failure code or a notification of success along with information extracted from the signed WSA.

As described in the specification of WME-Sec-SignedWsaVerification.confirm, if the WSA is valid the security processing services shall store any new certificate information obtained while processing the data with the CME, for example via CME-AddCertificate.request.

WAVE security services shall reject a signed WSA as failing verification if the data and associated tests fail the validity tests specified in 5.5. WAVE security services may reject a signed WSA if the full certificate chain from the certificate to a trust anchor, as described in 5.5.2, is longer than 5 certificates. WAVE security services may reject a signed WSA if it contains more than 32 ServiceInfo entries (see IEEE Std 1609.3 for the definition of WSAs and ServiceInfo entries). WAVE security services may reject a signed WSA if it is not signed by an explicit certificate (see 5.5.2).

The signed WSA verification process is illustrated in Figure 21, where the heavy dashed lines indicate entities and processing defined in this standard. Primitive names in the figure are abbreviated for compactness.

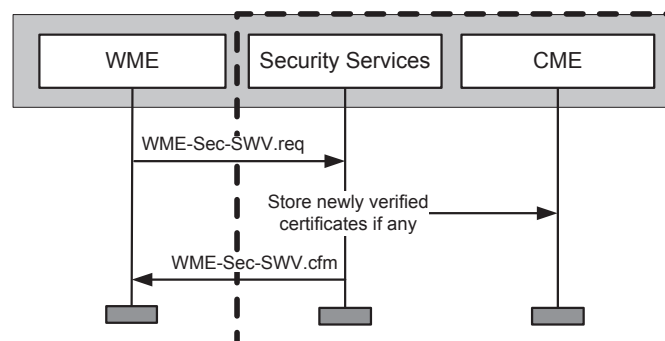


Figure 21—Process flow for validating a signed WSA

¹⁴ This is referred to in IEEE Std 1609.3-2010 as WaveSecurityServices-SignedWsaValidation.request, and the corresponding confirm primitive is referred to as WaveSecurityServices-SignedWsaValidation.confirm.

5.5 Validity of signed communications

5.5.1 General

A signed communication for a secure communications entity is valid if all of the following hold:

- The communication is correctly formed using the data structures of Clause 6.
- There is a certificate chain that leads from the signing certificate to a known trust anchor, such that:
- All of the certificates in the chain are correctly formed using the data structures of Clause 6.
- The certificate chain is internally consistent.
- None of the certificates in the chain have been revoked.
- All certificates in the chain are cryptographically verified with the appropriate public keys.
- (Optional) The number of certificates in the chain, from end-entity certificate to root certificate inclusive, is less than some entity-specific limit¹⁵.
- The communication is consistent with the signing certificate.
- The communication cryptographically verifies with the appropriate public keys.
- The communication meets other appropriate criteria (for example, to protect against data that are replays or are not relevant).

This clause specifies the criteria that determine whether a message is valid:

- Clause 5.5.2.1 specifies a validly constructed certificate chain.
- Clause 5.5.2.2 specifies cryptographic verification of certificates and communications.
- Clause 5.5.3 specifies consistency of data with a certificate or two certificates with each other.
- Clause 5.5.4 specifies expiry and revocation.
- Clause 5.5.5 specifies relevance checking.

Table 1 summarizes the steps to be used when validating a signed communication. The order of processing is fixed only where noted. If any of the steps fail, the signed communication shall be rejected. Clause 7 specifies in detail processing that may be carried out to check signed communications against these criteria.

¹⁵ Whether there is such a limit, and if so what value it takes, is out of scope for this standard. If there is such a limit, it would be appropriate for it to be specified as part of the specification of the signed communications entity.

Table 1—Steps for validating a signed communication

Step	Description	Flow specification	Processing specification	Notes
a)	Construct the certificate chain	5.5.2	Sec-SignedData-Verification.request or WME-Sec-SignedWsa-Verification.request, CME-Function-ConstructCertificateChain	Process is illustrated in D.6.2. If the end-entity certificate is already known and no revocation information has been received since it was last used to verify a message, this step and steps b), c) and f) can be skipped.
b)	Check that the certificates in the chain were not revoked	5.5.4	CME-CertificateInfo.request	Cannot be completed before a) is completed.
c)	Check the certificate chain for consistency	5.5.3	Sec-SignedData-Verification.request or WME-Sec-SignedWsa-Verification.request, Sec-Function-CheckCertificateChain-Consistency	Cannot be completed before a) is completed.
d)	Check that the message is consistent with the certificate	5.5.3	Sec-SignedData-Verification.request or WME-Sec-SignedWsa-Verification.request	If the signing certificate used inherited permissions, depends on a). Otherwise, no dependencies.
e)	Check that the message is internally consistent and consistent with transport layers	5.5.3.2.3, 5.5.3.2.4	Sec-SignedData-Verification.request or WME-Sec-SignedWsa-Verification.request	If the signing certificate used inherited permissions, depends on a). Otherwise, no dependencies.
f)	Cryptographically verify the signature on explicit certificates	5.5.2.2	Sec-SignedData-Verification.request or WME-Sec-SignedWsa-Verification.request, Sec-Function-VerifyChainAndSignature	Depends on a).
g)	Cryptographically verify the signature on the communication	5.5.2.2	Sec-SignedData-Verification.request or WME-Sec-SignedWsa-Verification.request, Sec-Function-VerifyChainAndSignature	Includes cryptographic verification of implicit certificates. Depends on a) if there were implicit certificates.
h)	Check relevance, check for replay	5.5.5	Sec-SignedData-Verification.request, CME-Sec-ReplayDetection.request	Optional; only for signed data exchange.
i)	Check that the WSA is fresh	5.4.2	PSSME-OutOfOrder-Detection.request	Only for signed WSAs.

5.5.2 Certificate chains

5.5.2.1 General

Received signed communications are verified using the *end-entity certificate* associated with the signed data (an end-entity certificate is a certificate used for any purpose other than issuing certificates).

A certificate contains, explicitly or otherwise:

- At least one public key for a public key cryptosystem, belonging to the certificate holder.
- A list of the permissions associated with that public key.
- An identifier for the issuer (who may be the same as the certificate holder or may be different).
- A cryptographic demonstration that the issuer authorized the linkage between the public key and the permissions.

The entity that uses the private key corresponding to the public key is referred to as the *certificate holder*. A certificate used to validate application or management data is an *end-entity certificate*. A certificate used to validate another certificate is a *CA certificate*. If the public key is explicitly given in the certificate, the certificate is an *explicit certificate*. If the public key is not explicitly given in the certificate, but is obtained by performing additional processing, the certificate is an *implicit certificate*. The difference between implicit and explicit certificates is illustrated in Figure 22.

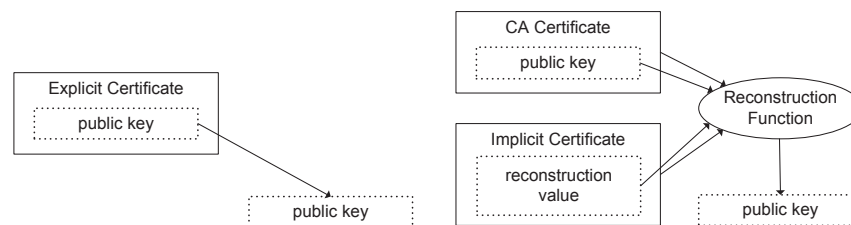


Figure 22—Implicit and explicit certificates

A necessary (but not sufficient) condition for a receiver to trust a received certificate is that the receiver can construct a *certificate chain* from the certificate to a *trust anchor*. A certificate chain is an ordered set of certificates such that each certificate in the chain is the *issuing certificate* for the subsequent one. One certificate is the issuing certificate for another 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 creating cryptomaterial to be used in recovering the public key (in the case of an implicit certificate). To enable construction of certificate chains, IEEE 1609.2 certificates contain an identifier for the signer, referred to as the *signer_id* (see 6.3.3). An algorithm for the construction of certificate chains is specified in 7.8.2 and illustrated in D.6.2.

The counterpart of an issuing certificate is a *subordinate certificate*; each certificate in a chain, other than the end-entity certificate and the trust anchor, is the issuer of the certificate after it and the subordinate of the certificate before it.¹⁶

A trust anchor is any certificate that is known to be trustworthy by itself; in other words, not by reference to any other certificate. The trust anchor may be a *root certificate*, which is a certificate that is verified with

¹⁶ Technically, the certificate holder is the issuer of the certificate after it in the chain, rather than the certificate itself being the issuer. For conciseness, this standard refers to the certificate as the issuer.

the public key included directly in the certificate. Alternatively, the trust anchor may be any other CA certificate that is known to be trustworthy.

The data structures given in 6.2 allow the sender to indicate which certificate is to be used to verify the signed communication. The sender may include a hash of the signing certificate, the signing certificate itself, or a certificate chain including the end-entity certificate and a series of issuing certificates up to, but not including, the root.

In constructing the certificate chain, the receiver may use the certificates that were sent by the sender or locally cached copies of certificates. Local copies of certificates are managed by the CME. An algorithm for constructing a certificate chain using certificates from a signed communication and from local storage is given in 7.8.2 and illustrated in Figure D.19.

A receiver shall reject a communication signed by a certificate unless the chain from the certificate to its trust anchor meets the following conditions:

- The trust anchor is an explicit certificate.
- All explicit certificates are issued by explicit certificates.¹⁷

A specification of a secure communications entity may use the IEEE 1609.2 security profile (introduced in 4.3.2 and specified fully in Annex B) to specify a maximum length for the full certificate chain. Certificate chains greater than the specified maximum length are not considered valid in the context of that secure communications entity. The length of a full certificate chain is the number of certificates in the chain including the root certificate. Therefore, the minimum length of a full certificate chain is 2: the end-entity certificate and the root certificate.

The Protocol Implementation Conformance Statement (PICS) proforma given in Annex A allows the vendor of a general-purpose implementation of WAVE Security Services to state the maximum length of certificate chain that the implementation supports.

NOTE 1—An implementation only needs to construct a chain to a trust anchor, not to a root, to determine that it is valid; however, this does not allow an unambiguous definition of certificate chain length, as a particular non-root certificate may be considered a trust anchor by one instance of WAVE Security Services and not by another. The only unambiguous definition of length, therefore, is a length that includes the root.

NOTE 2—Secure communications entities may have a maximum certificate chain length, but may also give guidance to developers that an appropriate certificate chain length is less than this maximum. For example, since long certificate chains increase packet size and therefore channel congestion and error rates, it is appropriate for the specification of the secure communications entity to give guidance that short (relative to the maximum) certificate chains should be used. This is particularly important for secure communications entities that transmit frequently.

5.5.2.2 Cryptographic verification of certificate chains

The CME shall only accept an explicit certificate as a trust anchor.

Each subordinate explicit certificate in a chain is verified by ensuring that the public key in the issuing certificate can be used to verify the signature on the subordinate certificate. See Figure 23 for an illustration.

If the end-entity certificate is an explicit certificate, the signature on the communication is verified by ensuring that the public key in the end-entity certificate can be used to verify the signature on the communication. See Figure 23 for an illustration.

¹⁷ This implies that if the certificate chain contains both implicit certificates and explicit certificates, all of the implicit certificates are at the end of the chain and all of the explicit certificates are at the start of the chain.

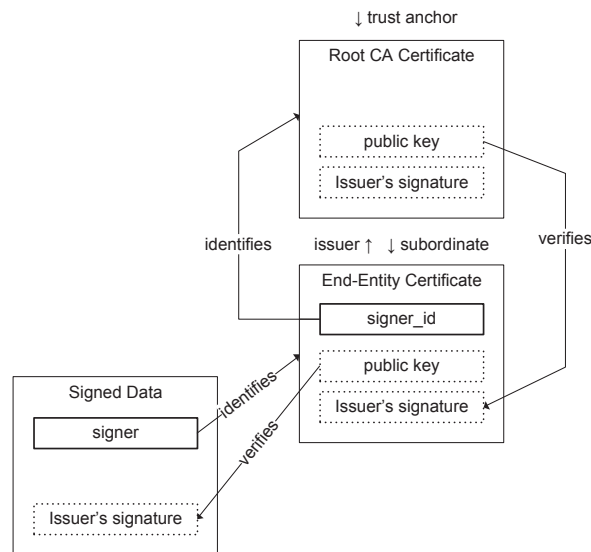


Figure 23—Cryptographic verification with explicit certificates

If the end-entity certificate is an implicit certificate, the signature on the communication is verified by ensuring that the public key that is obtained from the end-entity certificate using the methods of 5.8.6 can be used to verify the signature on the communication. Figure 24 illustrates that the verification process involves the end-entity's implicit certificate and the end-entity's issuing certificate's public key.

If any CA certificates are implicit certificates, they are (implicitly) verified by verifying the signature on the communication.

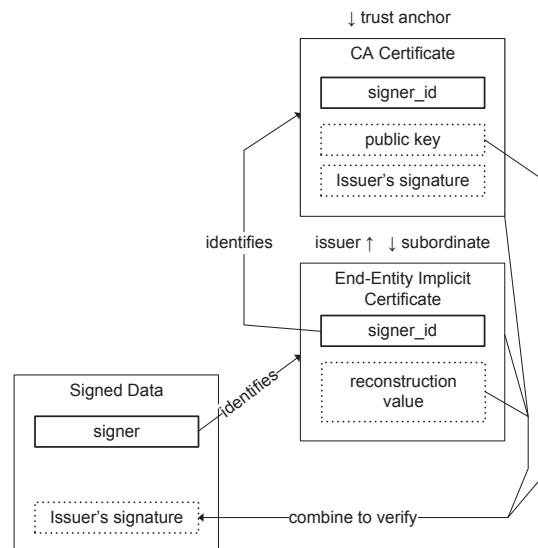


Figure 24—Length-2 certificate chain with implicit certificates and a non-root CA as trust anchor

For further illustrations, see D.6.1.

5.5.3 Permissions and consistency of permissions

5.5.3.1 General

A certificate communicates the permissions of its holder, namely:

- Geographic permissions: the region within which the certificate is valid, if relevant.
- Validity period: the time period within which the certificate is valid.
- Operational permissions: the activities that the holder is allowed to perform.
- For a CA certificate, the type(s) of certificate that it is permitted to issue.

Operational permissions are expressed using the following variables:

- The PSID data type defined in IEEE Std 1609.3.
- The SSPs. These permissions are defined outside this standard but subject to the constraints of 6.3.25.
- For WSA certificates, the maximum priority at which the WME shall permit a service to be advertised.

For a CA, the operational permissions contain PSIDs only, not SSPs (see 6.3.8, 6.3.20, 6.3.21). For a CA certificate, the operational permissions in the CA certificate specify the operational permissions that the CA is permitted to include in subordinate certificates issued by that CA certificate. For an enrolment certificate (see 5.6.1), the operational permissions are the list of permissions that the certificate holder is permitted to request be included in its operational certificates.

NOTE— For an end-entity certificate holder, the permitted activities corresponding to a given PSID and SSP are defined by the organization that reserves the PSID. IEEE Std 1609.12 specifies a list of reserved PSIDs and the organizations that have reserved them.

5.5.3.2 Consistency between signed communications and signing certificates

5.5.3.2.1 General

A signed communication (signed data or WSA) is consistent with the signing certificate if:

- The communication originated from within the geographic region indicated in the certificate.
- The communication was created within the validity period of the certificate.
- The communication's expiry time is within the validity period of the certificate.
- The information within the communication is consistent with the operational permissions given in the certificate where consistency is determined using rules specified by the organization that reserved the PSID.
- The public key indicated by the certificate can be used to cryptographically verify the signature on the communication.

A signed communication that is inconsistent with its signing certificate is invalid.

The following clauses define consistency of permissions with signed data (5.5.3.2.2) and with WSAs (5.5.3.2.5).

5.5.3.2.2 Signed data

Signed data is encoded using the data structures of Clause 6. The encoding contains a PSID. Signed data is consistent with the permissions in the signing certificate if the PSID information element in the signed data is one of the PSIDs in the operational permissions of the certificate.

A certificate also contains a SSP field for each PSID. The security processing services shall make the SSP available to an SDEE that verifies signed data.

NOTE—The SSP enables the SDEE to perform additional validity checks on the data payload. These checks are out of scope for this standard. See E.5.1 and E.5.2 for more discussion.

Figure 25 illustrates the process of checking that signed data is consistent with the associated signing certificate, and additionally captures the process of checking that the signed data is internally consistent as defined in Clause 6. The boxes within the data and certificate identify the information elements that are conveyed by those data structures. For clarity, only the relevant fields within the data structures are shown. The generation time, expiry time, and generation location may be included within the fields added by the security processing or may be a part of the payload. See Clause 6 for a specification of the exact fields in the certificate and the signed data that encode this information. See Clause 7 for specification of processing that correctly performs these checks.

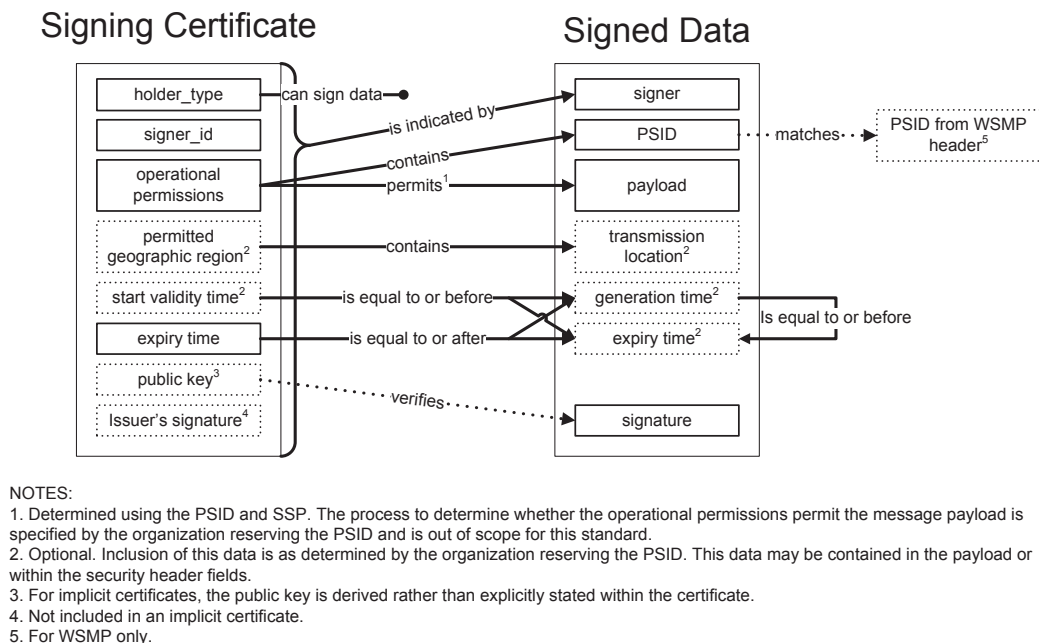


Figure 25—Checking consistency of permissions between signed data and its signing certificate, and within signed data

5.5.3.2.3 Consistency between signed data and transport layers

If the signed data was received within the WAVE Short Message Protocol (WSMP) described in IEEE Std 1609.3, it shall be considered invalid unless the PSID in the signed data structure is the same as the WSMP PSID header field. This is also illustrated in Figure 25. The Sec-SignedDataVerification.request primitive specified in Clause 7 supports carrying out this check within the security services.

5.5.3.2.4 Consistency within signed data

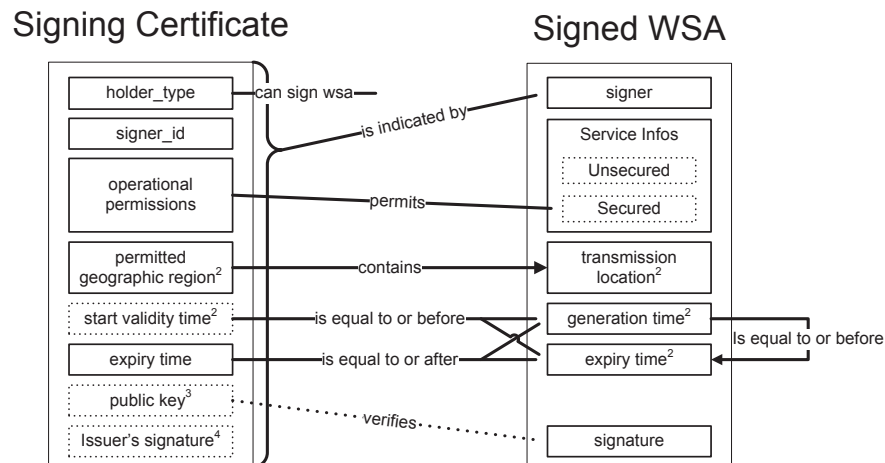
If the signed data contains both a generation time and an expiry time, it shall be considered invalid if the expiry time is after the generation time. This is also illustrated in Figure 25.

5.5.3.2.5 Signed WSA

The process of checking consistency of operational permissions within a signed WSA is specified in full in 7.3.4 and illustrated in D.4.

A signed WSA may contain both secured and unsecured ServiceInfo fields. The security services determine the validity of the secure provider services only.

Figure 26 illustrates the process overall of checking that a signed WSA is consistent with the associated signing certificate, and additionally captures the process of checking that a signed WSA is internally consistent as defined in Clause 6. The boxes within the data and certificate identify the information elements that are conveyed by those data structures. For clarity, only the relevant fields within the data structures are shown. See Clause 6 for a specification of the exact fields in the certificate and the signed WSA that encode this information. See Clause 7 for specification of the exact processing.



NOTES:

1. For implicit certificates, the public key is derived rather than explicitly stated within the certificate.
2. Not included in an implicit certificate.

Figure 26—Checking consistency of permissions between a signed WSA and its signing certificate, and within the signed WSA

A signed WSA is consistent with the operational permissions in its signing certificate if each of the ServiceInfo entries in the WSA is consistent with the corresponding permissions entry in the signing certificate. The ServiceInfo is defined in IEEE Std 1609.3. See later in this clause for the specification of how to determine which permissions entry in the certificate corresponds to which ServiceInfo, and how to determine which entries in the WSA are secured or unsecured.

A ServiceInfo entry consists of information elements including:

- A PSID.
- The priority at which the service is offered.

A permissions entry in a WSA certificate consists of the following information elements:

- A PSID.
- A maximum priority for the service.
- Optionally, SSPs, which are interpreted by the receiving user service rather than by the security services.

A ServiceInfo entry is consistent with a permissions entry if:

- The PSID entries are identical.
- The priority in the ServiceInfo is less than or equal to the maximum priority in the permissions.

Figure 27 illustrates the process of checking that a WSA certificate permissions entry is consistent with a particular ServiceInfo.

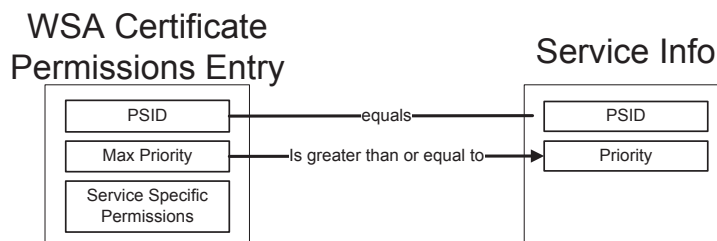


Figure 27—Checking consistency between a ServiceInfo in a signed WSA and the permissions in its signing certificate

The permissions in a WSA signing certificate are mapped to the ServiceInfos in the WSA by means of the `permission_indices` field in the signed WSA. See Clause 6 for a specification of the exact fields in the certificate and the signed data that encode this information. The `permission_indices` field is an array of integers the same length as the array of ServiceInfos in the WSA. For each ServiceInfo, the corresponding entry in the `permission_indices` field is either 0, indicating that the ServiceInfo describes an unsecured service, or a non-zero integer, indicating the relevant field in the permissions in the certificate.

Figure 28 illustrates the process of mapping from the ServiceInfos in the WSA to the permissions in the certificate. See Clause 7 for specification of the exact processing. An example is given in D.3.2. The use of the SSP by the user services is discussed in E.6.3.

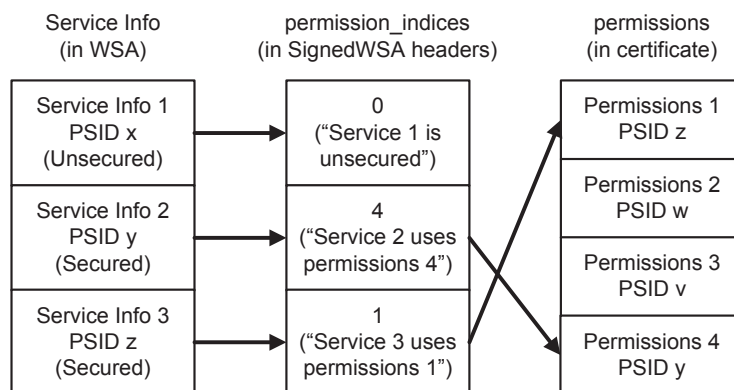


Figure 28—Mapping from the list of ServiceInfos in a WSA to the list of permissions in a certificate via permission_indices

5.5.3.3 Consistency between subordinate certificates and issuing certificates

A subordinate certificate is consistent with its issuing certificate if:

- The subordinate certificate's validity region is wholly contained in the issuing certificate's validity region.
- The subordinate certificate's validity period is within the issuing certificate's validity period.
- The subordinate certificate's operational permissions are a subset of the issuing certificate's operational permissions.
- The issuing certificate signed the subordinate certificate if it is an explicit certificate or the cryptographic material in the certificates can be used to verify a signature if it is an implicit certificate.
- The issuing certificate is allowed to issue a certificate of the type of the subordinate certificate, as determined by the `permitted_holder_types` field in the certificate (see 6.3.8, 6.3.20, 6.3.21).

A signed communication is invalid if any subordinate certificate in the chain is inconsistent with its issuing certificate.

If any subordinate certificate is inconsistent with its issuing certificate, the subordinate certificate in the pair, and any certificates that chain back to it, are invalid.

If any subordinate certificate is inconsistent with its issuing certificate but has a signature that verifies with the issuing certificate's public key, the security services shall treat the issuing certificate and all certificates issued by it as invalid.

Figure 29 illustrates the process of checking that a subordinate certificate is consistent with its issuing certificate, and additionally captures the process of checking that a certificate is internally consistent as defined in Clause 6. For clarity, only the relevant fields within the data structures are shown. See Clause 6 for a specification of the exact fields in the certificate and the signed data that encode this information. See Clause 7 for a specification of the exact processing.