

IEEE Std 1363a™-2004
(Amendment to
IEEE Std 1363™-2000)

IEEE Standards

1363a™

**IEEE Standard Specifications for
Public-Key Cryptography—
Amendment 1: Additional Techniques**

IEEE Computer Society

Sponsored by the
Microprocessor and Microcomputer Standards Committee



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

2 September 2004

Print: SH95223

PDF: SH95223



Documents

Document Sharing Hub

This is a copyrighted IEEE Standard. For personal or standards development use only.

Full document details: IEEE 1363a-2004

Recognized as an
American National Standard (ANSI)

IEEE Std 1363a™-2004
(Amendment to
IEEE Std 1363™-2000)

IEEE Standard Specifications for Public-Key Cryptography— Amendment 1: Additional Techniques

Sponsor

Microprocessor and Microcomputer Standards Committee
of the
IEEE Computer Society

Approved 22 July 2004

American National Standard Institute

Approved 25 March 2004

IEEE-SA Standards Board

Abstract: This standard specifies additional public-key cryptographic techniques beyond those in IEEE Std 1363-2000. It is intended to be merged with IEEE Std 1363-2000 during future revisions.

Keywords: digital signature, encryption, key agreement, public-key cryptography

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

Copyright © 2004 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 2 September 2004. 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.

Print: ISBN 0-7381-4003-1 SH95223
PDF: ISBN 0-7381-4004-X SS95223

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.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The 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 the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The 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 this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, 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 herein 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 five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, 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 this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received 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 interpretation requests except in those cases where the matter has previously received formal consideration. 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, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331USA

NOTE—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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

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.

Introduction

(This introduction is not part of IEEE Std 1363a-2004, IEEE Standard for Specifications for Public-Key Cryptography—Amendment 1: Additional Techniques.)

IEEE Std 1363a-2004 started in 1996 as an amendment to IEEE P1363, which later became IEEE Std 1363™-2000. At the time of the project's inception, the techniques included in P1363 were fairly stable; however, there were additional techniques that had been submitted to the Working Group that warranted further study and consideration. The P1363a project became the entry point for newly proposed and comparatively less-established techniques, while the P1363 document was solidified and prepared for the IEEE balloting process.

In early 1999, the Working Group began to limit the contents of P1363a primarily to include techniques that fit within the framework of the base document. In particular, the group concentrated on techniques in each family that performed functions that were not available in the P1363 document, such as encryption schemes in the discrete logarithm (DL) and elliptic curve (EC) families and signature schemes with message recovery in all families [DL, EC, and IF (integer factorization)].

Because of the large number of good candidates and the close similarities between many of the different techniques, the Working Group began to formalize the criteria for acceptance into P1363a, comparing all aspects of the techniques. By late 1999, the Working Group had converged on several candidate techniques to include in P1363a, making selections to provide a variety of relevant tradeoffs among security attributes, computational efficiency, and industry adoption for each family. As with the P1363 project, the Working Group chose to close P1363a to additional techniques and prepare it for ballot, leaving standardization of additional techniques that may not have been ready for inclusion at that time to future standards.

Throughout the process of creating IEEE Std 1363a-2004, the Working Group has been very fortunate to receive numerous submissions of creative, useful, and well-designed public-key cryptographic techniques. In addition to techniques in the IF, DL, and EC families that were not included in 1363a, there were also a number of techniques submitted that fell outside of the scope of both IEEE Std 1363-2000 and IEEE Std 1363a-2004. As a result, the Microprocessor Standards Committee (MSC) commissioned a study group, in which many of the 1363 working group members participated, to explore the possibility of creating additional standards related to public-key cryptography. In late 2000, the working group began work on P1363.1 (Public-Key Cryptographic Techniques Based on Hard Problems over Lattices) and P1363.2 (Password-Based Public-Key Cryptographic Techniques). At the time of this writing, the Working Group is also attempting to put together a project for the second amendment to Std 1363-2000 (P1363b), which would continue the work from IEEE Std 1363-2000 and IEEE Std 1363a-2004. The Working Group is considering the creation of a registry for public-key cryptographic techniques.

The process of developing IEEE Std 1363a-2004 has been very challenging, but it has allowed the Working Group to continue to refine the general understanding of public-key cryptography and to follow up on the work put into IEEE Std 1363-2000. The IEEE P1363 Working Group continues to be an excellent forum for experts to discuss technical and standardization issues associated with public-key cryptography. It has provided a focal point for the presentation of new developments in public-key cryptography and remains a source for up-to-date information on the topic. For the duration of its existence, the Working Group intends to maintain a web page that will support all of the 1363 standards and current projects (see <http://grouper.ieee.org/groups/1363/index.html>).

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/updates/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

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 with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates and nondiscriminatory, reasonable terms and conditions to applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates, terms, and conditions of the license agreements offered by patent holders or patent applicants. Further information may be obtained from the IEEE Standards Department.

Organization

This standard is written as an amendment to the IEEE Std 1363-2000 document and is intended to be merged into a future version of that document. Familiarity with IEEE Std 1363-2000 is assumed. The following clauses and annexes are updated:

- Clause 2, References
- Clause 3, Definitions
- Clause 4, Types of cryptographic techniques
- Clause 5, Mathematical conventions
- Clause 6, Primitives based on the discrete logarithm problem
- Clause 7, Primitives based on the elliptic curve discrete logarithm problem
- Clause 8, Primitives based on the integer factorization problem
- Clause 9, Key agreement schemes
- Clause 10, Signature schemes
- Clause 11, Encryption schemes
- Clause 12, Message-encoding methods
- Clause 13, Key derivation functions
- Clause 14, Auxiliary techniques (new title)
- Annex A (informative) Number-theoretic algorithms
- Annex B (normative) Conformance
- Annex C (informative) Rationale

- Annex D (informative) Security considerations
- Annex E (informative) Formats
- Annex G (informative) Bibliography (new number; was Annex F)

One new annex is inserted:

- Annex F (informative) Information about patents

Participants

The following is a list of participants in the IEEE P1363 Working Group.

William Whyte , <i>Chair</i>	Don Johnson , <i>Vice-Chair</i>	
Ari Singer , <i>Secretary</i>	David Jablon , <i>Treasurer</i>	
Mike Brenner , <i>Primary Editor</i>	Burt Kaliski , <i>P1363a Technical Editor</i>	
Beni Arazi	Pieter Kasselmann	Satomi Okazaki
Daniel V. Bailey	Tetsutaro Kobayashi	Anand Rajan
Daniel Brown	David Kravitz	Allen Roginsky
Lily Chen	Pil Joong Lee	Roger Schlafly
Wei Dai	Daniel Lieman	Rich Schroepfel
Louis Finkelstein	Michael Markowitz	Jerry Solinas
Gurgen Kachatryan	Tatsuaki Okamoto	David Stern

In addition, the Working Group would like to thank the following people for their contributions to this amendment, whether by submitting techniques for consideration, participating in mailing list discussions, or attending meetings.

Michel Abdalla	David Hopwood	Hong Nguyen
Tolga Acar	Fumitaka Hoshino	Sang Ho Oh
Carlisle Adams	Jim Hughes	Hilarie Orman
Kochira Akiyama	Jakob Jonsson	Christof Paar
Kazumato Aoki	KCDSA Task Force Team	China Pellacur
Terry Arnold	Chang Han Kim	Mohammad Peyravian
Paulo Barreto	Hee Jin Kim	David Pointcheval
Mihir Bellare	Jeong-Soo Kim	Martin Rehwald
Simon Blake-Wilson	Kunio Kobayashi	Holger Reif
Daniel Bleichenbacher	Kristin Lauter	Leo Reyzin
Jurjen Bos	Peter Lawrence	Phillip Rogaway
Dong Hyun Cheon	Byoungcheon Lee	Rei Safavi-Naini
Bram Cohen	Chang-Hyi Lee	Erkay Savas
Nicolas Courtois	Sung Jae Lee	Dominikus Scherkl
Carlin Covey	Franck Leprevost	Martin Schulze
Scott Crenshaw	Jon In Lim	Thomas Schweinberger
Lucien Dancanet	Helger Lipmaa	Michael Scott
Lei Fang	Moses Liskov	Victor Shoup
Ludovic Flament	Stefan Löwe	Joe Silverman
Eiichiro Fujisaki	Phil MacKenzie	Nigel Smart
Walter Fumy	John Malone-Lee	David Sowinski
Ernst Giessmann	James Manger	Kazuo Takaragi
Tom Gindin	Marcel Martin	Scott Vanstone
Louis Goubin	Stephen M. Matyas	Serge Vaudenay
Peter Gutmann	Preda Mihailescu	Tom Wu
Safuat Hamdy	Bodo Möller	Ok Yeon Yi
Florian Hess	Jean Monnerat	Yiqun Lisa Yin
Shouichi Hirose	Shiho Moriai	Joong Chul Yoon
Robert Hofer	Hikaru Morita	Susumu Yoshida
Erik Falk Højsted	James Muir	Yuliang Zheng
Seak Hie Hong	Anthony Mulcahy	

The Working Group apologizes for any inadvertent omissions from the previous list. Please note that inclusion of a person's name on the previous two lists on page v does not imply that the person agrees with all of the materials in this amendment.

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

Terry Arnold	Jack Johnson	Roger Schlafly
Steven Bard	Burt Kaliski	Ari Singer
Mitchell Bonnett	David Kravitz	Joseph Tardo
Guru Dutt Dhingra	Thomas M. Kurihara	Jerry Thrasher
Dante Del Corso	Gregory Luri	Joan Viaplana
Sourav Dutta	Robert Mortonson	Paul Work
Neil Horman	David M. Rockwell	Don Wright
Don Johnson	Tom Schaal	Cheng-Wen Wu
		Oren Yuen

When the IEEE-SA Standards Board approved this standard on 25 March 2004, it had the following membership:

Don Wright, Chair

Steve M. Mills, Vice Chair

Judith Gorman, Secretary

Chuck Adams	Mark S. Halpin	Daleep C. Mohla
H. Stephen Berger	Raymond Hapeman	Paul Nikolich
Mark D. Bowman	Richard J. Holleman	T. W. Olsen
Joseph A. Bruder	Richard H. Hulett	Ronald C. Petersen
Bob Davis	Lowell G. Johnson	Gary S. Robinson
Roberto de Boisson	Joseph L. Koepfinger*	Frank Stone
Julian Forster*	Hermann Koch	Malcolm V. Thaden
Arnold M. Greenspan	Thomas J. McGean	Doug Topping
		Joe D. Watson

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan Cookson, *NIST Representative*

Don Messina
IEEE Standards Project Editor

Contents

1.	Overview.....	2
1.1	Scope.....	2
1.2	Purpose.....	2
2.	References.....	2
3.	Definitions	3
4.	Types of cryptographic techniques	4
4.2	Primitives	4
4.3	Schemes	4
4.4	Additional methods.....	4
4.5	Table summary	5
5.	Mathematical conventions	7
5.1	Mathematical notation	7
5.4	Elliptic curves and points.....	9
5.5	Data type conversion	10
6.	Primitives based on the discrete logarithm problem.....	15
6.2	Primitives	15
7.	Primitives based on the elliptic curve discrete logarithm problem.....	21
7.2	Primitives	21
8.	Primitives based on the integer factorization problem	26
8.2	Primitives	28
9.	Key agreement schemes.....	33
9.2	DL/ECKAS-DH1.....	33
9.3	DL/ECKAS-DH2.....	34
9.4	DL/ECKAS-MQV	35
10.	Signature schemes.....	36
10.1	General model.....	36
10.2	DL/ECSSA.....	38
10.3	IFSSA.....	39
10.4	DL/ECSSR.....	41
10.5	DL/ECSSR-PV	43
10.6	IFSSR.....	46

11.	Encryption schemes	48
11.1	General model	48
11.2	IFES	50
11.3	DL/ECIES	50
11.4	IFES-EPOC	55
12.	Message-encoding methods	57
12.1	Message-encoding methods for signatures with appendix	57
12.2	Message-encoding methods for encryption	64
12.3	Message-encoding methods for signatures giving message recovery	71
13.	Key derivation functions	77
13.2	KDF2	78
14.	Auxiliary techniques	79
14.1	Hash functions	79
14.2	Mask generation functions	82
14.3	Symmetric encryption schemes	83
14.4	Message authentication codes	87
	Annex A (informative) Number-theoretic algorithms	89
	Annex B (normative) Conformance	111
	Annex C (informative) Rationale	114
	Annex D (informative) Security considerations	119
	Annex E (informative) Formats	147
	Annex F (informative) Informative about patents	151
	Annex G (informative) Bibliography	152

IEEE Standard Specifications for Public-Key Cryptography— Amendment 1: Additional Techniques

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

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make small corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed either by using ~~strike through~~ (to remove old material) or underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instructions. ***Replace*** is used to make large changes in existing text, subclauses, tables, or figures by removing existing material and replacing it with new material. Editorial notes will not be carried over into future editions.

1. Overview

Replace subclauses 1.1 and 1.2 with the following text:

1.1 Scope

Specifications of common public-key cryptographic techniques supplemental to those considered in IEEE Std 1363™-2000, including mathematical primitives for secret value (key) derivation, public-key encryption, digital signatures, and identification, and cryptographic schemes based on those primitives, are provided. Specifications of related cryptographic parameters, public keys and private keys, are also provided. Class of computer and communications systems is not restricted.

1.2 Purpose

The transition from paper to electronic media brings with it the need for electronic privacy and authenticity. Public-key cryptography offers fundamental technology addressing this need. Many alternative public-key techniques have been proposed, each with its own benefits. However, there has been no single, comprehensive reference defining a full range of common public-key techniques covering key agreement, public-key encryption, digital signatures, and identification from several families, such as discrete logarithms, integer factorization, and elliptic curves.

It is not the purpose of this project to mandate any particular set of public-key techniques, or particular attributes of public-key techniques such as key sizes. Rather, the purpose is to provide a reference for specifications of a variety of techniques from which applications may select.

When the IEEE 1363a project began, work was in progress in IEEE Std 1363-2000 on specifying many of the basic techniques in a standard way. Additional techniques remained to be specified, which could eventually be added to IEEE Std 1363-2000. However, many of those additional techniques required further development before being standardized, whereas the basic techniques in IEEE Std 1363-2000 were relatively more established. To facilitate the completion of the work on the basic techniques while also providing a forum for discussing additional techniques, the IEEE 1363 Working Group sought to have separate projects for the two efforts, which would result in separate, companion standards for some period of time. It is the working group's intention that the standards will be merged during future revisions.

Replace Clause 2 with the following text:

2. References

This standard shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision shall apply.

ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation.¹

ANSI X9.71-2000, Keyed Hash Message Authentication Code (MAC).

FIPS PUB 46-3, Data Encryption Standard (DES), Federal Information Processing Standards Publication 46-3, U.S. Department of Commerce/National Institute of Standards and Technology, October 1999.²

¹ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

FIPS PUB 81, DES Modes of Operation, Federal Information Processing Standards Publication 81, U.S. Department of Commerce/National Institute of Standards and Technology, December 1980.

FIPS PUB 180-2, Secure Hash Standard, Federal Information Processing Standards Publication 180-2, U.S. Department of Commerce/National Institute of Standards and Technology, August 1, 2002.

FIPS PUB 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, U.S. Department of Commerce/National Institute of Standards and Technology, November 2001.

FIPS PUB 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198, U.S. Department of Commerce/National Institute of Standards and Technology, March 2002.

ISO/IEC 10118-3:1998, Information Technology-Security Techniques—Hash-Functions—Part 3: Dedicated hash-functions.³

NOTES

1—The above references are required for implementing some, but not all, of the techniques in this standard.

2—The mention of any standard in this document is for reference only, and it does not imply conformance with that standard. Readers should refer to the relevant standard for full information on conformance with that standard.

3—A bibliography is provided in Annex G.⁴

3. Definitions

Replace 3.37 with the following text:

3.37 private key: The private element of the public/private key pair. (Note that for convenience of implementation, some components of the public key may be included in the private key as well.) *See also:* **public/private key pair; valid key.**

Delete 3.50, insert the following subclause after 3.48, and renumber the existing 3.49 as 3.50:

3.49 signature scheme giving message recovery: A digital signature scheme where the signature contains enough information for recovery of part or all of the message that is signed. *Contrast:* **signature scheme with appendix.** *See also:* **digital signature; digital signature scheme.**

Insert the following definitions in alphabetical order and renumber as required:

binary field: A finite field containing 2^m elements for some integer $m > 0$. Also known as characteristic two finite field.

²FIPS publications are available from the National Technical Information Service (NTIS), U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161 (<http://www.ntis.gov/>) and/or from the Federal Information Processing Standards Publications web site at <http://www.itl.nist.gov/fipspubs/>.

³ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

⁴Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

characteristic: The prime integer p for the finite field $GF(p^m)$.

odd-characteristic extension field: A finite field containing p^m elements for some integer $m > 1$ where p is an odd prime number.

NOTE—Although mathematically, the various techniques based on odd-characteristic extension fields may generally be employed when $m = 1$, for the purposes of this standard, it will be assumed that $m > 1$ when the term “odd-characteristic extension field” is given.

prime field: A finite field of p elements, where p is an odd prime number. Also known as prime finite field.

4. Types of cryptographic techniques

4.2 Primitives

Replace the list of types of primitives with the following text:

- Secret Value Derivation Primitives (SVDP): components of key agreement schemes
- Pre-Signature Primitives (PSP), Signature Primitives (SP), and Verification Primitives (VP): components of signature schemes
- Encryption Primitives (EP) and Decryption Primitives (DP): components of encryption schemes

4.3 Schemes

Replace the list of types of schemes with the following text:

- Key agreement schemes (KAS), in which two parties use their public/private key pairs, and possibly other information to agree on a shared secret key. The shared secret key may then be used for symmetric cryptography. (See Clause 3 for the definition of symmetric cryptography.)
- Signature schemes with appendix (SSA), in which one party signs a message using its private key, and any other party can verify the signature by examining the message, the signature, and the signer’s corresponding public key.
- Signature schemes giving (message) recovery (SSR), in which one party signs a message using its private key such that part or all of the message can be recovered from the signature, and any other party can verify the signature, and recover that part of the message by examining the nonrecoverable part of the message (if any), the signature and the signer’s corresponding public key.
- Encryption schemes (ES), in which any party can encrypt a message using a recipient’s public key, and only the recipient can decrypt the message by using its corresponding private key. Encryption schemes may be used for establishing secret keys to be used in symmetric cryptography.

Replace 4.4 with the following text:

4.4 Additional methods

This standard specifies the following additional methods:

- Message-encoding methods, which are components of signature or encryption schemes:
 - 1) Encoding methods for signatures with appendix (EMSA)
 - 2) Encoding methods for signatures giving message recovery (EMSR)
 - 3) Encoding methods for encryption (EME)

- Key derivation functions (KDF), which are components of key agreement schemes
- Auxiliary techniques, which are building blocks for other additional methods
 - 1) Mask generation functions (MGF), which are used in EME
 - 2) Hash functions, which are used in EMSA, EMSR, EME, KDF, and MGF
 - 3) Symmetric encryption schemes, which are used in some encryption schemes, EMSR and EME
 - 4) Message authentication codes (MAC), which are a component of some encryption schemes

The specified additional methods are strongly recommended for use in the schemes. The use of an inadequate message-encoding method, key derivation function, auxiliary function, or message authentication code may compromise the security of the scheme in which it is used. Therefore, any implementation that chooses not to follow the recommended additional methods for a particular scheme should perform its own thorough security analysis of the resulting scheme.

Replace 4.5 with the following text:

4.5 Table summary

Table 1 gives a summary of all schemes in this standard, together with the primitives and additional methods that are invoked within a scheme.

Table 1—Summary of techniques in the standard

Scheme name	Primitives	Additional methods
<i>Key agreement schemes</i>		
DL/ECKAS-DH1	DL/ECSVDP-DH or -DHC	KDF1 or KDF2 (each uses a hash function)
DL/ECKAS-DH2	DL/ECSVDP-DH and/or -DHC	KDF1 or KDF2 (each uses a hash function)
DL/ECKAS-MQV	DL/ECSVDP-MQV or -MQVC	KDF1 or KDF2 (each uses a hash function)
<i>Signature schemes with appendix</i>		
DL/ECSSA	DLSP-NR and DLVP-NR OR DLSP-DSA and DLVP-DSA OR ECSP-NR and ECVP-NR OR ECSP-DSA and ECVP-DSA	EMSA1 (uses a hash function)
IFSSA	IFSP-RSA1 and IFVP-RSA1 OR IFSP-RSA2 and IFVP-RSA2 OR IFSP-RW and IFVP-RW OR IFSP-ESIGN and IFVP-ESIGN	EMSA2 or EMSA3 or EMSA4 or EMSA5 (each uses a hash function; EMSA4 and EMSA5 also use a mask generation function)
<i>Signature schemes giving message recovery</i>		

Table 1—Summary of techniques in the standard (continued)

Scheme name	Primitives	Additional methods
DL/ECSSR	DLPSP-NR2/PV, DLSP-NR2 and DLVP-NR2 OR ECPSP-NR2/PV, ECSP-NR2 and ECVN-NR2	EMSR1 (uses a hash function)
DL/ECSSR-PV	DLPSP-NR2/PV, DLSP-PV and DLVP-PV OR ECPSP-NR2/PV, ECSP-PV and ECVN-PV	EMSR2 (uses a key derivation function and possibly a symmetric encryption scheme); a hash function
IFSSR	IFSP-RSA1 and IFVP-RSA1 OR IFSP-RSA2 and IFVP-RSA2 OR IFSP-RW and IFVP-RW	EMSR3 (uses a hash function and a mask generation function)
<i>Encryption schemes</i>		
DL/ECIES	DL/ECSVDP-DH or -DHC	KDF2; MAC1 (each uses a hash function); possibly a symmetric encryption scheme
IFES	IFEP-RSA and IFDP-RSA	EME1 (uses a hash function and a mask generation function)
IFES-EPOC	IFEP-OU and IFDP-OU	EME2 or EME3 (each uses a hash function and a mask generation function; EME3 also uses a key derivation function and possibly a symmetric encryption scheme)
NOTE—Acronym definitions are as follows: Families—DL: discrete logarithm; EC: elliptic curve; IF: integer factorization Schemes—KAS: key agreement scheme; SSA: signature scheme with appendix; SSR: signature scheme giving message recovery; IES: integrated encryption scheme; ES: encryption scheme Primitives—SVDP: secret value derivation primitive; PSP: pre-signature primitive; SP: signature primitive; VP: verification primitive; EP: encryption primitive; DP: decryption primitive Additional methods—KDF: key derivation function; EMSA: encoding method for signatures with appendix; EMSR: encoding method for signatures giving message recovery; EME: encoding method for encryption; MAC: message authentication code Names of techniques—DH: Diffie-Hellman; DHC: Diffie-Hellman with cofactor exponentiation/multiplication; MQV: Menezes-Qu-Vanstone; MQVC: Menezes-Qu-Vanstone with cofactor exponentiation/multiplication; NR: Nyberg-RueppelDSA: Digital signature algorithm; RSA: Rivest-Shamir-Adleman; RW: Rabin-Williams; PV: Pintsov-Vanstone; OU: Okamoto-Uchiyama		

5. Mathematical conventions

5.1 Mathematical notation

Replace the definition of $GF(q)$ with the following text:

$GF(q)$ The finite field containing q elements. For this standard, q will be either a power of an odd prime p or a power of 2.

Insert the following definition:

$GF(p^m)$ The finite field containing p^m elements for some integer $m > 1$ where p is an odd prime number. Also known as odd-characteristic extension field. See 5.3.3 for more information.

NOTE—Although mathematically, the various techniques based on $GF(p^m)$ may generally also be employed when $m = 1$, for the purposes of this standard, it will be assumed that $m > 1$ when the notation “ $GF(p^m)$ ” or the term “odd-characteristic extension field” is given. The cases $GF(p^m)$ and $GF(p)$ are thus treated separately.

Insert the following subclause after 5.3.2.2:

5.3.2.3 Composite basis over $GF(2^d)$

This representation is possible for fields $GF(2^m)$ if m is a composite integer. The representation is determined by choosing a subfield $GF(2^d)$ of $GF(2^m)$, where $1 < d < m$, and $m = ds$. Elements of $GF(2^m)$ are then represented as vectors of s elements from $GF(2^d)$ using any basis representation, including those in 5.3.2.3.1 or 5.3.2.3.2. Elements of $GF(2^d)$ may in turn be represented in any basis, including those in 5.3.2.1 or 5.3.2.2, or with another composite basis if d is a composite integer. This process of choosing representations for subfields of $GF(2^d)$ may be repeated for each of the divisors of d .

5.3.2.3.1 Polynomial basis over $GF(2^d)$

This representation is similar to that defined in 5.3.2.1 with the difference that $d \geq 1$. This representation is determined by choosing an irreducible polynomial $f(t)$ over $GF(2^d)$ of degree $s = m/d$. Then $GF(2^m)$ is isomorphic to $GF(2^d)[t]/f(t)$.

Then if the polynomial basis representation over $GF(2^d)$ is used, for purposes of conversion, the string

$$(a_{s-1} \dots a_2 a_1 a_0)$$

where each a_i is a bit string of length d shall be taken to represent the polynomial

$$a_{s-1}t^{s-1} + \dots + a_2t^2 + a_1t + a_0$$

where the coefficients a_i are elements of $GF(2^d)$. The coefficients a_i will in turn be represented with some basis over some subfield of $GF(2^d)$ [perhaps $GF(2)$].

As in 5.3.2.1, the additive identity (zero) element of the field $GF(2^m)$ is represented by a string of s representations of the zero element in the chosen representation for $GF(2^d)$. Thus, if $GF(2^d)$ is represented with a polynomial or normal basis, the zero element of $GF(2^m)$ is represented by a string of all 0 bits. Similarly, the multiplicative identity (one) element of $GF(2^m)$ is represented by a string of $(s - 1)$ representations of the zero element of $GF(2^d)$ followed by the representation of the one element of $GF(2^d)$. Thus, if $GF(2^d)$ is rep-

resented with a polynomial basis, the one element of $GF(2^m)$ is represented by a string of $m - 1$, 0 bits followed by a 1-bit. If $GF(2^d)$ is represented by a normal basis, the one element of $GF(2^m)$ is represented by a string of $(s - 1)d$ 0-bits followed by d 1-bits.

The arithmetic in this basis is identical to that of 5.3.2.1 with the exception that all coefficient operations are performed in the field $GF(2^d)$.

5.3.2.3.2 Normal basis over $GF(2^d)$

This representation is similar to that defined in 5.3.2.2 with the difference that $d \geq 1$. This representation is determined by choosing a normal polynomial $f(t)$ of degree $s = m/d$. Then $GF(2^m)$ is isomorphic to $GF(2^d)[t]/f(t)$.

Then if the normal basis representation over $GF(2^d)$ is used, for purposes of conversion, the string

$$(a_0 a_1 \dots a_{s-1})$$

where each a_i is a bit string of length d , shall be taken to represent the element

$$a_0\theta + a_1\theta^2 + a_2\theta^{2^2} + \dots + a_{s-1}\theta^{2^{s-1}(2)}$$

where q is a root of $f(t)$ in $GF(2^s)$. The coefficients a_i will in turn be represented with some basis over some subfield of $GF(2^d)$ [perhaps $GF(2)$].

As in 5.3.2.2, the additive identity (zero) element of the field $GF(2^m)$ is represented by a string of s representations of zero in $GF(2^d)$. For example, if $GF(2^d)$ is represented with a polynomial or normal basis, a string of 0 bits represents 0 in $GF(2^m)$. The multiplicative identity (one) element of the field is represented by a string of s representations of one in $GF(2^d)$. For example, if $GF(2^d)$ is represented with a polynomial basis, the string consists of the pattern: $(d - 1)$ 0 bits followed by a 1-bit. The pattern repeated s times represents one in $GF(2^m)$. If $GF(2^d)$ is represented with a normal basis, the one element of $GF(2^m)$ is represented with a string of all 1-bits.

The arithmetic in this basis is identical to that of 5.3.2.1 with the exception that all coefficient operations are performed in the field $GF(2^d)$.

Insert the following subclause after 5.3.2:

5.3.3 Odd-characteristic extension field

An *odd-characteristic extension field* is a finite field whose number of elements is a power of an odd prime. If $m > 1$, then there is a unique field (up to isomorphism) $GF(p^m)$ with p^m elements. For the purposes of conversion, the elements of $GF(p^m)$ shall be represented in a polynomial basis and converted to integers in the set $\{0, 1, \dots, p^m - 1\}$ as follows. The polynomial basis representation is determined by choosing an irreducible polynomial $f(t)$ over $GF(p)$. Then $GF(p^m)$ is isomorphic to $GF(p)[t]/f(t)$. The element with representation

$$a_{m-1}t^{m-1} + \dots + a_2t^2 + a_1t + a_0$$

shall be represented by the integer

$$a_{m-1}p^{m-1} + \dots + a_2p^2 + a_1p + a_0$$

(note that each a^i is between 0 and $p - 1$, so the result is between 0 and $p^m - 1$). (The coefficients can be recovered from this integer by successively dividing by p and keeping the remainder. For instance, denoting the integer by a , the coefficient a_0 can be obtained as $a_0 = a \bmod p$, then a can be updated to $a = \lfloor a/p \rfloor$, then a_1 can be obtained as $a_1 = a \bmod p$, and so on.)

Note that for the purposes of conversion, the additive identity (zero) element of the field is represented by the integer 0, and the multiplicative identity (one) element of the field is represented by the integer 1.

A description of the arithmetic of $GF(p^m)$ is given in A.17.

5.4 Elliptic curves and points

Replace the second and third paragraphs and equations with the following text:

If $q = p^m$, $p > 3$, $m \geq 1$, then a and b shall satisfy $4a^3 + 27b^2 \neq 0$ in $GF(q)$, and every point $P = (x_P, y_P)$ on E (other than the point O) shall satisfy the following equation in $GF(q)$:

$$y_P^2 = x_P^3 + ax_P + b$$

If q is a power of 2, then b shall be nonzero in $GF(q)$, and every point $P = (x_P, y_P)$ on E (other than the point O) shall satisfy the following equation in $GF(q)$:

$$y_P^2 + x_P y_P = x_P^3 + ax_P^2 + b$$

If q is a power of 3, then a and b shall be nonzero in $GF(q)$, and every point $P = (x_P, y_P)$ on E (other than the point O) shall satisfy the following equation in $GF(q)$:

$$y_P^2 = x_P^3 + ax_P^2 + b$$

5.5 Data type conversion

Replace Figure 1 with the following:

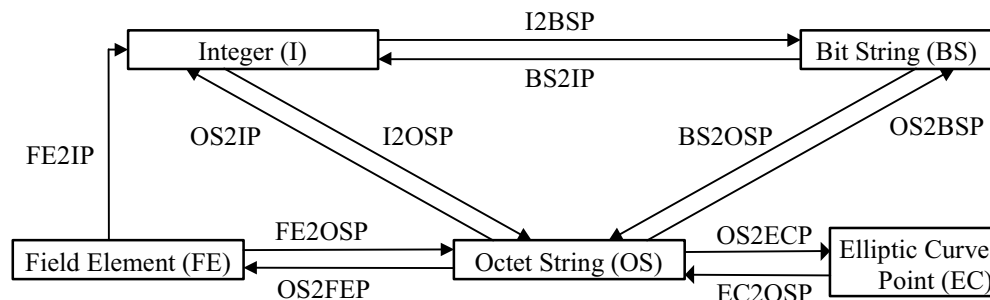


Figure 1—Summary of data type conversion primitives

Replace 5.5.4 with the following text:

5.5.4 Converting between finite field elements and octet strings (FE2OSP and OS2FEP)

An element x of a finite field $GF(q)$, for the purposes of conversion, is represented by an integer if q is an odd prime or an odd prime power (see 5.3.1 and 5.3.3) or by a bit string if q is a power of 2 (see 5.3.2). If q is an odd prime or an odd prime power, then to represent x as an octet string, I2OSP shall be used with the integer value representing x and the length $\lceil \log_{256} q \rceil$ as inputs. If q is a power of 2, then to represent x as an octet string, BS2OSP shall be applied to the bit string representing x .

The primitive that converts finite field elements to octet strings is called the Field Element to Octet String Conversion Primitive or FE2OSP. It takes a field element x , the field size q , and both the field characteristic p and the extension degree m if q is an odd prime-power as inputs and outputs the corresponding octet string.

To convert an octet string back to a field element, if q is an odd prime or an odd prime power, then OS2IP shall be used with the octet string as the input. If q is a power of 2, then OS2BSP shall be used with the octet string and the length $\log_2 q$ as inputs. The primitive that converts octet strings to finite field elements is called the Octet String to Field Element Conversion Primitive or OS2FEP. It takes the octet string and the field size q as inputs and outputs the corresponding field element. It shall output “error” if OS2BSP or OS2IP outputs “error.”

Insert the following subclause after 5.5.5:

5.5.6 Converting between elliptic curve points and octet strings

An elliptic curve point P (which is not the point at infinity O) can be represented in either compressed or uncompressed form. (For internal calculations, it may be advantageous to use other representations, e.g., the projective coordinates of A.9.6. Also see A.9.6 for more information on point compression.) The uncompressed form of P is simply given by its two coordinates. The compressed form is presented in 5.5.6.1. The octet string format is defined to support both compressed and uncompressed points.

5.5.6.1 Compressed elliptic curve points

The *compressed form* of an elliptic curve point $P \neq O$ defined over $GF(p)$ and $GF(2^m)$ is the pair $(x_P, \tilde{y}_P)$ where x_P is the x -coordinate of P , and $\tilde{y}_P$ is a bit that is computed as defined next.

Two compressed forms are defined: an *LSB compressed form* defined for elliptic curves over $GF(p)$ and $GF(2^m)$, and an *SORT compressed form* defined for elliptic curves over $GF(2^m)$ and $GF(p^m)$.

NOTE—Elliptic curve points over $GF(2^m)$ can be compressed with either the LSB or SORT method. The LSB method is equivalent to the point compression method in Annex E of IEEE Std 1363-2000.

5.5.6.1.1 LSB compressed form

Over $GF(p)$, the LSB compressed form has $\tilde{y}_P = \text{FE2IP}(y_P) \bmod 2$. Put another way, $\tilde{y}_P$ is the rightmost bit of y_P .

Over $GF(2^m)$, the LSB compressed form has $\tilde{y}_P = 0$ if $x_P = 0$ and $\tilde{y}_P = \text{FE2IP}(y_P x_P^{-1}) \bmod 2$ if $x_P \neq 0$. That is, $\tilde{y}_P$ is the rightmost bit of the field element $y_P x_P^{-1}$ if $x_P \neq 0$. (This rule applies for any of the basis representations given in 5.3.2.)

Procedures for *point decompression* (i.e., recovering y_P given x_P and $\tilde{y}_P$) are given in A.12.8 (for q an odd prime) and A.12.9 (for q a power of 2).

NOTE—The name “LSB” refers to the fact that the compressed bit is the least significant bit of the integer representation of y_P or $y_P x_P^{-1}$.

5.5.6.1.2 SORT compressed form

Let (x_P, y_P) be the inverse of the point (x_P, y_P) . (For $GF(p^m)$, $y'_P = -y_P$; for $GF(2^m)$, $y'_P = x_P + y_P$).

The SORT compressed form has $\tilde{y}_P = 1$ if $\text{FE2IP}(y_P) > \text{FE2IP}(y'_P)$, and $\tilde{y}_P = 0$ otherwise.

A procedure for point decompression is given in A.12.11.

NOTES

1—It may be more efficient to determine $\tilde{y}_P$ by comparing the coefficients of y_P and y'_P directly, rather than first computing $\text{FE2IP}(y_P)$ and $\text{FE2IP}(y'_P)$.

2—Although the SORT compressed form is defined here for any field, in the representations in 5.5.6.2, it is only employed for elliptic curves over $GF(2^m)$ and $GF(p^m)$.

3—The name “SORT” refers to the fact that the compressed bit is based on comparing the integer representations of y_P and y'_P .

5.5.6.2 Two-coordinate point representations

For all conversion primitives in this subclause, the point O shall be represented by an octet string containing a single 0 octet. The rest of this subclause discusses octet string representation of a point $P \neq O$. Let the x -coordinate of P be x_P and the y -coordinate of P be y_P . Let $(x_P, \tilde{y}_P)$ be the compressed representation of P in one of the forms above.

The representations in this subclause are all “lossless”; i.e., the elliptic curve point can be uniquely recovered from its octet string representation, because both coordinates are represented.

An octet string PO representing P shall have one of the following three formats: *compressed*, *uncompressed*, or *hybrid*. (The hybrid format contains information of both compressed and uncompressed form.) For all primitives in this subclause, PO shall have the following general form:

$$PO = PC \| X \| Y$$

where

PC is a single octet of the form 0000SUC $\tilde{Y}$ defined as follows:

- Bit S is 1 if the format uses the SORT compressed form; 0 otherwise.
- Bit U is 1 if the format is uncompressed or hybrid; 0 otherwise.
- Bit $\tilde{C}$ is 1 if the format is compressed or hybrid; 0 otherwise.
- Bit $\tilde{Y}$ is equal to the bit $\tilde{y}_P$ if the format is compressed or hybrid; 0 otherwise.

X is the octet string of length $\lceil \log_{256} q \rceil$ representing x_P according to FE2OSP (see 5.5.4).

Y is the octet string of length $\lceil \log_{256} q \rceil$ representing y_P of P according to FE2OSP (see 5.5.4) if the format is uncompressed or hybrid; Y is an empty string if the format is compressed.

The primitive that converts elliptic curve points to octet strings for a given representation is called Elliptic Curve Point to Octet String Conversion Primitive- R , or EC2OSP- R , where R is the representation. It takes an elliptic curve point P and the size q of the underlying field as input and outputs the corresponding octet string PO .

The primitive that converts octet strings to elliptic curve points is called Octet String to Elliptic Curve Point Conversion Primitive- R , or OS2ECP- R . It takes the octet string and the field size q as inputs and outputs the corresponding elliptic curve point, or “error.” It shall use OS2FEP to get x_P . It shall use OS2FEP to get y_P if the format is uncompressed, and it may output “error” if the recovered point is not on the elliptic curve. It shall use point decompression (see 5.5.6.1) to get y_P if the format is compressed. It can get y_P by either of these two means if the format is hybrid, and if the format is hybrid, it may output “error” if different values are obtained by the two means. It shall output “error” in the following cases:

- If the first octet is not as expected for the representation
- If the octet string length is not as expected for the representation
- If an invocation of OS2FEP outputs “error”
- If an invocation of the point decompression algorithm outputs “error”

The pairs of primitives for each of five representations are defined in the following subclauses.

5.5.6.2.1 Uncompressed representation: EC2OSP-XY and OS2ECP-XY

This representation is defined for elliptic curves over all finite fields in this standard.

In this representation, the octet PC shall have binary value 0000 0100 and the octet strings X and Y shall represent x_P and y_P , respectively. The length of the octet string PO shall be $1 + 2 \lceil \log_{256} q \rceil$. The corresponding primitives are called EC2OSP-XY and OS2ECP-XY.

5.5.6.2.2 LSB compressed representation: EC2OSP-XL and OS2ECP-XL

This representation is defined for elliptic curves over $GF(p)$ and $GF(2^m)$ only.

In this representation, the octet PC shall have binary value 0000 001 $\tilde{Y}$ where $\tilde{Y}$ is equal to the bit $\tilde{y}_P$ in the LSB compressed form, the octet string X shall represent x_P , and the octet string Y shall be the empty string. The length of the octet string PO shall be $1 + \lceil \log_{256} q \rceil$. The corresponding primitives are called EC2OSP-XL and OS2ECP-XL.

5.5.6.2.3 SORT compressed representation: EC2OSP-XS and OS2ECP-XS

This representation is defined for elliptic curves over $GF(2^m)$ and $GF(p^m)$ only.

In this representation, the octet PC shall have binary value 0000 101 $\tilde{Y}$ where $\tilde{Y}$ is equal to the bit $\tilde{y}_P$ in the SORT compressed form, the octet string X shall represent x_P , and the octet string Y shall be the empty string. The length of the octet string PO shall be $1 + \lceil \log_{256} q \rceil$. The corresponding primitives are called EC2OSP-XS and OS2ECP-XS.

5.5.6.2.4 LSB hybrid representation: EC2OSP-XYL and OS2ECP-XYL

This representation is defined for elliptic curves over $GF(p)$ and $GF(2^m)$ only.

In this representation, the octet PC shall have binary value 0000 011 $\tilde{Y}$ where $\tilde{Y}$ is equal to the bit $\tilde{y}_P$ in the LSB compressed form, and the octet strings X and Y shall represent x_P and y_P , respectively. The length of the octet string PO shall be $1 + 2 \lceil \log_{256} q \rceil$. The corresponding primitives are called EC2OSP-XYL and OS2ECP-XYL.

5.5.6.2.5 SORT hybrid representation: EC2OSP-XYS and OS2ECP-XYS

This representation is defined for elliptic curves over $GF(2^m)$ and $GF(p^m)$ only.

In this representation, the octet PC shall have binary value $0000\ 111\ \tilde{Y}$ where $\tilde{Y}$ is equal to the bit $\tilde{y}_p$ in the SORT compressed form, and the octet strings X and Y shall represent x_p and y_p , respectively. The length of the octet string PO shall be $1 + 2 \lceil \log_{256} q \rceil$. The corresponding primitives are called EC2OSP-XYS and OS2ECP-XYS.

5.5.6.3 x-coordinate only representation: EC2OSP-X and OS2ECP-X

The x -coordinate only representation in this subclause is “lossy”; i.e., the elliptic curve point *cannot* be uniquely recovered from its octet string representation, because only the x -coordinate is represented.

This representation is defined for elliptic curves over all fields in this standard.

In this representation, the octet PC shall have binary value $0000\ 0001$, the octet string X shall represent x_p , and the octet string Y shall be empty. The length of the octet string PO shall be $1 + \lceil \log_{256} q \rceil$. The corresponding primitives are called EC2OSP-X and OSC2ECP-X.

OS2ECP-X may output any of the (at most two) elliptic curve points with the given x -coordinate. Thus, the original y -coordinate is not necessarily recovered.

This representation should be employed only if the recipient of the octet string PO does not need to resolve the ambiguity in the y -coordinate, or it can do so by other means.

NOTE—In some situations, only the x -coordinate is needed. For instance, the shared secret value computed by ECSVDP-DH or ECSVDP-DHC depends only on the x -coordinate of the other party’s public key, not the y -coordinate. If this representation is employed in such a situation, then when the “octet-string-to-point” conversion primitive is called, the implementation need not compute a y -coordinate at all (although it may output “error” if no point exists with the given x -coordinate).

5.5.6.4 Summary of representations

Table 2 summarizes the point representations in 5.5.6.2 and 5.5.6.3.

Table 2—Elliptic curve point representations

Representation	Primitives	PC	X	Y	Finite fields
Uncompressed	EC2OSP-XY and OS2ECP-XY	0000 0100	x_P	y_P	All
LSB compressed	EC2OSP-XL and OS2ECP-XL	0000 001 $\tilde{Y}$	x_P	Empty	$GF(p)$ and $GF(2^m)$
SORT compressed	EC2OSP-XS and OS2ECP-XS	0000 011 $\tilde{Y}$	x_P	Empty	$GF(2^m)$ and $GF(p^m)$
LSB hybrid	EC2OSP-XYL and OS2ECP-XYL	0000 011 $\tilde{Y}$	x_P	y_P	$GF(p)$ and $GF(2^m)$
SORT hybrid	EC2OSP-XYS and OS2ECP-XYS	0000 111 $\tilde{Y}$	x_P	y_P	$GF(2^m)$ and $GF(p^m)$
x-coordinate-only	EC2OSP-X and OS2ECP-X	0000 0001	x_P	Empty	All
Point O	All	0000 0000	Empty	Empty	All
NOTES 1—The first four bits of the first octet PC are reserved and may be used in future formats defined in an amendment to, or in future versions of, this standard. It is essential that they be set to 0 and checked for 0 in order to distinguish the formats defined here from other formats. Of course, implementations may support other, nonstandard formats that employ the reserved bits, but these formats would not conform with the ones defined in this clause. 2—The various representations employ distinct values for the first octet PC, so the octet strings produced by the different representations are nonoverlapping, except at the point O. Consequently, a “generic” OS2ECP primitive may be constructed that handles all of the representations.					

6. Primitives based on the discrete logarithm problem

6.1.1 Notations

Insert the following entries in the list of notation:

(c, d)	Signature, a pair of integers, computed by a signature primitive
c	Signature part, an integer, computed by a signature primitive
d	Signature part, an integer, computed by a signature primitive
h	Randomized hash value, an integer
i	Pre-signature, an integer
u	Randomizer, an integer
C	Encrypted message, an octet string, computed by an encryption scheme
M_1	Recoverable message part, an octet string, the part of a message that can be recovered from the signature in a signature scheme giving message recovery
M_2	Nonrecoverable message part (if any), an octet string, the part of a message that cannot be recovered from the signature in a signature scheme giving message recovery
T	Authentication tag, a bit string, computed by an encryption scheme

6.1.2 DL domain parameters

Replace the first paragraph with the following text:

DL *domain parameters* are used in every DL primitive and scheme and are an implicit component of every DL key. A set of DL domain parameters specifies a field $GF(q)$, where q is a positive odd prime integer p , 2^m for some positive integer m , or p^m for an odd prime p and some integer $m \geq 2$; a positive prime integer r dividing $(q - 1)$; and a field element g of multiplicative order r (g is called the *generator* of the subgroup of order r). If $q = 2^m$, it also specifies a representation for the elements of $GF(q)$ to be used by the conversion primitives (see 5.3 and 5.5). Implicitly it also specifies the cofactor $k = (q - 1) / r$. If the DLSVDP-DHC or DLSVDP-MQVC primitive is to be applied, then it shall also be the case that $\text{GCD}(k, r) = 1$ (i.e., r does not divide k ; see A.1.1.1).

6.2 Primitives

6.2.5 DLSP-NR

Replace the first paragraph with the following text:

DLSP-NR is the Discrete Logarithm Signature Primitive, Nyberg-Rueppel version. It is based on the work of Nyberg and Rueppel [B120].⁵ It can be invoked in a scheme to compute a signature on a message representative with the private key of the signer, in such a way that the message representative can be recovered from the signature using the public key of the signer by the DLVP-NR primitive. Note, however, that the DLPSP-NR2/PV and DLSP-NR2 primitives are intended for signature schemes giving message recovery in this version of the standard. DLSP-NR can be invoked in the scheme DLSSA as part of signature generation.

⁵The numbers in brackets correspond to those in the bibliography in Annex G.

Replace the note with the following text:

NOTES

1—The key pair in step 1) should be a one-time key pair that is generated and stored by the signer following the security recommendations of D.3.1, D.4.1.2, D.6, and D.7. The private key should be selected in an unbiased manner (see D.5.2.1 for further discussion), and a new key pair should be generated for every signature. The one-time private key u should be securely deleted after step 4), as its recovery by an opponent can lead to the recovery of the private key s .

2—The one-time key pair in step 1) may be precomputed in advance of the availability of the message representative. See D.5.2.1 for further discussion.

6.2.6 DLVP-NR

Replace the first paragraph with the following text:

DLVP-NR is the Discrete Logarithm Verification Primitive, Nyberg–Rueppel version. It is based on the work of Nyberg and Rueppel [B120]. This primitive recovers the message representative that was signed with DLSP-NR, given only the signature and public key of the signer. It can be invoked in a scheme as part of signature verification and, possibly, message recovery. Note, however, that the DLVP-NR2 primitive is intended for signature schemes giving message recovery in this version of the standard. DLVP-NR can be invoked in the scheme DLSSA as part of signature verification.

6.2.7 DLSP-DSA

Replace the note with the following text:

NOTES

1—The key pair in step 1) should be a one-time key pair that is generated and stored by the signer following the security recommendations of D.3.1, D.4.1.2, D.6, and D.7. The private key should be selected in an unbiased manner (see D.5.2.1 for further discussion), and a new key pair should be generated for every signature. The one-time private key u should be securely deleted after step 4), as its recovery by an opponent can lead to the recovery of the private key s .

2—Similar to DLSP-NR, the one-time key pair in step 1) may be precomputed in advance of the availability of the message representative. Step 2) and step 3) as well as the values u^{-1} and sc in step 4) may also be precomputed. See D.5.2.1 for further discussion.

6.2.8 DLVP-DSA

Insert the following five subclauses (6.2.9–6.2.13) after 6.2.8:

6.2.9 DLPSP-NR2/PV

DLPSP-NR2/PV is the Discrete Logarithm Pre-Signature Primitive, Nyberg–Rueppel/Pintsov–Vanstone version. It is based on the work of ISO/IEC 9796-3:2000 [B79], Nyberg and Rueppel [B120], and Pintsov and Vanstone [PV99]. The primitive generates a randomizer and a pre-signature for a signature scheme. It can be invoked in the scheme DLSSR or DLSSR-PV as part of signature generation. DLPSP-NR2/PV corresponds to the first two steps of DLSP-NR.

Input: The DL domain parameters q , r , and g

Assumptions: DL domain parameters q , r , and g are valid.

Output:

- The randomizer, which is an integer u such that $1 \leq u < r$
- The pre-signature, which is an integer i such that $1 \leq i < q$

Operation: The randomizer u and the pre-signature i shall be computed by the following or an equivalent sequence of steps:

- 1) Generate a key pair (u, v) with the set of domain parameters. (See Notes 1 and 2 below.)
- 2) Convert v into an integer i with primitive FE2IP [recall that v is an element of $GF(q)$].
- 3) Output u as the randomizer and i as the pre-signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of DL domain parameters q , r , and g

NOTES

1—The key pair in step 1) should be a one-time key pair that is generated by the signer following the security recommendations of D.3.1, D.4.1.2, D.6, and D.7. The randomizer should be selected in an unbiased manner (see D.5.2.1 for further discussion), and a new randomizer/pre-signature pair should be generated for every signature. (If signatures are computed with the same randomizer for two or more different message representatives, or if the randomizer becomes known, an opponent may be able to recover the signer's private key.) The randomizer should be stored following the same security recommendations as for an ephemeral private key (it should be securely deleted after it is used in DLSP-NR2 or DLSP-PV).

2—Similar to DLSP-NR, the key pair may be precomputed in advance of the availability of the message representative.

6.2.10 DLSP-NR2

DLSP-NR2 is the Discrete Logarithm Signature Primitive, Nyberg–Rueppel version 2. It is based on the work of ISO/IEC 9796-3:2000 [B79] and Nyberg and Rueppel [B120]. The primitive generates a signature on a message representative with the private key of the signer, given a randomizer and a pre-signature generated by DLPSP-NR2/PV, in such a way that the message representative and the pre-signature can be recovered from the signature using the public key of the signer by the DLVP-NR or DLVP-NR2 primitive. It can be invoked in the scheme DLSSR as part of signature generation. DLSP-NR2 corresponds to the last three steps of DLSP-NR.

Except for having DL domain parameters as input rather than EC domain parameters, the primitive is identical to ECSP-NR2.

Input:

- The DL domain parameters q , r , and g associated with the key s
- The signer's private key s
- The randomizer, which is an integer u such that $1 \leq u < r$
- The pre-signature, which is an integer i such that $1 \leq i < q$
- The message representative, which is an integer f such that $0 \leq f < r$

Assumptions: Private key s and DL domain parameters q , r , and g are valid and associated with each other; $1 \leq u < r$; $1 \leq i < q$; $0 \leq f < r$.

Output: The signature, which is a pair of integers (c, d) such that $1 \leq c < r$ and $0 \leq d < r$; or “error”

Operation: The signature (c, d) shall be computed by the following or an equivalent sequence of steps:

- 1) Compute an integer $c = i + f \bmod r$. If $c = 0$, then output “error.” (This is an extremely rare event for recommended values of r —see Note 2.)
- 2) Compute an integer $d = u - sc \bmod r$.
- 3) Output the pair (c, d) as the signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of DL domain parameters q, r , and g
- At least one valid private key s for each set of domain parameters
- All randomizers u in the range $[1, r - 1]$
- All pre-signatures i in the range $[1, q - 1]$
- All message representatives f in the range $[0, r - 1]$

NOTES

1—Following the discussion for DLPSP-NR2/PV in this subclause, for security reasons, the randomizer u should be securely deleted after step 2).

2—The probability that the primitive will output “error” is about $1/r$ (i.e., essentially 0 for recommended values of r), assuming the pre-signature i is generated at random by DLPSP-NR2/PV. Thus, the check at the end of step 1) is included for completeness only; it is unlikely to occur in practice. The motivation for the check is to ensure that the signature depends on the signer’s private key s ; otherwise the signature could be verified with anyone’s public key with the same domain parameters. [An “error” output in any case does not disclose any information about the private key s , because the private key is not involved in the pre-signature primitive or prior to step 2).]

6.2.11 DLVP-NR2

DLVP-NR2 is the Discrete Logarithm Verification Primitive, Nyberg–Rueppel version 2. It is based on the work of ISO/IEC 9796-3:2000 [B79] and Nyberg and Rueppel [B120]. This primitive recovers the message representative that was signed with DLSP-NR or DLSP-NR2, given only the signature and public key of the signer, and also recovers the pre-signature. It can be invoked in the scheme DLSSR as part of signature verification and message recovery. DLVP-NR2 is the same as DLVP-NR, except that the recovered pre-signature is output in addition to the message representative.

Input:

- The DL domain parameters q, r , and g associated with the key w
- The signer’s public key w
- The signature to be verified, which is a pair of integers (c, d)

Assumptions: Public key w and DL domain parameters q, r , and g are valid and associated with each other.

Output:

- The message representative, which is an integer f such that $0 \leq f < r$; or “invalid”
- The pre-signature, which is an integer i such that $1 \leq i < q$

Operation: The message representative f and the pre-signature i shall be computed by the following or an equivalent sequence of steps:

- 1) If c is not in $[1, r - 1]$ or d is not in $[0, r - 1]$, output “invalid” and stop.
- 2) Compute a field element $j = \exp(g, d) \times \exp(w, c)$.
- 3) Convert the field element j to an integer i with primitive FE2IP.
- 4) Compute an integer $f = c - i \bmod r$.
- 5) Output f as the message representative and i as the pre-signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of DL domain parameters q , r , and g
- At least one valid public key w for each set of domain parameters
- All purported signatures (c, d) that can be input to the implementation; this should at least include all (c, d) such that c and d are in the range $[0, r - 1]$

6.2.12 DLSP-PV

DLSP-PV is the Discrete Logarithm Signature Primitive, Pintsov–Vanstone version. It is based on the work of Pintsov and Vanstone [PV99]. The primitive generates a signature part on a randomized hash value with the private key of the signer, given a randomizer generated by DLPSP-NR2/PV, in such a way that the pre-signature generated by DLPSP-NR2/PV can be recovered from the signature part using the public key of the signer by the DLVP-PV primitive. It can be invoked in the scheme DLSSR-PV as part of signature generation.

Except for having DL domain parameters as input rather than EC domain parameters, the primitive is identical to ECSP-PV.

Input:

- The DL domain parameters q , r , and g associated with the key s
- The signer’s private key s
- The randomizer, which is an integer u such that $1 \leq u < r$
- The randomized hash value, which is a non-negative integer h

Assumptions: Private key s and DL domain parameters q , r , and g are valid and associated with each other; $1 \leq u < r$; $h \geq 0$.

Output: A signature part, which is an integer d , where $0 \leq d < r$

Operation: The signature part d shall be computed by the following or an equivalent sequence of steps:

- 1) Compute an integer $d = u - sh \bmod r$.
- 2) Output d as the signature part.

Conformance region recommendation: A conformance region should include:

- At least one valid set of DL domain parameters q , r , and g
- At least one valid private key s for each set of domain parameters
- All randomizers u in the range $[1, r - 1]$
- All randomized hash values h in the range $[0, r - 1]$, where r is from the domain parameters of s

NOTES

1—Following the discussion for DLPSP-NR2/PV in this subclause, for security reasons, the randomizer u should be securely deleted after step 1).

2—This primitive does not check whether $h = 0$ to ensure that the signature depends on the signer's public key, because it is such an unlikely occurrence in the scheme in which DLSP-PV is employed, but an implementation could do so if desired.

6.2.13 DLVP-PV

DLVP-PV is the Discrete Logarithm Verification Primitive, Pintsov–Vanstone version. It is based on the work of Pintsov and Vanstone [PV99]. This primitive recovers a pre-signature from a signature part generated with DLSP-PV, given only the randomized hash value and the public key of the signer. It can be invoked in the scheme DLSSR-PV as part of signature verification and message recovery.

Input:

- The DL domain parameters q , r , and g associated with the key w
- The signer's public key w
- The randomized hash value, which is an integer $h \geq 0$
- The signature part from which the pre-signature is to be recovered, which an integer d

Assumptions: Public key w and DL domain parameters q , r , and g are valid and associated with each other.

Output: The pre-signature, which is an integer i such that $1 \leq i < q$

Operation: The pre-signature i shall be computed by the following or an equivalent sequence of steps:

- 1) If d is not in $[0, r - 1]$, output “invalid” and stop.
- 2) Compute a field element $j = \exp(g, d) \times \exp(w, h)$.
- 3) Convert the field element j to an integer i with primitive FE2IP.
- 4) Output i as the pre-signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of DL domain parameters q , r , and g
- At least one valid public key w for each set of domain parameters
- All randomized hash values $h \geq 0$
- All signature parts d that can be input to the implementation; this should at least include all d such that d is in the range $[0, r - 1]$

7. Primitives based on the elliptic curve discrete logarithm problem

7.1.1 Notations

Insert the following entries in the list of notation:

(c, d)	Signature, a pair of integers, computed by a signature primitive
c	Signature part, an integer, computed by a signature primitive
d	Signature part, an integer, computed by a signature primitive
h	Randomized hash value, an integer
i	Pre-signature, an integer
u	Randomizer, an integer
C	Encrypted message, an octet string, computed by an encryption scheme
M_1	Recoverable message part, an octet string, the part of a message that can be recovered from the signature in a signature scheme giving message recovery
M_2	Nonrecoverable message part (if any), an octet string, the part of a message that cannot be recovered from the signature in a signature scheme giving message recovery
T	Authentication tag, a bit string, computed by an encryption scheme

7.1.2 EC domain parameters

EC domain parameters are used in every EC primitive and scheme and are an implicit component of every EC key. A set of EC domain parameters specifies a field $GF(q)$, where q is a positive odd prime integer p , 2^m for some positive integer m , or p^m for an odd prime p and some integer $m \geq 2$; two elliptic curve coefficients a and b , elements of $GF(q)$, that define an elliptic curve E ; a positive prime integer r dividing the number of points on E ; and a curve point G of order r (G is called the generator of a subgroup of order r). If $q = 2^m$, it also specifies a representation for the elements of $GF(q)$ to be used by the conversion primitives (see 5.3 and 5.5). Implicitly it also specifies the cofactor $k = \#E/r$ (where $\#E$ is the number of points on E). If key validation is to be performed or if the ECSVDP-DHC or ECSVDP-MQVC primitive is to be applied, then it shall also be the case that $\text{GCD}(k, r) = 1$ (i.e., r does not divide k ; see A.1.1).

7.2 Primitives

7.2.5 ECSP-NR

Replace the first paragraph with the following text:

ECSP-NR is the Elliptic Curve Signature Primitive, Nyberg–Rueppel version. It is based on the work of Koblitz [B94], Miller [B117], and Nyberg and Rueppel [B120]. It can be invoked in a scheme to compute a signature on a message representative with the private key of the signer, in such a way that the message representative can be recovered from the signature using the public key of the signer by the ECVP-NR primitive. Note, however, that the ECSP-NR2/PV and ECSP-NR2 primitives are intended for signature schemes giving message recovery in this version of the standard. ECSP-NR can be invoked in the scheme ECSSA as part of signature generation.

Replace the note with the following text:

NOTES

1—The key pair in step 1) should be a one-time key pair that is generated and stored by the signer following the security recommendations of D.3.1, D.4.2.2, D.6, and D.7. The private key should be selected in an unbiased manner (see D.5.2.1 for further discussion), and a new key pair should be generated for every signature. The one-time private key u should be securely deleted after step 4), as its recovery by an opponent can lead to the recovery of the private key s .

2—The one-time key pair in step 1) may be precomputed in advance of the availability of the message representative. See D.5.2.1 for further discussion.

7.2.6 ECVP-NR

Replace the first paragraph with the following text:

ECVP-NR is the Elliptic Curve Verification Primitive, Nyberg–Rueppel version. It is based on the work of Koblitz [B94], Miller [B117], and Nyberg and Rueppel [B120]. This primitive recovers the message representative that was signed with ECSP-NR, given only the signature and public key of the signer. It can be invoked in a scheme as part of signature verification and, possibly, message recovery. Note, however, that the ECVP-NR2 primitive is intended for signature schemes giving message recovery in this version of the standard. ECVP-NR can be invoked in the scheme ECSSA as part of signature verification.

7.2.7 ECSP-DSA

Replace the note with the following text:

NOTES

1—The key pair in step 1) should be a one-time key pair that is generated and stored by the signer following the security recommendations of D.3.1, D.4.2.2, D.6, and D.7. The private key should be selected in an unbiased manner (see D.5.2.1 for further discussion), and a new key pair should be generated for every signature. The one-time private key u should be securely deleted after step 4), as its recovery by an opponent can lead to the recovery of the private key s .

2—Similar to ECSP-NR, the one-time key pair in step 1) may be precomputed in advance of the availability of the message representative. Step 2) and step 3) as well as the values u^{-1} and sc in step 4) may also be precomputed. See D.5.2.1 for further discussion.

Insert the following five subclauses (7.2.9–7.2.13) after 7.2.8:

7.2.9 ECPSP-NR2/PV

ECPSP-NR2/PV is the Elliptic Curve Pre-Signature Primitive, Nyberg–Rueppel and Pintsov–Vanstone version. It is based on the work of ISO/IEC 9796-3:2000 [B79], Koblitz [B94], Miller [B117], and Nyberg and Rueppel [B120]. The primitive generates a randomizer and a pre-signature for a signature scheme. It can be invoked in the scheme ECSSR or ECSSR-PV as part of signature generation. ECPSP-NR2/PV corresponds to the first two steps of ECSP-NR.

Input: The EC domain parameters q , a , b , r , and G .

Assumptions: EC domain parameters q , a , b , r , and G are valid.

Output:

- The randomizer, which is an integer u such that $1 \leq u < r$
- The pre-signature, which is an integer i such that $1 \leq i < q$

Operation: The randomizer u and the pre-signature i shall be computed by the following or an equivalent sequence of steps:

- 1) Generate a key pair (u, V) with the set of domain parameters. (See Notes 1 and 2 below.) Let $V = (x_V, y_V)$ ($V \neq O$ because V is a public key).
- 2) Convert x_V into an integer i with primitive FE2IP [recall that x_V is an element of $GF(q)$].
- 3) Output u as the randomizer and i as the pre-signature.

Conformance region recommendation. A conformance region should include:

- At least one valid set of EC domain parameters q, a, b, r , and G

NOTES

1—The key pair in step 1) should be a one-time key pair that is generated by the signer following the security recommendations of D.3.1, D.4.1.2, D.6, and D.7. The randomizer should be selected in an unbiased manner (see D.5.2.1 for further discussion), and a new randomizer/pre-signature pair should be generated for every signature. (If signatures are computed with the same randomizer for two or more different message representatives, or if the randomizer becomes known, an opponent may be able to recover the signer's private key.) The randomizer should be stored following the same security recommendations as for an ephemeral private key (it should be securely deleted after it is used in ECSP-NR2 or ECSP-PV).

2—Similar to ECSP-NR, the key pair may be precomputed in advance of the availability of the message representative.

7.2.10 ECSP-NR2

ECSP-NR2 is the Elliptic Curve Signature Primitive, Nyberg–Rueppel version 2. It is based on the work of ISO/IEC 9796-3:2000 [B79], Koblitz [B94], Miller [B117], and Nyberg and Rueppel [B120]. The primitive generates a signature on a message representative with the private key of the signer, given a randomizer and a pre-signature generated by ECSP-NR2/PV, in such a way that the message representative and the pre-signature can be recovered from the signature using the public key of the signer by the ECVP-NR or ECVP-NR2 primitive. It can be invoked in the scheme ECSSR as part of signature generation. ECSP-NR2 corresponds to the last three steps of ECSP-NR.

Except for having EC domain parameters as input rather than DL domain parameters, the primitive is identical to DLSP-NR.

Input:

- The EC domain parameters q, a, b, r , and G associated with the key s
- The signer's private key s
- The randomizer, which is an integer u such that $1 \leq u < r$
- The pre-signature, which is an integer i such that $1 \leq i < q$
- The message representative, which is an integer f such that $0 \leq f < r$

Assumptions: Private key s and EC domain parameters q, a, b, r , and G are valid and associated with each other; $1 \leq u < r$; $1 \leq i < q$; $0 \leq f < r$.

Output: The signature, which is a pair of integers (c, d) such that $0 \leq c < r$ and $0 \leq d < r$ or “error”

Operation: The signature (c, d) shall be computed by the following or an equivalent sequence of steps:

- 1) Compute an integer $c = i + f \bmod r$. If $c = 0$, then output “error.” (This is an extremely rare event for recommended values of r —see Note 2.)
- 2) Compute an integer $d = u - sc \bmod r$.
- 3) Output the pair (c, d) as the signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of EC domain parameters q, a, b, r , and G
- At least one valid private key s for each set of domain parameters
- All randomizers u in the range $[1, r - 1]$
- All pre-signatures i in the range $[1, q - 1]$
- All message representatives f in the range $[0, r - 1]$

NOTES

1—Following the discussion for ECPSP-NR2/PV in this subclause, for security reasons, the randomizer u should be securely deleted after step 2).

2—The probability that the primitive will output “error” is about $1/r$ (i.e., essentially 0 for recommended values of r), assuming the pre-signature i is generated at random by ECPSP-NR2/PV. Thus, the check at the end of step 1) is included for completeness only; it is unlikely to occur in practice. The motivation for the check is to ensure that the signature depends on the signer’s private key s ; otherwise the signature could be verified with anyone’s public key with the same domain parameters. [An “error” output in any case does not disclose any information about the private key s , because the private key is not involved in the pre-signature primitive or prior to step 2).]

7.2.11 ECVP-NR2

ECVP-NR is the Elliptic Curve Verification Primitive, Nyberg–Rueppel version 2. It is based on the work of ISO/IEC 9796-3:2000 [B79], Koblitz [B94], Miller [B117], and Nyberg and Rueppel [B120]. This primitive recovers the message representative that was signed with ECSP-NR2, given only the signature and public key of the signer, and recovers the pre-signature. It can be invoked in the scheme ECSSR as part of signature verification and message recovery. ECVP-NR2 is the same as ECVP-NR, except that the recovered pre-signature is output in addition to the message representative.

Input:

- The EC domain parameters q, a, b, r , and G associated with the key W
- The signer’s public key W
- The signature to be verified, which is a pair of integers (c, d)

Assumptions: Public key W and EC domain parameters q, a, b, r , and G are valid and associated with each other.

Output:

- The message representative, which is an integer f such that $0 \leq f < r$; or “invalid”
- The pre-signature, which is an integer i such that $1 \leq i < q$

Operation: The message representative f and the pre-signature i shall be computed by the following or an equivalent sequence of steps:

- 1) If c is not in $[1, r - 1]$ or d is not in $[0, r - 1]$, output “invalid” and stop.
- 2) Compute an elliptic curve point $P = dG + cW$. If $P = O$, output “invalid” and stop. Otherwise, $P = (x_P, y_P)$.
- 3) Convert the field element x_P to an integer i with primitive FE2IP.
- 4) Compute an integer $f = c - i \bmod r$.
- 5) Output f as the message representative and i as the pre-signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of EC domain parameters q, a, b, r , and G
- At least one valid public key W for each set of domain parameters
- All purported signatures (c, d) that can be input to the implementation; this should at least include all (c, d) such that c and d are in the range $[0, r - 1]$

7.2.12 ECSP-PV

ECSP-PV is the Elliptic Curve Signature Primitive, Pintsov–Vanstone version. It is based on the work of Koblitz [B94], Miller [B117], and Pintsov and Vanstone [PV99]. The primitive generates a signature part on a randomized hash value with the private key of the signer, given a randomizer generated by ECPSP-NR2/PV, in such a way that the pre-signature generated by ECPSP-NR2/PV can be recovered from the signature using the public key of the signer by the ECVP-PV primitive. It can be invoked in the scheme ECSSR-PV as part of signature generation.

Except for having EC domain parameters as input rather than DL domain parameters, the primitive is identical to DLSP-PV.

Input:

- The EC domain parameters q, a, b, r , and G associated with the key s
- The signer’s private key s
- The randomizer, which is an integer u such that $1 \leq f < r$
- The randomized hash value, which is a nonnegative integer h

Assumptions: Private key s and EC domain parameters q, a, b, r , and G are valid and associated with each other; $0 \leq f < r$; $h \geq 0$.

Output: A signature part, which is an integer d , where $0 \leq d < r$

Operation: The signature part d shall be computed by the following or an equivalent sequence of steps:

- 1) Compute an integer $d = u - sh \bmod r$.
- 2) Output the pair d as the signature part.

Conformance region recommendation: A conformance region should include:

- At least one valid set of EC domain parameters q, a, b, r , and G
- At least one valid private key s for each set of domain parameters
- All randomizers u in the range $[1, r - 1]$
- All randomized hash values h in the range $[0, r - 1]$

NOTES

1—For security reasons, the randomizer u should be securely deleted after step 2).

2—This primitive does not check whether $h = 0$ to ensure that the signature depends on the signer's public key, because it is such an unlikely occurrence in the scheme in which ECSP-PV is employed, but an implementation could do if desired.

7.2.13 ECVP-PV

ECVP-PV is the Elliptic Curve Verification Primitive, Pintsov–Vanstone version. It is based on the work of Koblitz [B94], Miller [B117], and Pintsov and Vanstone [PV99]. This primitive recovers a pre-signature from a signature part generated with ECSP-PV, given only the randomized hash value and the public key of the signer. It can be invoked in the scheme ECSSR-PV as part of signature verification and message recovery.

Input:

- The EC domain parameters q, a, b, r , and G associated with the key W
- The signer's public key W
- The randomized hash value, which is an integer $h \geq 0$
- The signature part from which the pre-signature is to be recovered, which is an integer d

Assumptions: Public key W and EC domain parameters q, a, b, r , and G are valid and associated with each other.

Output: The pre-signature, which is an integer i such that $1 \leq i < q$

Operation: The pre-signature i shall be computed by the following or an equivalent sequence of steps:

- 1) If d is not in $[0, r - 1]$, output “invalid” and stop.
- 2) Compute an elliptic curve point $P = dG + hW$. If $P = O$, output “invalid” and stop. Otherwise, $P = (x_P, y_P)$.
- 3) Convert the field element x_P to an integer i with primitive FE2IP.
- 4) Output i as the pre-signature.

Conformance region recommendation: A conformance region should include:

- At least one valid set of EC domain parameters q, a, b, r , and G
- At least one valid public key W for each set of domain parameters
- All randomized hash values $h \geq 0$
- All signature parts d that can be input to the implementation; this should at least include all d such that d is in the range $[0, r - 1]$

8. Primitives based on the integer factorization problem

Replace the second paragraph with the following text:

There are four types of primitives in this family. Although they are all related to the integer factorization problem, they are based on different keys and operations. One type is known as RSA, for “Rivest-Shamir-Adleman” (see Rivest et al. [B129]); a second is known as RW, for “Rabin-Williams” (see Rabin [B128] and Williams [B149]). The third is known as OU, for “Okamoto-Uchiyama” (see Okamoto and Uchiyama [OU98]), and the fourth is known as ESIGN (see Fujisaki and Okamoto [FO98] and Okamoto [Oka90]).

8.1.1 Notation***Replace the following entries in the list of notation:***

- p, q The prime factors of the modulus n
 e The public exponent, part of a public key (e is odd for RSA; e is even for RW; $e \geq 8$ for ESIGN)

Insert the following entries:

- k The length in bits of the primes for OU and ESIGN
 u Part of the OU public key
 u_p The value $\exp(u, p-1) \bmod p^2$, used in OU
 r Randomized hash value, an integer
 v Part of the OU public key
 $L(x)$ The function $(x-1)/p$, used in OU
 (n, u, v, k) An OU public key
 (p, q, k, w) An OU private key
 (n, e, k) An ESIGN public key
 (p, q, k, e) An ESIGN private key
 C Encrypted message, an octet string, computed by an encryption scheme
 M_1 Recoverable message part, an octet string, the part of a message that can be recovered from the signature in a signature scheme giving message recovery
 M_2 Nonrecoverable message part (if any), an octet string, the part of a message that cannot be recovered from the signature in a signature scheme giving message recovery

Insert the following two subclauses (8.1.3.3 and 8.1.3.4) after 8.1.3.2:**8.1.3.3 OU key pairs**

An OU *public key* consists of a *modulus* n , which is of the form p^2q for two odd positive prime integers p and q , where the length of each of p and q is k bits for some integer k , a pair of integers (u, v) ($1 < u < n$, $1 < v < n$) such that the order of $u_p = \exp(u, p-1) \bmod p^2$ is p and where $v = v_0^n \bmod n$ for some integer v_0 ($1 < v_0 < n$). Define the function $L(x)$ as

$$L(x) = (x-1)/p$$

and let $w = L(u_p)$. The corresponding OU private key consists of the primes p and q , the length parameter k , and the value w .

NOTE—The value v_0 need not be kept secret. A typical selection of the value v_0 is u . See D.5.3.2.1 for further discussion.

8.1.3.4 ESIGN key pairs

An ESIGN *public key* consists of a *modulus* n , which is of the form p^2q for two odd positive prime integers p and q where the length of each of p and q is k bits for some integer k and the length of n is $3k$ bits, a *public exponent* e ($8 \leq e < n$) such that $\text{GCD}(e, n) = 1$, and the length parameter k . The corresponding ESIGN *private key* consists of the pair of primes (p, q) , the length parameter k , and the exponent e .

Replace the existing 8.1.3.3 with the following text (note the renumbering):

8.1.3.5 Considerations common to IF key pairs

An IF key pair may or may not be generated by the party using it, depending on the trust model. This and other security issues related to IF key pair generation are discussed in D.3.1 and D.4.3. A suggested method for generating RSA key pairs is contained in A.16.11. A suggested method for generating RW key pairs is contained in A.16.12.

Parties establish IF keys as part of key management for a scheme. Depending on the key management technique, it is possible that an established key does not satisfy the intended definition for the key, even though it has the same general form (e.g., components n and e). To accommodate this possibility, the terms *RSA public key*, *RW public key*, *OU public key*, *ESIGN public key*, and similarly for private keys shall be understood in this standard as referring to instances of the general form for the key. The terms *valid RSA public key*, *valid RW public key*, *valid OU public key*, *valid ESIGN public key*, *valid RSA key pair*, *valid RW key pair*, and so on shall be reserved for keys satisfying the definitions.

Key validation is the process of determining whether a key is valid. Further discussion of key validation is contained in D.3.3, although no algorithm for IF public key validation is suggested in Annex A. No algorithm is given for IF private key validation, because, generally, a party controls its own private key and need not validate it. However, private key validation may be performed if desired.

For security, the primes p and q need to be generated so that they are unpredictable and inaccessible to an adversary. Whatever form a private key is represented in, it needs to be stored so that it is inaccessible to an adversary. The compromise of the primes or other private key components (that are not already part of the public key) may aid the adversary in recovering the private key. These values should be securely deleted if not used.

Three representations are provided for RSA and RW private keys because of performance tradeoffs between the representations. Use of the quintuple representation tends to result in faster performance for larger sizes of n .

8.2 Primitives

Replace the third paragraph with the following text:

The primitives described in this subclause can be combined into six pairs: message representatives encrypted with IFEP-RSA can be decrypted with IFDP-RSA; message representatives signed with IFSP-RSA1, IFSP-RSA2, or IFSP-RW can be recovered with IFVP-RSA1, IFVP-RSA2, or IFVP-RW, respectively; message representatives encrypted with IFEP-OU can be decrypted with IFDP-OU; and message representatives signed with IFSP-ESIGN can be verified with IFVP-ESIGN. Although IFEP-RSA is mathematically identical to IFVP-RSA1, and IFDP-RSA is mathematically identical to IFSP-RSA1, these primitives are used for entirely different purposes and should not be confused. The -RSA1, -RSA2, and -RW signature primitives are similar, but they have some important differences (see C.3.6). To aid in defining these three primitives, the operation “IF Private Key Operation” is defined first.

8.2.4 IFSP-RSA1

Replace the first paragraph with the following text:

IFSP-RSA1 is the RSA Signature Primitive, version 1. It is based on the work of Rivest et al. [B129]. It can be invoked in a scheme to compute a signature on a message representative with the private key of the signer, in such a way that the message representative can be recovered from the signature using the public key of the signer by the IFVP-RSA1 primitive. IFSP-RSA1 can be invoked in the schemes IFSSA and IFSSR as part of signature generation.

8.2.5 IFVP-RSA1

Replace the first paragraph with the following text:

IFVP-RSA1 is the RSA Verification Primitive, version 1. It is based on the work of Rivest et al. [B129]. This primitive recovers the message representative that was signed with IFSP-RSA1, given only the signature and public key of the signer. It can be invoked in a scheme as part of signature verification and, possibly, message recovery. IFVP-RSA1 can be invoked in the schemes IFSSA and IFSSR as part of signature verification.

8.2.6 IFSP-RSA2

Replace the first paragraph with the following text:

IFSP-RSA2 is the RSA Signature Primitive, version 2. It is based on the work of ISO/IEC 9796:1991 [B78] and Rivest et al. [B129]. Its output is at least 1 bit shorter than the RSA modulus n . It can be invoked in a scheme to compute a signature on a message representative of a certain form with the private key of the signer, in such a way that the message representative can be recovered from the signature using the public key of the signer by the IFVP-RSA2 primitive. IFSP-RSA2 can be invoked in the schemes IFSSA and IFSSR as part of signature generation.

8.2.7 IFVP-RSA2

Replace the first paragraph with the following text:

IFVP-RSA2 is the RSA Verification Primitive, version 2. It is based on the work of ISO/IEC 9796:1991 [B78] and Rivest et al. [B129]. This primitive recovers the message representative that was signed with IFSP-RSA2, given only the signature, the public key of the signer, and the last four bits of the message representative. It can be invoked in a scheme as part of signature verification and, possibly, message recovery. IFVP-RSA2 can be invoked in the schemes IFSSA and IFSSR as part of signature verification.

8.2.8 IFSP-RW

Replace the first paragraph with the following text:

IFSP-RW is the RW Signature Primitive. It is based on the work of ISO/IEC 9796:1991 [B78], Rabin [B128], and Williams [B149]. Its output is at least 1 bit shorter than the RW modulus n . It can be invoked in a scheme to compute a signature on a message representative with the private key of the signer, in such a way that the message representative can be recovered from the signature using the public key of the signer by the IFVP-RW primitive. IFSP-RW can be invoked in the schemes IFSSA and IFSSR as part of signature generation.

8.2.9 IFVP-RW

Replace the first paragraph with the following text:

IFVP-RW is the RW Verification Primitive. It is based on the work of ISO/IEC 9796:1991 [B78], Rabin [B128], and Williams [B149]. This primitive recovers the message representative that was signed with IFSP-RW, given only the signature, public key of the signer, and the last four bits of the message representative. It can be invoked in a scheme as part of signature verification and, possibly, message recovery. IFVP-RW can be invoked in the schemes IFSSA and IFSSR as part of signature verification.

Insert the following four subclauses (8.2.10–8.2.13) after 8.2.9:

8.2.10 IFEP-OU

IFEP-OU is the OU Encryption Primitive. It is based on the work of Okamoto and Uchiyama [OU98]. It is invoked in the scheme IFES-EPOC as part of encrypting a message, given the message and the public key of the intended recipient, and it is also used to verify that decryption was successful. The message can be decrypted within a scheme by invoking IFDP-OU.

Input:

- The recipient's OU public key (n, u, v, k)
- The message representative, which is an integer f such that $0 \leq f < 2^{k-1}$
- The randomized hash value, which is an integer r such that $0 \leq r < n$

Assumptions: Public key (n, u, v, k) is valid; $0 \leq f < 2^{k-1}$; $0 \leq r < n$.

Output: The encrypted message representative, which is an integer g such that $0 \leq g < n$

Operation: The encrypted message representative g shall be computed by the following or an equivalent sequence of steps:

- 1) Let $g = \exp(u, f) \times \exp(v, r) \bmod n$.
- 2) Output g as the encrypted message representative.

Conformance region recommendation: A conformance region should include:

- At least one valid OU public key (n, u, v, k)
- All message representatives f in the range $[0, 2^{k-1} - 1]$

NOTE—The randomized hash value r should be different for encryptions of different message representatives f , as its reuse can lead to the recovery of the partial or total information about the message representative f .

8.2.11 IFDP-OU

IFDP-OU is the OU Decryption Primitive. It is based on the work of Okamoto and Uchiyama [OU98]. It is used in the scheme IFES-EPOC as part of decrypting a message encrypted with the use of IFEP-OU, given the encrypted message representative and the private key of the recipient.

Input:

- The recipient's OU private key (p, q, k, w)
- The encrypted message representative, which is an integer g such that $0 \leq g < n$, where n is from the corresponding public key

Assumptions: Private key (p, q, k, w) is valid; $0 \leq g < n$.

Output: The message representative, which is an integer f such that $0 \leq f < 2^{k-1}$; or "invalid"

Operation: The message representative f shall be computed by the following or an equivalent sequence of steps:

- 1) Compute $g_p = \exp(g, p-1) \bmod p^2$.
- 2) Compute $L(g_p) = (g_p - 1)/p$.
- 3) Compute $f = L(g_p)/w \bmod p$.
- 4) Check whether $0 \leq f < 2^{k-1}$.
- 5) If it holds, output f . Otherwise, output "invalid."

Conformance region recommendation: A conformance region should include:

- At least one valid OU private key (p, q, k, w)
- All encrypted message representatives g in the range $[0, n-1]$

8.2.12 IFSP-ESIGN

IFSP-ESIGN is ESIGN Signature Primitive. It is based on the work of Fujisaki and Okamoto [FO98] and Okamoto [Oka90]. It can be invoked in a scheme to compute a signature on a message representative with the private key of the signer, in such a way that the message representative can be recovered from the signature using the public key of the signer by the IFVP-ESIGN primitive. It is intended for a signature scheme with appendix and can be invoked in the scheme IFSSA as part of signature generation.

Input:

- The signer's ESIGN private key (p, q, k, e)
- The message representative, which is an integer f such that $0 \leq f < 2^{k-1}$

Assumptions: Private key (p, q, k, e) is valid; $0 \leq f < 2^{k-1}$.

Output: The signature, which is an integer s such that $0 \leq s < n$, where n is from the corresponding public key

Operation: The signature s shall be computed by the following or an equivalent sequence of steps:

- 1) Generate a random integer r , $0 \leq r < pq$, such that $\text{GCD}(r, n) = 1$.
- 2) Compute an integer $z = f \times 2^{2k}$.
- 3) Compute an integer $a = (z - \exp(r, e)) \bmod n$.
- 4) Compute an integer $w_0 = \lceil a/pq \rceil$.
- 5) Compute an integer $w_1 = w_0 \times pq - a$.
- 6) If $w_1 \geq 2^{2k-1}$, then go to step 1).
- 7) Compute an integer $t = w_0 / (e \times \exp(r, e-1)) \bmod p$.
- 8) Compute an integer $s = r + tpq \bmod n$.
- 9) Output s as the signature.

Conformance region recommendation: A conformance region should include:

- At least one valid ESIGN private key (p, q, k, e)
- All message representatives f in the range $[0, 2^{k-1} - 1]$

NOTES

1—The integer r in step 1) should be a one-time value that is generated by the signer following the security recommendations of D.6 and D.7. A new integer r should be generated for every signature, as its reuse can lead to the recovery of the private key. The integer r and other intermediate values should be securely deleted after they are used as their recovery by an opponent may lead to the recovery of the private key.

2—The integer r in step 1), the value $\exp(r, e) \bmod n$ in step 3), and the value $1/(e \times \exp(r, e - 1)) \bmod p$ in step 7) may be precomputed in advance of the availability of the message representative, although such precomputation will only be useful for a fraction of the message representatives if the optional test in step 6) is included.

3—Note that the length in bits of the modulus n must be a multiple of 3 (as is defined in 8.1.3.4) in order for the signature and verification primitives to operate correctly. Otherwise, it is possible that some signatures generated by IFSP-ESIGN may be rejected by IFVP-ESIGN, as noted by Shipsey [Shi01].

8.2.13 IFVP-ESIGN

IFVP-ESIGN is ESIGN Verification Primitive. It is based on the work of Fujisaki and Okamoto [FO98] and Okamoto [Oka90]. This primitive recovers the message representative that was signed with IFSP-ESIGN, given only the signature and public key of the signer. It is intended for a signature scheme with appendix and can be invoked in the scheme IFSSA as part of signature verification.

Input:

- The signer's ESIGN public key (n, e, k)
- The signature to be verified, which is an integer s

Assumptions: Public key (n, e, k) is valid.

Output: The message representative, which is an integer f such that $f < 2^{k-1}$; or “invalid”

Operation: The message representative f shall be computed by the following or an equivalent sequence of steps:

- 1) If s is not in $[0, n - 1]$, output “invalid” and stop.
- 2) Let $f = \lfloor (\exp(s, e) \bmod n) / 2^{2k} \rfloor$.
- 3) If f is not in $[0, 2^{k-1} - 1]$, output “invalid” and stop.
- 4) Output f .

Conformance region recommendation: A conformance region should include:

- At least one valid ESIGN public key (n, e, k)
- All purported signatures s that can be input to the implementation; this should at least include all s in the range $[0, n - 1]$

9. Key agreement schemes

9.2 DL/ECKAS-DH1

Insert the following sentence after the first paragraph:

Figure 2 illustrates the scheme.

Insert the following figure:

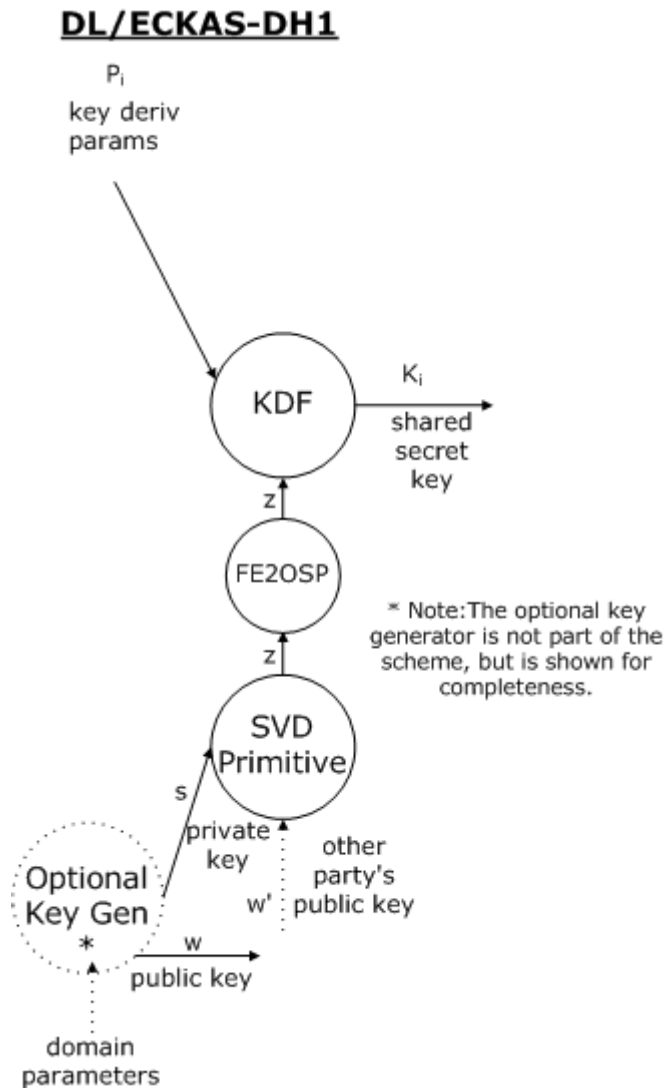


Figure 2—DL/ECKAS-DH1 key agreement operation

9.2.1 Scheme options

Replace the list of options with the following text:

- a) A secret value derivation primitive, which shall be DLSVDP-DH, DLSVDP-DHC, ECSVDP-DH, or ECSVDP-DHC
- b) For a -DHC secret value derivation primitive, an indication as to whether or not compatibility with the corresponding -DH primitive is desired
- c) A key derivation function, which should be KDF1 (see 13.1), KDF2 (see 13.2), or a function designated for use with DL/ECKAS-DH1 in an amendment to this standard

9.3 DL/ECKAS-DH2

Insert the following sentence after the first paragraph:

Figure 3 illustrates the scheme.

Insert the following figure:

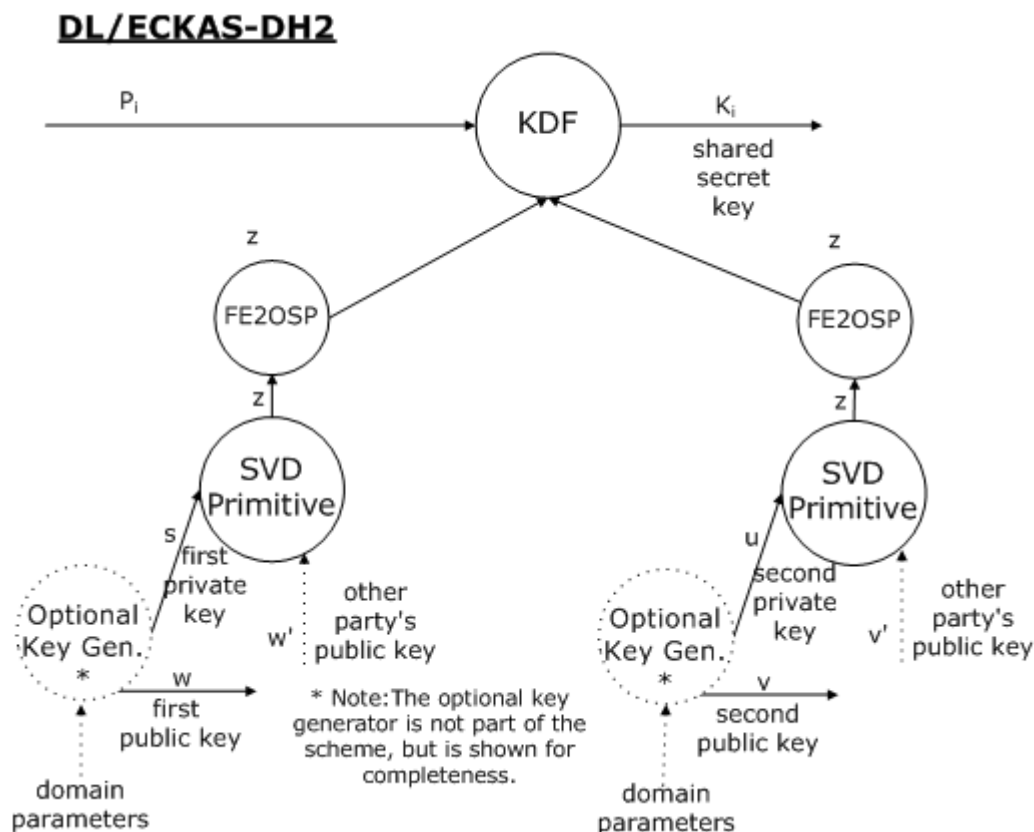


Figure 3—DL/ECKAS-DH2 key agreement operation

9.3.1 Scheme options

Replace the list of options with the following text:

- Two secret value derivation primitives, each of which (independently) shall be DLSVDP-DH, DLSVDP-DHC, ECSVDP-DH, or ECSVDP-DHC
- For each -DHC secret value derivation primitive, an indication as to whether or not compatibility with the corresponding -DH primitive is desired
- If the two secret value derivation primitives are the same (and have the same compatibility option in the -DHC case), an indication whether compatibility with DL/ECKAS-DH1 is desired
- A key derivation function, which should be KDF1 (see 13.1), KDF2 (see 13.2), or a function designated for use with DL/ECKAS-DH2 in an amendment to this standard

9.4 DL/ECKAS-MQV

Insert the following sentence after the first paragraph:

Figure 4 illustrates the scheme.

Insert the following figure:

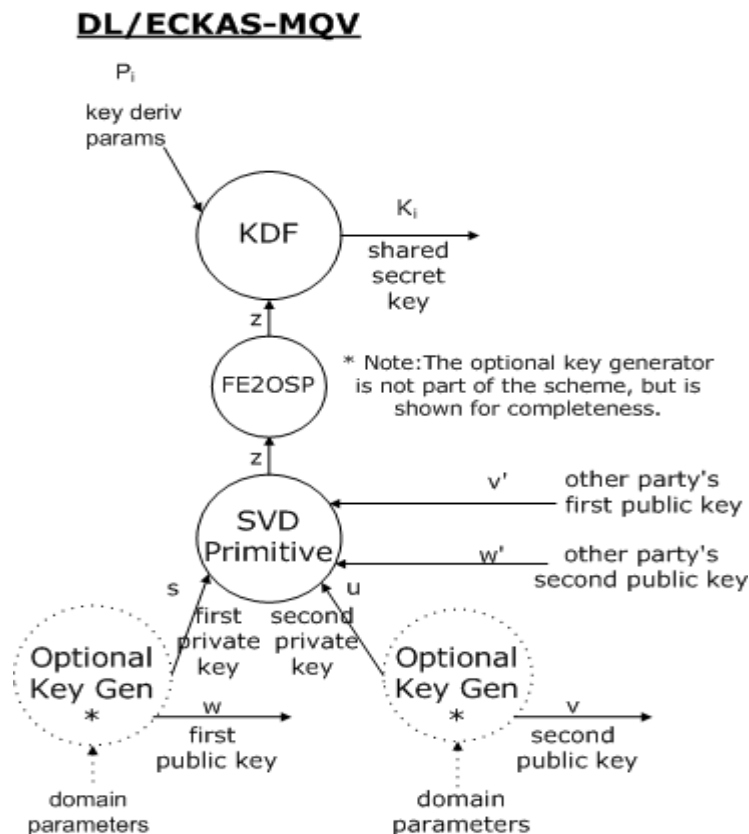


Figure 4—DL/ECKAS-DH2 key agreement operation

9.4.1 Scheme options

Replace the list of options with the following text:

- a) A secret value derivation primitive, which shall be DLSVDP-MQV, DLSVDP-MQVC, ECSVDP-MQV, or ECSVDP-MQVC
- b) For an -MQVC secret value derivation primitive, an indication as to whether or not compatibility with the corresponding -MQV primitive is desired
- c) A key derivation function, which should be KDF1 (see 13.1), KDF2 (see 13.2), or a function designated for use with DL/ECKAS-MQV in an amendment to this standard

9.4.2 Key agreement options

Change step 5) to the following:

- 5) Compute a shared secret value z from the selected private keys s , ~~and u~~ the selected key pair (u, v) , and the other party's two public keys w' and v' with the selected secret value derivation primitive (see 9.4.1).

10. Signature schemes

Replace the first paragraph with the following text:

The general model for signature schemes is given in 10.1. Five specific schemes and their allowable options are given in 10.2, 10.3, 10.4, 10.5, and 10.6.

10.1 General model

Replace the second paragraph with the following text:

There are two types of signature scheme. In a *signature scheme with appendix*, the signer conveys both the message and the signature to the verifier, who verifies their consistency. In a *signature scheme giving message recovery*, a message is processed in two parts: a *recoverable message part* that can be recovered from the signature and a *nonrecoverable message part*. The signer conveys the signature and the nonrecoverable message part to the verifier, who verifies their consistency and recovers the recoverable part. Such a scheme is said to provide *total message recovery* if the nonrecoverable part is empty and *partial message recovery* otherwise.

A signature scheme giving message recovery will potentially be more efficient in terms of the message expansion for signing a message than a signature scheme with appendix. A signature scheme giving message recovery is typically considered for shorter messages, for which the message expansion is relatively a larger overhead. In a signature scheme with appendix, all of the message will be handled the same way, which may be more convenient in terms of implementation and integration with other operations, such as multiple signatures on the same message.

Replace the third and fourth paragraphs (including their lists) with the following text:

A signature generation operation in a signature scheme with appendix has the following form:

- 1) Select a valid private key (together with its set of domain parameters, if any) for the operation.
- 2) Apply certain cryptographic operations to the message and the private key to produce a signature.
- 3) Output the signature.

A signature verification operation in a signature scheme with appendix has the following form:

- 1) Obtain the signer's purported public key (together with its set of domain parameters, if any) for the operation (Note 1 below).
- 2) *(Optional)* Validate the public key and its associated set of domain parameters, if any. If validation fails, output "invalid" and stop (Note 2 below).
- 3) Apply certain cryptographic operations to the message, the signature, and the public key to verify the signature.
- 4) Output "valid" or "invalid" according to the result of step 3).

A signature generation operation in a signature scheme giving message recovery has the following form:

- 1) Select a valid private key (together with its set of domain parameters, if any) for the operation.
- 2) Apply certain cryptographic operations to the recoverable message part, the nonrecoverable part (if any), and the private key to produce a signature.
- 3) Output the signature.

A signature verification operation in a signature scheme giving message recovery has the following form:

- 1) Obtain the signer's purported public key (together with its set of domain parameters, if any) for the operation (Note 1 below).
- 2) *(Optional)* Validate the public key and its associated set of domain parameters, if any. If validation fails, output "invalid" and stop (Note 2 below).
- 3) Apply certain cryptographic operations to the nonrecoverable message part (if any), the signature, and the public key to verify the signature and recover the recoverable part of the message.
- 4) Output the recoverable part of the message or "invalid" according to the result of step 3).

Replace Note 3 with the following text:

3—*(Error conditions)* The signer's and verifier's steps may produce errors under certain conditions, such as the following:

- Private key not found in signer's step 1)
- Message, recoverable message part, non-recoverable message part, or private key not supported in signer's step 2)
- Public key not found in verifier's step 1)
- Public key not valid in verifier's step 2)
- Message, non-recoverable message part, signature, or public key not supported in verifier's step 3)

Such error conditions should be detected and handled appropriately by an implementation, but the specific methods for detecting and handling them are outside of the scope of this standard.

10.2 DL/ECSSA

Insert the following sentence after the first paragraph:

Figure 5 illustrates the scheme.

Insert the following figure:

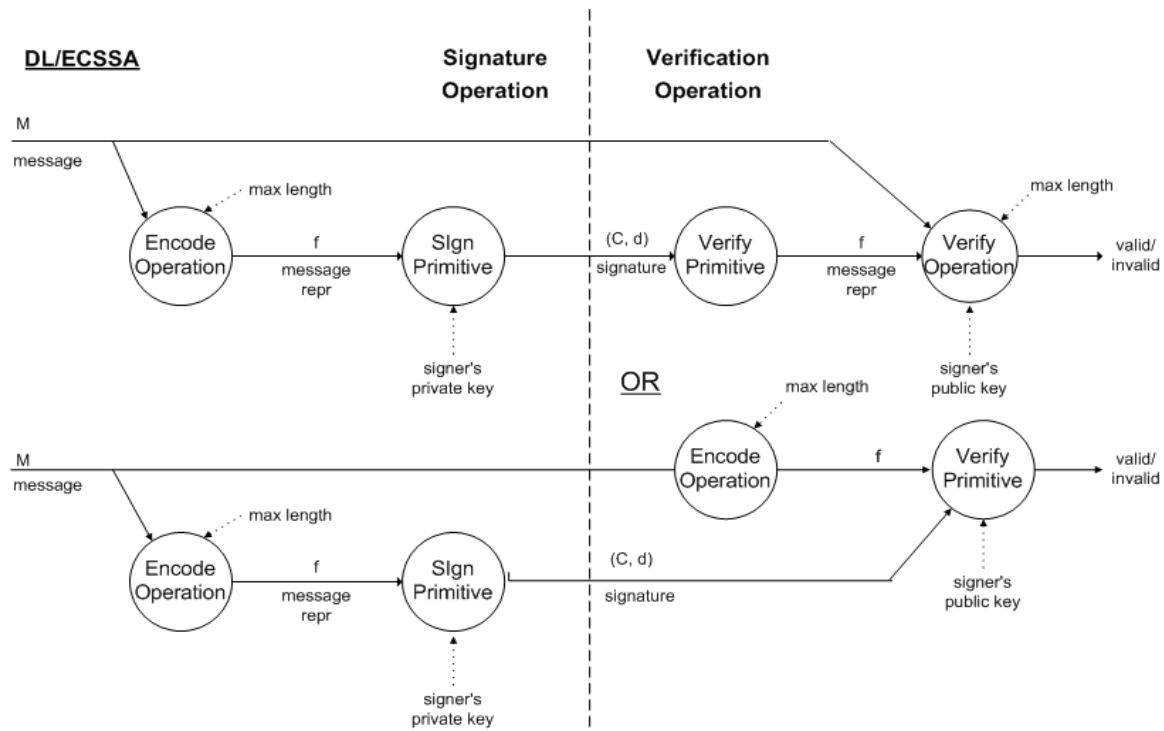


Figure 5—DL/ECSSA signature generation and verification operations

10.2.3 Signature verification operation

Change step 4b) to the following:

- b Use the ~~decoding~~-encoding operation of the selected message-encoding method (see 10.2.1) to produce a message representative f of maximum length l from the message M (f will be a non-negative integer).

10.3 IFSSA

Insert the following sentence after the first paragraph:

Figure 6 illustrates the scheme.

Insert the following figure:

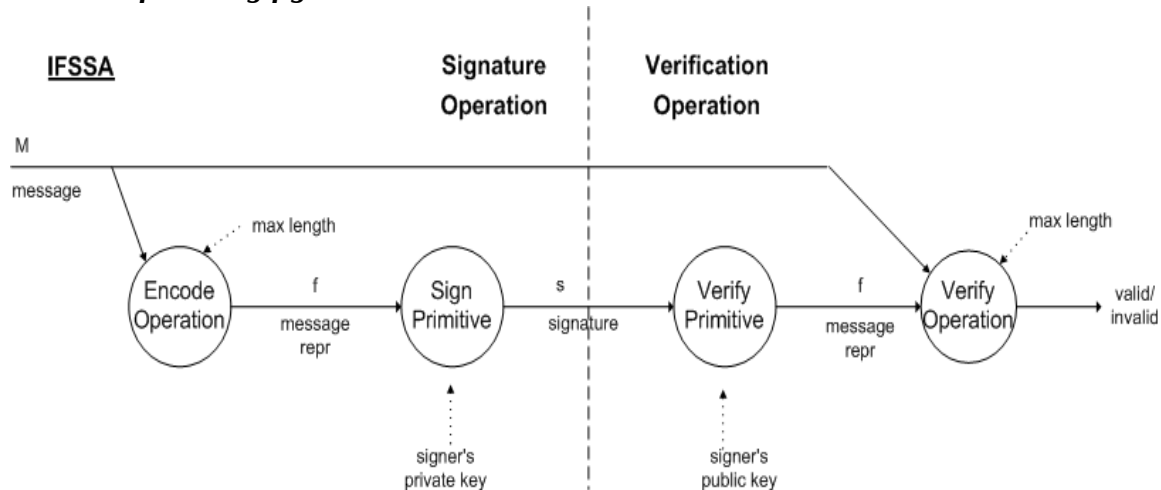


Figure 6—IFSSA signature generation and verification operations

10.3.1 Scheme options

Replace the list of options with the following text:

- a) The type of key pair for the signer (RSA, RW, or ESIGN)
- b) The signature and verification primitives, which shall be a pair from the following list:
 - 1) If the signer's public key is an RSA public key, then the primitives shall be either the pair IFSP-RSA1 and IFVP-RSA1 or the pair IFSP-RSA2 and IFVP-RSA2
 - 2) If the signer's public key is an RW public key, then the primitives shall be the pair IFSP-RW and IFVP-RW.
 - 3) If the signer's public key is an ESIGN public key, then the primitives shall be the pair IFSP-ESIGN and IFVP-ESIGN.
- c) The message-encoding method for signatures with appendix, which should be one of the following:
 - 1) If the signature primitive is IFSP-RSA1, then the message-encoding method should be EMSA2 (see 12.1.2), EMSA3 (see 12.1.3), EMSA4 (see 12.1.4), or a technique designated for use with IFSSA and this signature primitive in an amendment to this standard.
 - 2) If the signature primitive is IFSP-RSA2 or IFSP-RW, then the message-encoding method should be EMSA2, EMSA4, or a technique designated for use with IFSSA and this signature primitive in an amendment to this standard (where the message-encoding method shall always produce a message representative congruent to 12 modulo 16).
 - 3) If the signature primitive is IFSP-ESIGN, then the message-encoding method should be EMSA5 or a technique designated for use with IFSSA and this signature primitive in an amendment to this standard.

NOTE—EMSA2 requires that the message representative length, i.e., (length of n in bits) – 1 for RSA or RW, be congruent to 7 mod 8.

Replace 10.3.2 with the following text:

10.3.2 Signature generation option

A signature s shall be generated by a signer from a message M by the following or an equivalent sequence of steps:

- 1) Select a valid private key K for the operation.
- 2) If the selected signature primitive is IFSP-RSA1, IFSP-RSA2, or IFSP-RW, then set the maximum length of the message representative to be $l = (\text{length of } n \text{ in bits}) - 1$, where n is the modulus in the RSA or RW private key. If the selected signature primitive is IFSP-ESIGN, then set the maximum length to be $l = k - 1$, where k is from the ESIGN private key.
- 3) Use the encoding operation of the selected message-encoding method (see 10.3.1) to produce a message representative f of maximum length l from the message M (f will be a non-negative integer). If the selected encoding method is EMSA2 or EMSA4, f will be congruent to 12 modulo 16.
- 4) Apply the selected signature primitive (see 10.3.1) to the integer f and the selected private key K to generate the signature s .
- 5) Output the signature.

Conformance region recommendation: A conformance region should include:

- At least one valid private key K
- A range of messages M

Replace 10.3.3 with the following text:

10.3.3 Signature verification option

A signature s on a message M shall be verified by a verifier by the following or an equivalent sequence of steps:

- 1) Obtain the signer's purported public key.
- 2) *(Optional)* Validate the public key. Output "invalid" and stop if validation fails.
- 3) Apply the selected verification primitive (see 10.3.1) to the signature s and the signer's public key to recover an integer f . If the output of the primitive is "invalid," output "invalid" and stop.
- 4) If the selected verification primitive is IFVP-RSA1, IFVP-RSA2, or IFVP-RW, then set the maximum length of the message representative to be $l = (\text{length of } n \text{ in bits}) - 1$, where n is the modulus in the RSA or RW public key. If the selected verification primitive is IFVP-ESIGN, then set the maximum length to be $l = k - 1$, where k is from the ESIGN public key.
- 5) Verify that the integer f is a correct encoded representative of the message M using the verification operation of the selected message-encoding method (see 10.3.1) and maximum length l . If that is the case, output "valid." Else, output "invalid."

Conformance region recommendation: A conformance region should include:

- At least one valid public key; if key validation is performed, invalid public keys that are appropriately rejected by the implementation may also be included in the conformance region.
- All messages M that can be input to the implementation.
- All purported signatures s that can be input to the implementation; this should at least include all s in the range $[0, n - 1]$ for IFVP-RSA1 and IFVP-ESIGN or $[0, (n - 1)/2]$ for IFVP-RSA2 and IFVP-RW.

NOTES

1—The upper bound on the length in bits of the message representative is so that the message representative is less than the modulus n , as required by the RSA and RW signature primitives, or so that it is less than $2^k - 1$, as required by the ESIGN signature primitive. As message-encoding methods for signatures with appendix are usually based on hash functions, the length of a message that can be signed by this scheme is usually unrestricted or constrained by a very large number.

2—This scheme, with appropriate restrictions on the scheme options and inputs, may be compatible with techniques in ANSI X9.31-1998 [B7], ISO/IEC 14888-3:1998 [B80], and PKCS #1 v2.1 [PKCS1v2_1].

Insert the following three subclauses (10.4–10.6) after 10.3:

10.4 DL/ECSSR

DL/ECSSR is Discrete Logarithm and Elliptic Curve Signature Scheme with Recovery.

Figure 7 illustrates the scheme.

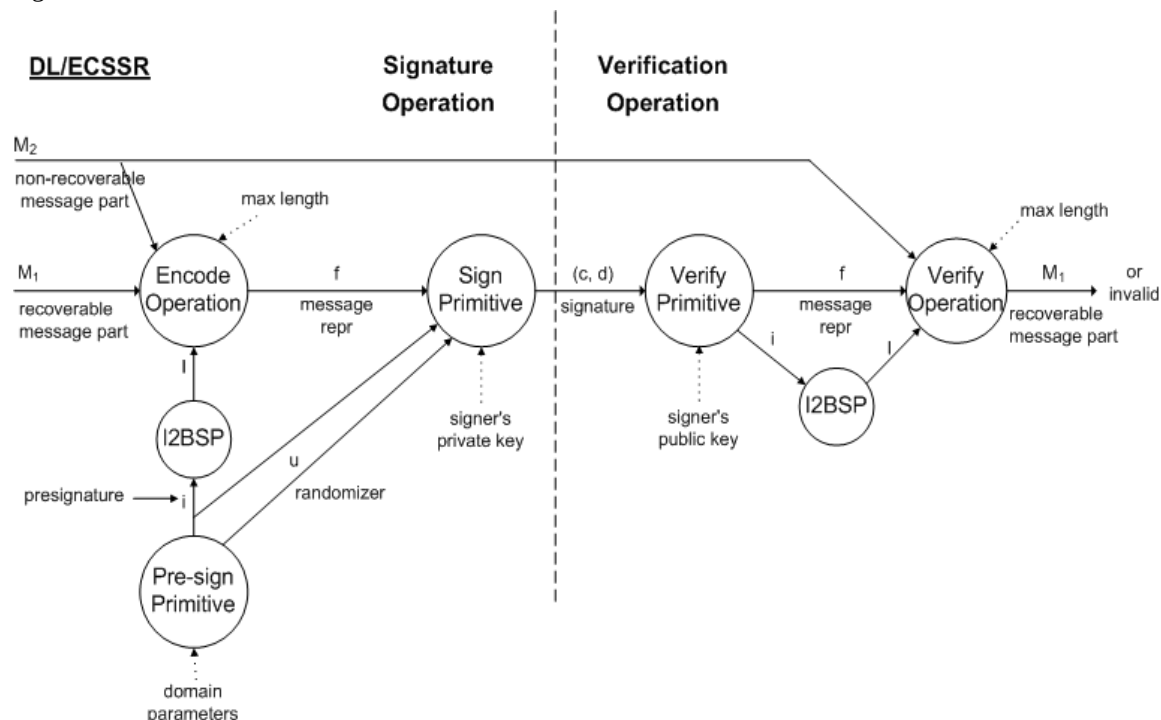


Figure 7—DL/ECSSR signature generation and verification operations