



IEEE Standard Specifications for Password-Based Public-Key Cryptographic Techniques

IEEE Computer Society

Sponsored by the
Microprocessor Standards Committee

1363.2TM

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
29 January 2009

IEEE Std 1363.2TM-2008

IEEE Standard Specifications for Password-Based Public-Key Cryptographic Techniques

Sponsor

**Microprocessor Standards Committee
of the
IEEE Computer Society**

Approved 26 September 2008
IEEE-SA Standards Board

Approved 2 February 2009
American National Standards Institute

Abstract: This standard covers specifications of public-key cryptographic techniques for password-based authentication and key establishment, supplemental to the techniques described in IEEE Std 1363™-2000 and IEEE Std 1363a™-2004. It is intended as a companion standard to IEEE Std 1363-2000 and IEEE Std 1363a-2004. It includes specifications of primitives and schemes designed to utilize passwords and other low-grade secrets as a basis for securing electronic transactions, including schemes for password-authenticated key agreement and password-authenticated key retrieval.

Keywords: authentication, key agreement, password, public-key cryptography

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

Copyright © 2009 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 29 January 2009. Printed in the United States of America.

2nd Printing 12 June 2009. The following ISBNs were incorrect in the 1st Printing on 29 January 2009.

PDF: ISBN 978-0-7381-5806-8 STD95824
Print: ISBN 978-0-7381-5807-5 STDPD95824

The correct ISBNs are:

PDF: ISBN 978-0-7381-6016-0 STD95824
Print: ISBN 978-0-7381-6017-7 STDPD95824

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

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 or the soundness of any judgments 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 his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

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. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, 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 submitted to the following address:

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

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 1363.2-2008, IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques.

The history of the IEEE P1363.2^a project began in late 1996 with a presentation to the IEEE P1363 Working Group of a password-based public-key method for key agreement. Originally submitted as material for potential inclusion in the P1363 document, which later became IEEE Std 1363-2000,^b this class of technique was deemed to be sufficiently interesting and yet sufficiently different from the original focus of the IEEE P1363 Working Group as to merit further study to determine how it should be addressed. At that time, the techniques included in the P1363 draft were fairly stable; however, there were also other 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 methods that were similar in function to and in the same mathematical families as methods in the P1363 project, which focused on key exchange, digital signature, and public-key encryption schemes in the integer factorization, discrete logarithm (DL), and elliptic curve (EC) families. Work on P1363a progressed while IEEE Std 1363-2000 was solidified and prepared for the IEEE balloting process. In time, the Working Group similarly chose to close P1363a to additional techniques and prepare it for ballot, leaving standardization of additional techniques to future standards. P1363a resulted in the IEEE Std 1363a-2004 amendment.

Throughout the process of creating these standards, the Working Group continued to receive numerous submissions of creative, useful, and well-designed public-key cryptographic techniques, including password-based techniques. To address the number of submitted techniques that fell outside the functional or familial scope of both IEEE Std 1363-2000 and IEEE Std 1363a-2004, the Microprocessor Standards Committee (MSC) commissioned a study group, in which many of the IEEE P1363 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 P1363.2 (password-based public-key cryptographic techniques) and P1363.1 (public-key cryptographic techniques based on hard problems over lattices). The Working Group has also begun work on P1363.3 (identity-based cryptographic techniques using pairings) and has discussed other potential projects to continue the work from IEEE Std 1363a-2004.

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 site that will support all of the IEEE 1363 standards and current projects.^c

The IEEE P1363 Working Group would once again like to thank all of the participants and outside experts that have contributed to this standard, to the development of public-key technology and to the P1363 process.

^a The P indicates an IEEE authorized standards project that was not approved by the IEEE-SA Standards Board at the time the Working Group was formed.

^b Information on references can be found in Clause 2.

^c Information can be found at: <http://grouper.ieee.org/groups/1363/index.html>.

Notice to users

Laws and regulations

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

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association Web site at <http://ieeexplore.ieee.org/xpl/standards.jsp>, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA Web site at <http://standards.ieee.org>.

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

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the IEEE P1363 Working Group had the following membership:

William Whyte, *Chair*
Don B. Johnson, *Vice Chair*
Mike Brenner, *Primary Editor*
David Jablon, *1363.2 Project Editor*

Guido Appenzeller
Xavier Boyen
Daniel Brown
Mark Chimley

Andy Dancer
Luther Martin
Roger Schlafly
Hovav Shacham

Ari Singer
Jerry Solinas
Terence Spies
Yongge Wang

In addition, the Working Group would like to thank the following people for their contributions to the standard:

Wole Akpose
Kendall Ananyi
Mihir Bellare
Liqun Chen
Wei Dai
Warwick Ford
Eric Fung
Craig Gentry
Jim Hughes

Bill Jennings
Burt Kaliski
Pieter Kasselmann
Jonathan Katz
David Kravitz
Hugo Krawczyk
Taekyoung Kwon
Philip D. MacKenzie
James H Manger
Chris Mitchell

Rafail Ostrovsky
Zulfikar Ramzan
Phil Rogaway
Victor Shoup
Ram Swaminathan
Michael Wiener
Tom Wu
Kim-Ee Yeoh
Moti Yung

The Working Group apologizes for any inadvertent omissions from the preceding lists. Please note that inclusion of a person's name does not imply that the person agrees with all the materials in the standard.

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

Matthew Ball
Mike Brenner
Juan Carreon
Weijen Chen
Keith Chow
Tommy Cooper
Andy Dancer
James Davis
Yaacov Fenster
Michael Geipel
Randall Groves
Werner Hoelzl
Atsushi Ito

David Jablon
Raj Jain
Piotr Karocki
William Lumpkins
G. Luri
Philip D. MacKenzie
Michael Markowitz
Edward McCall
Earl Meiers
Gary Michel
Apurva Mody
Michael S. Newman
Nick S. A. Nikjoo
Vikram Punj

Robert Robinson
Fernando Lucas Rodriguez
Michael Rush
Suman Sharma
Gil Shultz
Steven Smith
Thomas Starai
Rene Struik
Gerald Stueve
Mark-Rene Uchida
William Whyte
Oren Yuen
Janusz Zalewski

When the IEEE-SA Standards Board approved this standard on 26 September 2008, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Victor Berman
Richard DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
William R. Goldbach
Arnold M. Greenspan
Kenneth S. Hanus

Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Glenn Parsons

Ronald C. Petersen
Chuck Powers
Narayanan Ramachandran
Jon Walter Rosdahl
Anne-Marie Sahazizian
Malcolm V. Thaden
Howard L. Wolfman
Don Wright

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Michael Janezic, *NIST Representative*

Lorraine Patsco
IEEE Standards Program Manager, Document Development

Malia Zaman
IEEE Standards Program Manager, Technical Program Development

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Organization of the document.....	2
1.4 Document conventions	3
2. Normative references.....	4
3. Definitions, acronyms, and abbreviations	4
3.1 Definitions	4
3.2 Acronyms and abbreviations	6
4. Types of cryptographic techniques.....	7
4.1 Brief history of the field	8
4.2 Variations of the network password problem	8
4.3 General model.....	8
4.4 Primitives.....	9
4.5 Schemes.....	12
4.6 Additional methods.....	13
4.7 Table summary	14
5. Mathematical conventions.....	15
5.1 Mathematical notation	16
5.2 Bit strings and octet strings.....	16
5.3 Data type conversion	17
6. The DL setting.....	18
6.1 The DL setting	18
7. The EC setting	19
7.1 The EC setting	19
8. Primitives.....	20
8.1 Notation and definitions	20
8.2 Primitives.....	24
9. Password-authenticated key agreement schemes.....	46
9.1 General model.....	46
9.2 BPKAS-PAK	48
9.3 BPKAS-PPK.....	51
9.4 BPKAS-SPEKE.....	54
9.5 APKAS-AMP	57
9.6 APKAS-BSPEKE2.....	60
9.7 APKAS-PAKZ	64
9.8 [DL] APKAS-SRP3, APKAS-SRP6	68
9.9 [EC] APKAS-SRP5	72
9.10 APKAS-WSPEKE.....	75
10. Password-authenticated key retrieval schemes.....	79
10.1 General model.....	79
10.2 PKRS-1	79

11. Key derivation functions	82
11.1 KDF1	82
11.2 KDF2	82
12. Auxiliary techniques.....	83
12.1 Hash functions	83
12.2 Mask generation functions.....	83
12.3 Key confirmation functions	84
12.4 Converting between private keys and their octet string representations	84
12.5 Multiplier value creation functions for APKAS-SRP6.....	85
Annex A (informative) Number-theoretic background	87
Annex B (normative) Conformance	92
Annex C (informative) Rationale	96
Annex D (informative) Security considerations	102
Annex E (informative) Formats.....	123
Annex F (informative) Bibliography	124

IEEE Standard Specification for Password-Based Public-Key Cryptographic Techniques

IMPORTANT NOTICE: This standard is not intended to assure safety, security, health, or environmental protection in all circumstances. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

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

1. Overview

1.1 Scope

This standard covers specifications of common public-key cryptographic techniques for performing password-based authentication and key establishment, supplemental to the techniques described in IEEE Std 1363™-2000 and IEEE Std 1363a™-2004.¹ It includes specifications of primitives and schemes designed to utilize passwords and other low-grade secrets as a basis for securing electronic transactions, including schemes for password-authenticated key agreement and password-authenticated key retrieval.

1.2 Purpose

Ensuring privacy and authenticity in personal electronic transactions is a process that necessarily involves human beings. Memorized secrets are an important factor in human authentication. Many common cryptographic methods for authentication require large, random high-grade secret keys; yet, the secrets that human beings can conveniently memorize and reliably reproduce tend to be low-grade secrets. Passwords are widely used low-grade secrets that are typically not-so-random and relatively small, and introduce risks of brute-force attack when inappropriately used as cryptographic keys.

¹ Information on references can be found in Clause 2.

IEEE Std 1363.2™-2008 specifies public-key cryptographic techniques specifically designed to securely perform password-based authentication and key establishment. These techniques provide a way to authenticate people and distribute high-quality cryptographic keys for people, while preventing off-line brute-force attacks associated with passwords. A resulting high quality key may be more confidently used in combination with other cryptographic methods, such as symmetric encryption methods and public-key encryption, identification, and digital signature methods. IEEE Std 1363.2-2008 provides a reference for a variety of such password-based techniques within a suitable framework.

It is not the purpose of this document to mandate any particular set of password-based techniques or security requirements (including key sizes). Rather, the purpose is to provide: (1) a reference for specification of a variety of techniques from which applications may select, (2) the appropriate theoretic background, and (3) extensive discussion of security and implementation considerations so that a solution provider can choose appropriate security requirements.

1.3 Organization of the document

This standard is written as a companion document to IEEE Std 1363-2000 and IEEE Std 1363a-2004. IEEE Std 1363.2-2008 can be used as a standalone reference for description of password-based methods, but it heavily relies upon background material in, and uses techniques defined in, these earlier documents. Some familiarity with IEEE Std 1363-2000 and the field of cryptography is assumed.

The structure of this document roughly parallels that of IEEE Std 1363-2000, to make it easy to find relevant information, and to facilitate a potential merger of these documents in a future version of the standard.

This document contains two parts: the main document and seven annexes. The annexes provide background and helpful information for the users of this standard, as follows.

1.3.1 Structure of the main document

- Clause 1, Overview
- Clause 2, Normative references
- Clause 3, Definitions, acronyms, and abbreviations
- Clause 4, Types of cryptographic techniques
- Clause 5, Mathematical conventions
- Clause 6, The DL setting
- Clause 7, The EC setting
- Clause 8, Primitives
- Clause 9, Password-authenticated key agreement schemes
- Clause 10, Password-authenticated key retrieval schemes
- Clause 11, Key derivation functions
- Clause 12, Auxiliary techniques

1.3.2 Structure of the annexes

The annexes include the following:

- Annex A (informative) Number-theoretic background
- Annex B (normative) Conformance
- Annex C (informative) Rationale
- Annex D (informative) Security considerations
- Annex E (informative) Formats
- Annex F (informative) Bibliography

1.4 Document conventions

1.4.1 Bracket notation

For method descriptions that describe two or more related methods that work in different contexts, context-specific description may be found in { } brackets. Examples of how this notation is used are as follows:

- {DL,EC}, and the corresponding {discrete logarithm, elliptic curve}, are used to describe methods in both the discrete logarithm (DL) and elliptic curve (EC) families.
- {CLIENT,SERVER}, and the corresponding {Client, Server}, are used to describe pairs of methods for both parties in a two-party Client/Server scheme.
- {SRP3,SRP6} is used to describe each of two similar methods.
- {DL,EC}BPKAS-PAK-{CLIENT,SERVER} is equivalent to {DLBPKAS-PAK-CLIENT, DLBPKAS-PAK-SERVER, ECBPKAS-PAK-CLIENT, ECBPKAS-PAK-SERVER}.

A method description should be read consistently using just one of each alternative context.

The notation *For DL only*: indicates a feature that applies only to the DL context. The two parties that use -CLIENT and -SERVER methods are respectively designated *Client* and *Server*.

In the titles of methods defined in Clause 8, Clause 9, and Clause 10, a prefix of [DL] or [EC] indicates that the method is defined for the DL or EC setting only; otherwise the method is defined for both settings.

1.4.2 Ordering of steps

Many descriptions include an ordered sequence of steps, which may generally be performed in any order that results in an equivalent operation.

However, where a step includes substeps, all substeps shall be performed before the step is considered to be complete. For example, a step that receives a *public-key value* (see 8.1.3) and any associated substeps that abort if the value is unacceptable should all be completed before the value is used in other steps.

Steps that “output [some value] and stop” are used to limit disclosure of sensitive information. These steps shall be performed in the indicated order relative to any subsequent steps that either output a different value or send a value to another party.

1.4.3 Method parameters

Many methods are parameterized by choices of input domain parameters, functions, and related values. These parameters are generally considered to be options of the invoking scheme. For two parties to

successfully interoperate using a method, they need to use the same scheme options or parameters for the method.

Cases where related methods need to use the same value for a shared parameter in order to interoperate are specifically noted.

1.4.4 Future amendment

Where one or more functions are specified as choices that should be used for a scheme option or parameter, an amendment to this standard may specify additional functions that are also acceptable choices.

2. Normative references

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

IEEE Std 1363-2000, IEEE Standard Specifications for Public-Key Cryptography.^{2, 3}

IEEE Std 1363a-2004, IEEE Standard Specifications for Public-Key Cryptography, Amendment 1: Additional Techniques.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* [B21]⁴ should be referenced for terms not defined in this subclause.

NOTE—For some terms, multiple definitions are presented, in which case the first represents the primary definition for use within this document. Alternate definitions show different meanings that may be found in referenced literature and other common use, and are presented for historical context. See Annex D for further background discussion of these terms and related concepts.⁵

3.1.1 augmented password-authenticated key agreement (APKAS): A system for password-authenticated key agreement between a client and server that forces an adversary to perform a successful brute-force attack in order to masquerade as the client using stolen server data.

3.1.2 balanced password-authenticated key agreement (BPKAS): A system for password-authenticated key agreement in which two parties use the same shared password value to negotiate and authenticate a shared key.

² IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (<http://standards.ieee.org/>).

³ The standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

⁴ The numbers in brackets correspond to those of the bibliography in Annex F.

⁵ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

3.1.3 brute-force attack: An attack on a cryptographic system that systematically employs a trial and error search of a space of keys, passwords, or other secret data.

NOTE—For passwords, a *brute-force attack* is similar to, but may be much easier than, an *exhaustive attack*. It may not be necessary to explore the full space of all possible passwords when the targeted password is likely to be confined to a small subset of that space.

3.1.4 crack (a password): (A) (informal) Successfully brute-force attack password-derived data. (B) (informal) Successfully attack a password-based cryptographic system.

3.1.5 entropy (of a password or key): A measure of the uncertainty of the specific value that a password or key may be assigned, which is, roughly speaking, inversely related to the probability of guessing the value.

NOTE—The *entropy* of a password is relative to the state of knowledge and capabilities of the guesser.

3.1.6 dictionary attack (on a password-based system): (A) A brute-force attack using repeated guessing and verification of password values. (B) A brute-force attack using a stored list of specific password values. (C) A brute-force attack using a stored list of words from a natural language dictionary.

3.1.7 hashed password: The result of a cryptographic hash function of a password.

3.1.8 interactive proof: A two-party protocol in which a prover interacts with a verifier by exchanging messages, where both parties share a common input value, and where at the end of the interaction the verifier computes a predicate depending on the input value and the exchanged messages in order to accept or reject the input value.

NOTE—Definition of interactive proof adapted from Goldreich and Krawczyk [B19].

3.1.9 iterated hash: Repeated application of a cryptographic hash function on input data, typically used to increase the computation needed to crack a password.

3.1.10 low-grade secret: A secret that is vulnerable or presumed to be vulnerable to brute-force attack given the result of a cryptographic hash function of the secret.

3.1.11 password: A low-grade secret word, phrase, number, or sequence of characters used for authentication of identity or authorization.

3.1.12 password-authenticated key agreement: A class of key agreement methods in which parties, based only on their knowledge of password-derived data, establish a cryptographic key using an exchange of messages, such that one who controls the communication channel but does not possess the password-derived data will not be able to participate in the method and will be severely constrained from guessing the password.

3.1.13 password-authenticated key retrieval: (A) A process in which a client obtains a static key that is derived from a password-based negotiation with a server that knows data associated with the password. (B) (broadly) The retrieval of a key from a secure key repository or escrow requiring authentication derived in part from a password.

3.1.14 password-entangled public key: A public key derived from both a password and a high-grade secret private key, constructed specifically to avoid password disclosure.

3.1.15 password-limited private key: A private key where the sole source of randomness may be limited to that of a password.

3.1.16 password-limited public key: A value used as password verification data that is derived from a password-limited private key using a public-key generation method.

3.1.17 password verification data: Data that is used to verify a party's knowledge of a specific password and is typically regarded as secret data.

3.1.18 password verifier: (A) One of the parties in a password proof system that verifies knowledge of a password. (B) Password verification data.

NOTE—The second definition of *password verifier* is not used in this document, but has been used in relevant references.

3.1.19 PIN: Acronym for personal identification number, a short numeric password.

3.1.20 proof of knowledge: (A) A proof of an assertion that a prover knows something, in a system where whenever a verifier is convinced of the assertion, then the prover indeed knows it. (B) A proof of knowledge of a solution to a publicly stated computationally difficult problem.

NOTE—Definition (A), adapted from Bellare and Goldreich [B4], provides a basis for the definition of **zero knowledge password proof**. Many references refer to the narrower definition (B), which is limited to a proof that requires secret knowledge to be from a computationally large space. *See also:* **zero knowledge proof of knowledge**.

3.1.21 salt: (A) A variable incorporated as secondary input to a one-way or encryption function that derives password verification data. (B) An element that gives flavor or zest. [Presumed basis for definition (A).]

3.1.22 zero knowledge password proof: An interactive zero knowledge proof of knowledge of password-derived data shared between a prover and the corresponding verifier. *See also:* **interactive proof; password; zero knowledge proof of knowledge**.

3.1.23 zero knowledge proof: A proof of an assertion that has the property of yielding nothing but the validity of the assertion.

NOTE—Definition adapted from Goldreich and Krawczyk [B19].

3.1.24 zero knowledge proof of knowledge: A proof of knowledge that has the property of yielding nothing but the validity of the assertion that the prover knows something. *See also:* **proof of knowledge; zero knowledge proof**.

3.2 Acronyms and abbreviations

ANSI	American National Standards Institute
AMP	identifier for techniques based on Kwon [B38]
APKAS	augmented password-authenticated key agreement scheme
BPKAS	balanced password-authenticated key agreement scheme
BS2IP	bit string to integer (conversion) primitive
BS2OSP	bit string to octet string (conversion) primitive
DH	identifier for techniques based on Diffie and Hellman [B10]
DL	discrete logarithm (family of techniques)
EC	elliptic curve (family of techniques)
FE2IP	field element to integer (conversion) primitive

FE2OSP	field element to octet string (conversion) primitive
FIPS	Federal Information Processing Standards
GE2OSP	group element to octet string (conversion) primitive
GE2SVFEP	group element to secret value field element (conversion) primitive
I2BSP	integer to bit string (conversion) primitive
I2FEP	integer to field element (conversion) primitive
I2OSP	integer to octet string (conversion) primitive
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
KCF	key confirmation function
KDF	key derivation function
KRBP	key retrieval blinding primitive
KRPP	key retrieval permutation primitive
KRUP	key retrieval unblinding primitive
MGF	mask generation function
MVCF	multiplier value creation function
NIST	National Institute of Standards and Technology
OS2BSP	octet string to bit string (conversion) primitive
OS2FEP	octet string to field element (conversion) primitive
OS2IP	octet string to integer (conversion) primitive
PAK	identifier for techniques based on Boyko, MacKenzie, Patel [B8]
PEPKGP	password-entangled public-key generation primitive
PKAS	password-authenticated key agreement scheme
PKGP	public-key generation primitive
PKRS	password-authenticated key retrieval scheme
PPK	identifier for techniques based on Boyko, MacKenzie, Patel [B8]
PVDGP	password verification data generation primitive
REDP	random element derivation primitive
SPEKE	identifier for techniques based on Jablon [B31]
SRP	identifier for techniques based on Wu [B57]
SVDP	secret value derivation primitive

4. Types of cryptographic techniques

This clause gives an overview of the types of cryptographic techniques that are specified in this standard as well as some requirements for conformance with those techniques. See Annex B for more on conformance.

4.1 Brief history of the field

The first generation of the era of public cryptography was a time of rapid discovery, research, and development. Within this period, the concept of *zero knowledge proof* was introduced and rapidly refined. However, clear formulations of similar problems in password-based cryptography were initially elusive, like the concept of a *zero knowledge password proof*.⁶ How can one party securely prove knowledge of a password to another party over an insecure channel without using other prearranged cryptographic keys? This question was answered only within the last few years of this period. Once the first solutions were found (see Bellare and Merritt [B7], [B6], and Gong et al. [B20]), the field blossomed, resulting in the methods described and referenced in this document.

4.2 Variations of the network password problem

This standard describes three classes of password-based methods that solve three variations of the password-only network login problem. These methods can provide mutual zero knowledge password proof and remote password-authenticated establishment of cryptographic keys.

In the first class of methods, for *balanced password-authenticated key agreement*, two parties share a common password and they want to prove to each other that they know the password, and only then engage in secure communications, without revealing the password to others.

The second class of *augmented password-authenticated key agreement* methods is similar to the first, except that one of the parties, the Server, has password verification data derived using a one-way function of the password. In this case, although the sensitivity of the password verification data is still dependent on the quality of the password, the one-way function may provide extra protection for the Server's stored password-derived data.

The third class of *password-authenticated key retrieval* methods addresses the scenario where one desires to further decrease the sensitivity of stored password-derived data. These methods may be used to split password verification data into two or more shares, such that no number of shares less than a given threshold permits a brute-force password attack.

All these methods require one or more parties to use specific password-related data to make the method succeed. But when the participant does not use the correct password-related data, then the method is designed to fail in a way that does not reveal the password to those that do not already know the password. An adversary that does not know the appropriate password, who observes a legitimate interaction, or attempts to participate in the method with a legitimate party, is prevented from obtaining information that leads to unconstrained off-line guessing of the password. However, even in optimal cases, all methods permit an adversary to verify a minimum of one guess in each run of the method.

4.3 General model

As stated in Clause 1, the purpose of this standard is to provide a reference for specifications of a variety of password-based public-key cryptographic techniques from which applications may select. Therefore, it is desirable to define these techniques in a framework that would allow selection of techniques appropriate for particular applications. Different types of cryptographic techniques can be viewed abstractly according to the following general model, including primitives, schemes, and additional methods, the definitions of which are in IEEE Std 1363-2000 and are repeated here.

⁶ A zero knowledge password proof is, essentially, an interactive zero knowledge proof of knowledge of a small secret. See 3.1.22 for a definition of this and other italicized terms.

- *Primitives*: Basic mathematical operations. Historically, they were discovered based on number-theoretic hard problems. Primitives are not meant to achieve security just by themselves, but they serve as building blocks for schemes.
- *Schemes*: A collection of related operations combining primitives and additional methods. Schemes can provide complexity-theoretic security, which is enhanced when they are appropriately applied in protocols.
- *Additional methods*: Other components of schemes. These include key derivation functions (KDF), hash functions, key confirmation functions (KCF), and other techniques that provide security based on a broad blend of cryptographic analytic assumptions beyond number theory.

From an implementation viewpoint, primitives can be viewed as low-level implementations (e.g., implemented within cryptographic accelerators or software modules), schemes and additional methods can be viewed as medium-level implementations (e.g., implemented within cryptographic service libraries), and protocols can be viewed as high-level implementations (e.g., implemented within entire sets of applications).

In a departure from IEEE Std 1363-2000, this standard defines schemes that interact with related schemes in multiple stages, using intermediate transmitted and received values, as well as input and output values. Intermediate values are exchanged between parties during the operation of these schemes. And in addition to the added functions of password-based authentication, the scope of the key agreement schemes here is also broader in describing key confirmation operations.

This document describes the requirements for schemes at a detailed functional level, but application-specific data transmission formats and techniques are outside the scope of this standard.

General frameworks for primitives, additional methods, and schemes, are provided in 4.4, 4.5, 4.6, and 4.7 and specific techniques are defined in Clause 6 through Clause 12. This document also references techniques defined in IEEE Std 1363-2000 as components for constructing cryptographic primitives and schemes. This standard does not define application-specific protocols; the techniques defined in this standard are rather components used in such protocols. Annex D discusses security considerations related to how the techniques can be used in protocols and applications to achieve certain security attributes.

4.4 Primitives

The following types of primitives are defined in this standard:

- Random element derivation primitives (REDP), components of password-authenticated key agreement schemes (PKAS) and password-authenticated key retrieval schemes (PKRS).
- Password-entangled public-key generation primitives (PEPKGP); components of PKASs and PKRSs.
- Secret value derivation primitives (SVDP), components of password-authenticated key agreement and PKRSs.
- Password verification data generation primitives (PVDGP), components of augmented password-authenticated key agreement schemes (APKAS).
- Key retrieval blinding primitives (KRBP), key retrieval unblinding primitives (KRUP), and key retrieval permutation primitives (KRPP), components of key retrieval schemes.

Primitives in this standard are presented as mathematical operations, and they are typically not meant to provide security benefits by themselves. Such benefits can only be provided by the use of a primitive, along with other operations, in a properly implemented scheme.

Primitives assume that their inputs satisfy certain assumptions, as listed with the specification of each primitive. An implementation of a primitive is unconstrained on an input not satisfying the assumptions, as long as it does not adversely affect future operation of the implementation; the implementation may or may not return an error condition. For example, an implementation of a public-key generation primitive (PKGP) may return something that looks like a public key if its input was not a valid private key. It may also reject the input. It is up to the user of the primitive to guarantee that the input will satisfy the constraints or to include the relevant checks. For example, the user may choose to use relevant key and domain parameter validation techniques.

The specification of a primitive consists of the following information:

- *Input* to the primitive
- *Assumptions* about the input made in the description of the operation performed by the primitive
- *Output* from the primitive
- *Operation* performed by the primitive, expressed as a series of steps
- *Conformance region recommendations* describing the minimum recommended set of inputs for which an implementation should operate in conformance with the primitive (see Annex B for more on conformance)

Primitive specifications are functional specifications, not interface specifications. The format of inputs and outputs and the procedure by which an implementation of a scheme is invoked are outside the scope of this standard. See Annex E for more information on input and output formats.

The input and output values for primitives include *domain parameters* and *keys*, which may be defined as integers, EC points, field elements or *group elements*. These concepts are defined in 6.1 for the DL setting, in 7.1 for the EC setting, and in 8.1 for generalized groups.

4.4.1 Random element derivation primitives (REDP) model

An REDP has the following form:

Input:

- Domain parameters
- A random selector value

Output: The derived generator of the desired group, or “invalid” according to the result.

The REDP applies cryptographic hash operations to the selector value to derive a pseudo-random element of order r that is a generator of the desired group defined by the domain parameters. The function may output “invalid” for an input value that does not correspond to a generator of the desired group; however, the probability of a random input producing an invalid result should be negligibly small for any suitable choice of parameters.

The required security properties of REDPs are discussed in D.2.1.20.

4.4.2 Public-key generation primitives (PKGP) model

A PKGP has the following form:

Input:

- Domain parameters
- A valid private key

Output: The public key, or “invalid”

The PKGP applies cryptographic operations to the input to generate a public key. It outputs the public key or “invalid” according to the result.

NOTE—*Public keys* are a common feature of all key agreement schemes. This document defines the Diffie-Hellman public-key generation primitives (PKGP-DH), in accordance with the definitions of Diffie-Hellman public keys in IEEE Std 1363-2000, to support various schemes defined in this document.

4.4.3 Password-entangled public-key generation primitives (PEPKGP) model

A PEPKGP has the following form:

Input:

- Domain parameters
- A low-grade secret π , typically derived from a password
- A valid private key

Output: The password-entangled public key, or “invalid”

The PEPKGP applies cryptographic operations to the input to generate a *password-entangled public key*. It outputs the password-entangled public key or “invalid” according to the result.

A password-entangled public key is a public key that is additionally derived in part from a password. The password becomes entangled in the process of deriving the high-grade shared key in a way that prevents parties without access to the sensitive password-related data from successfully participating in the method and also prevents them from performing off-line password guessing.

A PEPKGP function $Pepkgp$ computes a password-entangled public key based on a password-based value octet string π and a private key integer s – as in $w = Pepkgp(\pi, s)$. Each PEPKGP is designed to make it infeasible for an adversary who chooses w to compute any significant number of pairs of values $\{\pi_i, s_i\}$ that corresponding to plausible passwords, such that each $Pepkgp(\pi_i, s_i)$ produces the same w . The number of $\{\pi_i, s_i\}$ pairs that an adversary can compute for a single w typically represents the number of password guesses that the adversary can make in one run of a PKAS. In the optimal case, only one pair $\{\pi_1, s_1\}$ is known and no alternate pairs can be computed, which may limit the adversary to making just one password guess (π_1) in each run of the scheme.

The application and deployment scenario determines whether a specific number of guesses represents a significant threat.

4.4.4 Secret value derivation primitives (SVDP) model

For the purposes of a PKAS, an SVDP has the following form:

Input:

- Domain parameters
- The party’s own private key

- The other party's public key or password-entangled public key

For specific cases only:

- The party's own low-grade secret π

Output: The derived shared secret value, or "invalid"

The SVDGP applies cryptographic operations to the input to generate a shared secret value. It outputs the shared secret value as output or it may output "invalid" according to the result.

4.4.5 Password verification data generation primitives (PVDGP) model

A PVDGP has the following form:

Input:

- Domain parameters
- A low-grade secret π , typically derived from a password

Output: Shared secret password verification data and other related secret values associated with the password.

The PVDGP applies cryptographic operations to the input password-based value π to generate and output password verification data and other related values. The value π may also incorporate salt, identifiers, or other values. (See D.2.1.15.) A Server party in an APKAS uses password verification data to verify an associated Client party's knowledge of π . Password verification data values are stored and used by a Server and are considered shared secrets between the Client and Server.

A PVDGP may use a *password-limited private key*, where the sole source of randomness in the key may be limited to that of a password. Password verification data may include *password-limited public keys* that are computed from a password-limited private key using a public-key generation or related exponential method. Password-limited public keys generally need to be kept secret; they should not be confused with true public keys that are designed to be revealed to the public. Both parties share responsibility for maintaining the secrecy of all password verification data.

The PVDGP may also output other secret values associated with the password that shall be kept secret by the Client and are used in APKAS key agreement operations.

4.5 Schemes

This standard focuses on password-authenticated schemes, to specifically leverage the strengths and address the weaknesses of low-grade secrets. The principal difference of a password-authenticated scheme, as compared to a key agreement scheme (KAS) (as defined in IEEE Std 1363-2000), is that one or more of the prearranged secrets that authenticate the parties may be *low-grade* secrets, such as passwords. As such, when passwords are used for mutual authentication, they are typically treated as shared secrets, in contrast with static public and private keys.

The value π is used to refer to a password-based value in descriptions of these methods.

The following types of schemes are defined in this standard:

- Balanced password-authenticated key agreement schemes (BPKAS), in which two parties that share π use ephemeral private keys with π to mutually derive an ephemeral shared key.
- Augmented password-authenticated key agreement schemes (APKAS), in which two parties use ephemeral private keys and corresponding password values, where one party has π and the other party has a one-way function π , to mutually derive an ephemeral shared key.
- Password-authenticated key retrieval schemes (PKRS), in which a Client party establishes one or more static keys using π and the assistance of a Server party that has secret data associated with π .

Of the two kinds of PKAS, balanced and augmented, an APKAS has the distinction that the Server's verification data is a one-way function of the Client's password, so that a thief that knows the verification data cannot masquerade directly as the Client without first cracking the password.

A PKRS differs from a PKAS in that a PKRS establishes *static* keys for the Client that are not necessarily known or derivable by the Server, whereas a PKAS establishes *ephemeral* keys that are shared by both parties. A PKRS may be used in multi-Server systems to limit vulnerabilities due to one or more compromised Servers.

Schemes in this standard (as in IEEE Std 1363-2000) are presented in a general form based on certain primitives and additional methods, such as key derivation methods. For example, a BPKAS is based on a PKGP, an SVDP, and a KDF.

Schemes also include key management operations, such as selecting a private key or obtaining another party's public key. For proper security, a party may need to be assured of the validity of the keys and domain parameters. Generation of domain parameters and keys needs to be performed properly, and in some cases, acceptability tests also need to be performed. Issues of proper key management that may be essential for security and yet are outside the scope of this standard are addressed for informational purposes in Annex D.

The specification of a scheme consists of the following information:

- *Scheme options*, such as choices for primitives and additional methods
- One or more *operations*, depending on the scheme, expressed as a series of *steps*
- *Conformance region recommendations* for implementations conforming with the scheme (see Annex B for more on conformance)

Scheme specifications are functional specifications, not interface specifications. The format of inputs and outputs and the procedure by which an implementation of a scheme is invoked are outside the scope of this standard. See Annex E for more information on input and output formats.

4.6 Additional methods

This standard specifies the following additional methods:

- Hash functions, which are used in many primitives and schemes
- Mask generation functions (MGF), which are used in many primitives and schemes
- Key confirmation functions (KCF), which are used in password authenticated key agreement schemes
- Multiplier value creation functions (MVCF), which are used in APKAS-SRP6.

The specified additional methods are strongly recommended for use in the schemes. The use of an inadequate KCF may compromise the security of the scheme in which it is used. Therefore, a thorough

security analysis should be performed for any implementation that does not use the recommended additional methods for a particular scheme.

4.7 Table summary

This subclause gives a summary of all the 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
Balanced password-authenticated key agreement schemes		
{DL,EC}BPKAS-PPK-CLIENT, {DL,EC}BPKAS-PPK-SERVER	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-PAK, and {DL,EC}SVDPAK2	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}BPKAS-PAK-CLIENT	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-PAK, and {DL,EC}SVDPAK1-CLIENT	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}BPKAS-PAK-SERVER	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PKGP-DH, and {DL,EC}SVDPAK2	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}BPKAS-SPEKE-CLIENT, {DL,EC}BPKAS-SPEKE-SERVER	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-SPEKE, and {DL,EC}SVDPAK-SPEKE	KDF1 or KDF2 (each uses a hash function), KCF1
Augmented password-authenticated key agreement schemes		
{DL,EC}APKAS-AMP-CLIENT	{DL,EC}PKGP-DH and {DL,EC}SVDPAK-AMP-CLIENT	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}APKAS-AMP-SERVER	{DL,EC}PVDGP-AMP, {DL,EC}PEPKGP-AMP-SERVER, and {DL,EC}SVDPAK-AMP-SERVER	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}APKAS-BSPEKE2-CLIENT	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-SPEKE, and {DL,EC}SVDPAK-SPEKE	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}APKAS-BSPEKE2-SERVER	{DL,EC}PVDGP-BSPEKE2, ({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-SPEKE, and {DL,EC}SVDPAK-SPEKE	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}APKAS-WSPEKE-CLIENT	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-SPEKE, and {DL,EC}SVDPAK-WSPEKE-CLIENT	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}APKAS-WSPEKE-SERVER	{DL,EC}PVDGP-BSPEKE2, ({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-SPEKE, and {DL,EC}SVDPAK-WSPEKE-SERVER	KDF1 or KDF2 (each uses a hash function), KCF1

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

Scheme name	Primitives	Additional methods
{DL,EC}APKAS-PAKZ-CLIENT	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-PAK, and {DL,EC}SVDGP-PAK1-CLIENT	KDF1 or KDF2 (each uses a hash function), KCF1
{DL,EC}APKAS-PAKZ-SERVER	{DL,EC}PVDGP-PAKZ, ({DL,EC}REDP-1 or REDP-2), {DL,EC}PKGP-DH, and {DL,EC}SVDGP-PAK2	KDF1 or KDF2 (each uses a hash function), KCF1
DLAPKAS-SRP3-CLIENT	DLPVDGP3-SRP3, DLPKGP-SRP-CLIENT, and DLSVDGP-SRP3-CLIENT	KDF1 or KDF2 (each uses a hash function), KCF1
DLAPKAS-SRP3-SERVER	DLPVDGP-SRP3, DLPEPKGP-SRP3-SERVER, and DLSVDGP-SRP3-SERVER	KDF1 or KDF2 (each uses a hash function), KCF1
ECAPKAS-SRP5-CLIENT	ECPVDGP-SRP5, ECPKGP-DH, and ECSVDGP-SRP5-CLIENT	KDF1 or KDF2 (each uses a hash function)
ECAPKAS-SRP5-SERVER	ECPVDGP-SRP5, ECPEPKGP-SRP5-SERVER, and ECSVDGP-SRP5-SERVER	KDF1 or KDF2 (each uses a hash function), KCF1
Augmented password-authenticated key agreement schemes (continued)		
DLAPKAS-SRP6-CLIENT	DLPVDGP-SRP6, DLPKGP-SRP-CLIENT, and DLSVDGP-SRP6-CLIENT	KDF1 or KDF2 (each uses a hash function), KCF1
DLAPKAS-SRP6-SERVER	DLPVDGP-SRP6, DLPEPKGP-SRP6-SERVER, and DLSVDGP-SRP6-SERVER	KDF1 or KDF2 (each uses a hash function), KCF1
Password-authenticated Key Retrieval Schemes		
{DL,EC}PKRS-1-CLIENT	(({DL,EC}REDP-1 or REDP-2), {DL,EC}PEPKGP-1, and {DL,EC}KRBP-1, KRUP-1	KDF1 or KDF2 (each uses a hash function)
{DL,EC}PKRS-1-SERVER	{DL,EC}KRPP-1	

5. Mathematical conventions

For most of the mathematical conventions, including description of notation, representation of bit strings, octet strings, finite field elements, and data type conversions, the reader is referred to IEEE Std 1363-2000, Clause 5. For convenient reference, an outline of this clause is reproduced here.

Outline of IEEE Std 1363-2000, Clause 5, as amended by IEEE Std 1363a-2004:

5.1 Mathematical notation

5.2 Bit strings and octet strings

5.3 Finite fields

5.3.1 Prime finite fields

5.3.2 Characteristic two finite fields

5.4 Elliptic curves and points

5.5 Data type conversion

5.5.1 Converting between integers and bit strings (I2BSP and BS2IP)

5.5.2 Converting between bit strings and octet strings (BS2OSP and OS2BSP)

5.5.3 Converting between integers and octet strings (I2OSP and OS2IP)

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

5.5.5 Converting finite field elements to integers (FE2IP)

5.5.6 Converting between elliptic curve points and octet strings

Exceptions and additions to these conventions are noted here.

5.1 Mathematical notation

5.1.1 Group notation

To unify the presentation of methods that work in multiple settings, this document provides generalized descriptions for methods that may use either the *DL setting* (6.1) or the *EC setting* (7.1). In these generalized descriptions, the convention is to use lowercase letters to represent DL group elements, and to generally use the same notation as that used in IEEE Std 1363-2000 in the DL setting.

Although group operations and group elements are instantiated differently in the DL and EC settings, the following $*$ and ^{n} operators are used to describe operations in both settings:

- $e_1 * e_2$ denotes the group operator applied to group elements e_1 and e_2 , and
- e^n denotes the iterated group operation $e * e * \dots * e$ on n values of group element e .

These operators assume that n is an integer and e , e_1 , and e_2 are elements of an appropriate group defined by either DL or EC domain parameters. (See 6.1.1, 7.1.1, and 8.1.1 for the DL, EC, and generalized group settings, respectively.)

NOTE 1—As in IEEE Std 1363-2000, this document uses $a \times b$ to denote multiplication of field elements a and b , multiplication of integers a and b , and scalar multiplication of an EC point b by integer a . Similarly, the $\times$ symbol may be omitted when such omission (ab) does not cause ambiguity. This document uses the notation $\text{exp}(w, c)$, to denote raising DL field element w to the power of integer c . The notation w^c denotes raising integer w to the power of integer c .

NOTE 2—As in IEEE Std 1363-2000, throughout this document, operations on finite field elements, group elements, and integers are used. Care needs to be exercised to distinguish between integers, field elements, and group elements, especially as operations on integers and field elements are sometimes denoted by the same symbols. See IEEE Std 1363-2000, 5.3, for more information on finite fields.

NOTE 3—Further discussion of the analogous relationship between the DL and EC settings can be found in IEEE Std 1363-2000, A.9.4.

5.2 Bit strings and octet strings

5.2.1 Single octet values

The notation $\text{hex}(nn)$ represents a single octet with hexadecimal value nn , where n is a hexadecimal digit.

5.2.2 Empty octet string

The notation $""$ represents the empty octet string of length zero.

5.3 Data type conversion

5.3.1 Converting integers to field elements (I2FEP)

In performing cryptographic operations, non-negative integers sometimes need to be converted to finite field elements. The primitive that performs this operation is called the *integer to field element conversion primitive*, or I2FEP.

An integer i in the range $[0, q-1]$ shall be converted to an element j of a finite field $GF(q)$ by the following or an equivalent procedure:

- a) Convert i to an octet string of length $\lceil \log_{256}(q) \rceil$ using I2OSP.
- b) Convert the resulting octet string to a field element j using OS2FEP.
- c) Output j .

5.3.2 Converting group elements to secret value field elements (GE2SVFEP)

This document uses the *group element to secret value field element primitive* (GE2SVFEP) to convert group elements into field elements. This primitive is defined in both the DL and EC settings.

The GE2SVFEP function shall convert a group element e to a secret value field element by the following, or an equivalent, procedure:

- a) In the DL setting:
 - 1) Output the field element e . (In the DL setting, group elements are field elements.)
- b) In the EC setting:
 - 1) If e is the identity element (the point at infinity $\mathcal{O}$), output field element 0 and stop.
 - 2) Set field element x_e = the x -coordinate of EC point e .
 - 3) Output field element x_e .

NOTE 1—There is no corresponding inverse function (like “SVFE2GEP”) because there is a loss of information in the conversion process from a group element to a field element in the EC setting; the y coordinate is omitted. The GE2SVFEP function is used only for the specific case of deriving the final output of an SVDP and helps to unify the generalized description of primitives that work in either a DL or EC setting.

NOTE 2—The conversion for the EC group identity element is arbitrary and may duplicate the result of a non-identity group element at $(0,y)$. Methods that abort for unacceptable values generally do so before invoking GE2SVFEP or may reject all 0 results under the assumption that the probability of obtaining a valid 0 result is negligible.

5.3.3 Converting group elements to octet strings (GE2OSP-X)

This document uses the Group Element to Octet String Secret primitive (GE2OSP-X) to convert group elements into octet strings. This primitive is defined in both the DL and EC settings.

The GE2OSP-X function shall convert a group element e to an octet string by the following, or an equivalent, procedure:

- a) In the DL setting:

- 1) Compute octet string $o_e = \text{FE2OSP}(e)$ (In the DL setting, group elements are field elements.)
- 2) Output octet string o_e
- b) In the EC setting:
 - 1) Compute octet string $o_e = \text{EC2OSP-X}(e)$ (See IEEE Std 1363a-2004, 5.5.6.3.)
 - 2) Output octet string o_e

NOTE 1—There is no corresponding inverse function (like “OS2GEP”) because there is a loss of information in the conversion process from a group element to an octet string in the EC setting; the y coordinate is omitted.

NOTE 2—In the EC setting, $\text{GE2OSP-X}(e)$ is *not* equivalent to $\text{FE2OSP}(\text{GE2SVFEP}(e))$, due to the initial format octet provided by EC2OSP-X .

6. The DL setting

This clause specifies the DL setting for the family of cryptographic primitives based on the discrete logarithm problem over multiplicative groups in finite fields, also known as the *DL family*. The reader is referred to IEEE Std 1363-2000, Clause 6, for more information, an outline of which is presented here.

Outline of IEEE Std 1363-2000, Clause 6:

6.1 The DL setting

6.1.1 Notation

6.1.2 DL domain parameters

6.2 Primitives

6.2.1 DLSVDP-DH

6.2.2 DLSVDP-DHC

6.1 The DL setting

6.1.1 Notation

The notation used in the DL setting as defined in IEEE Std 1363-2000, 6.1.1, applies to this document.

6.1.2 DL domain parameters

The definition of *DL domain parameters* from IEEE Std 1363a-2004, 6.1.2, applies to this document, with some modifications noted here.

See 8.1.2 for the generalized definition of domain parameters for both the DL and EC settings. In the DL setting, the *parent group* is the multiplicative group of all $(k \times r)$ non-zero field elements in $GF(q)$ and r divides $(q - 1)$. For most methods in the DL setting, the group generated by g is referred to as the *desired group*, and *public keys* are generally described as group elements. See 8.1.2.1 for the exceptions of SRP3 and SRP6, which operate in the parent group, and where public keys may also be described as DL field elements.

6.1.3 DL key pairs

The definitions of *DL key pair*, *DL private key*, *DL public key*, *valid DL private key*, and *valid DL public key* for most of the methods defined in this document are in IEEE Std 1363-2000, 6.1, and apply to this document. See 8.1.3 for the generalized definition of key pairs.

7. The EC setting

This clause specifies the EC setting for the family of cryptographic primitives based on the discrete logarithm problem over elliptic curve groups in finite fields, also known as the *EC family*. The reader is referred to IEEE Std 1363-2000, Clause 7, for more information, an outline of which is presented here.

Outline of IEEE Std 1363-2000, Clause 7:

7.1 The EC setting

7.1.1 Notation

7.1.2 EC domain parameters

7.1.3 EC key pairs

7.2 Primitives

7.2.1 ECSVDP-DH

7.2.2 ECSVDP-DHC

7.1 The EC setting

7.1.1 Notation

When EC operations are specified in terms of elliptic curve points, the same notation is used as in IEEE Std 1363-2000. That is, $P + Q$ denotes point addition of points P and Q , and $n \times P$ (or simply nP , when omission of $\times$ does not cause ambiguity) denotes scalar point multiplication of point P by integer n .

7.1.2 EC domain parameters

The definition of *EC domain parameters* from IEEE Std 1363a-2004, 7.1.2, applies to this document.

See 8.1.2 for the generalized definition of domain parameters for both the DL and EC settings. In the EC setting, the *parent group* is the group of points on the elliptic curve E defined by the two EC domain parameters coefficients a and b [elements of $GF(q)$], the parent group order r divides $\#E$ (the number of points on E), and cofactor $k = \#E / r$. For all methods in the EC setting, the desired group is the group of r group elements generated by g , where g is a synonym for the EC generator point G , and *public keys* are generally described as group elements.

7.1.3 EC key pairs

The definitions of *EC key pair*, *EC private key*, *EC public key*, *valid EC private key*, and *valid EC public key* are in IEEE Std 1363-2000, 7.1, and apply to this document. See 8.1.3 for the generalized definition of key pairs.

8. Primitives

This clause specifies cryptographic primitives for use with schemes in multiple families of underlying groups. Many of the primitives in this clause are defined in a general manner for both the DL setting (6.1) and the EC setting (7.1).

8.1 Notation and definitions

8.1.1 The generalized group setting

A generalized group setting notation and definitions are described in this clause for the primitives in Clause 8 and the schemes in Clause 9 and Clause 10. These methods may use either DL or EC settings, where $*$ denotes the group operator and $^{\wedge}$ denotes exponentiation for the appropriate group, as described in 5.1.1.

8.1.2 Generalized domain parameters

A generalized definition of the domain parameters q , k , r , and g for both the DL and EC settings is presented here. Further detail for the DL and EC settings is presented in 6.1.2 and 7.1.2, respectively.

Domain parameters are used in every primitive and scheme and are an implicit component of every key. A set of domain parameters in either the DL or EC setting specifies the following:

- 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 large prime integer r designating the order of the *desired group*
- A *parent group* of $(k \times r)$ group elements, where k is designated the cofactor
- A group element g of order r that generates the desired group (g is a *generator* of the desired group)

If $q = 2^m$, the set of domain parameters also specifies a representation for the elements of $GF(q)$ to be used by the conversion primitives (see 5.3 and IEEE Std 1363-2000, 5.3.2). If EC key validation is to be performed, or if the SVDP-DHC or SVDP-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 IEEE Std 1363a-2004, A.1.1). For all methods other than SRP3 and SRP6, the group generated by g is referred to as the *desired group*. Members of the desired group may be referred to as *group elements*, and *public keys* are generally described as group elements.

8.1.2.1 Domain parameters for SRP3 and SRP6

For the SRP3 and SRP6 methods, the *desired group* is the parent group, which is generated by an additional domain parameter element g_{q-1} of order $(k \times r)$, instead of the element g of order r . The SRP3 and SRP6 methods are described only for this modified DL setting.

NOTE—The definition of *valid {DL,EC} domain parameters* is in IEEE Std 1363-2000, 6.1.2 (for DL) and 7.1.2 (for EC), and generally applies to this document, with the exception that SRP3 and SRP6 require the additional domain parameter g_{q-1} . See D.2.2.4.2.

8.1.3 Key pairs

The definitions of *key pair*, *private key*, *public key*, *valid private key*, and *valid public key* for most of the methods defined in this document are in IEEE Std 1363-2000, 6.1 (DL) and 7.1 (EC), and apply to this document. For convenience, these definitions are restated here.

For all methods other than the SRP3 and SRP6 methods:

- A *valid private key* (designated as s) is an integer in the range $[1, r-1]$
- A *valid public key* is a group element of order r in the desired group generated by g
- The public key w that corresponds to s is computed as $w = g^s$

The qualified term *public-key value* indicates a value that represents a valid public key in an interaction solely between legitimate compliant parties, but may represent an invalid public key in a threat or error scenario.

8.1.3.1 Key pairs for SRP3 and SRP6

For the SRP3 and SRP6 methods:

- A *valid private key* (designated as s) is an integer in the range $[1, kr-1]$
- A *valid public key* is a field element in $GF(q)$ other than the additive identity element (0) or the multiplicative identity element (1)
- The public key w that corresponds to s is computed as $w = g_{q-1}^s$.

NOTE 1—The concept of *valid public key* is not particularly relevant to SRP3 and SRP6, since these methods incorporate other necessary acceptability tests for received public-key values.

NOTE 2—SRP3 and SRP6 public keys are generally described in terms of DL field elements, instead of group elements.

8.1.4 Password-entangled public key

A *password entangled public key* is a public key that is derived from a private key and a password-based value using a *password-entangled public key generation primitive* (PEPKGP). This document defines various PEPKGP functions that are designed for use by specific schemes.

8.1.5 Acceptable (password-entangled) public-key values

Some methods in this document use the concept of an *acceptable* public-key value, which may be a *password-entangled* public-key value, where the definition of *acceptable* may depend on the specific method. The requirements for an acceptable public-key value are generally looser than for a valid public key and may allow for the use of simpler or more efficient acceptability tests. For example, in some of the PAK methods, an acceptable password-entangled public-key value is any element of the parent group [of order $(k \times r)$], whereas in some of the SPEKE methods, an acceptable password-entangled public-key value generates a sufficiently large order subgroup of the parent group.

Note that the identity element and other small order elements of the parent group may be treated as special cases in these schemes, primitives, and validation techniques, and that the definition of *sufficiently large* may depend on both the domain parameters and the desired level of security to be achieved.

8.1.5.1 Public-key tests

Algorithms for determining whether key values are acceptable for specific PKAS methods are described in A.1.

General algorithms to verify that a DL public-key value is valid and to verify that a value is in the DL parent group are given in IEEE Std 1363-2000, A.16.6. General algorithms to verify that an EC public-key value is valid and to verify that a value is in the EC parent group are given in IEEE Std 1363-2000, A.16.10.

Updated discussion of further key validation issues is in IEEE Std 1363a-2004, D.5.1.6.

8.1.6 Summary of terms

This subclause lists terms used in Clause 8, Clause 9, and Clause 10. It is meant as a reference guide only; for complete definitions of the terms listed, refer to the appropriate text. Some other terms are also used occasionally; they are introduced in the text where appropriate.

π	A password-based octet string, used for authentication. π is generally derived from a password or a hashed password, and may incorporate a salt value, identifiers for one or more parties, and/or other shared data.
π_m	Password-based group element, derived from a function of π
π_R, π_T	Password-based group elements, derived from a function of π , respectively associated with received and transmitted values
b_1	A Boolean value, indicating whether cofactor multiplication is desired
C_S	A Server's key confirmation value octet string
C_C	A Client's key confirmation value octet string
$e, e_1, e_2, \dots$	Group elements
g	A group element of order r in the desired group (part of the domain parameters for most schemes and primitives)
g_π	A group element of order r in the desired group, derived from a function of π
g_a, g_b	Group elements of order r in the desired group (parameters for REDP-2)
$GF(q)$	The Galois field of order q , which may be a prime field $GF(p)$, a binary field $GF(2^m)$, or an odd characteristic extension field $GF(p^m)$ (see IEEE Std 1363-2000, A.1.2 and A.3.1, and IEEE Std 1363a-2004, A.17)
g_{q-1}	A field element of multiplicative order $q-1$ in $GF(q)$ (part of the DL domain parameters for the SRP3 and SRP6 schemes and primitives)
<i>HashWC</i>	A hash function used on a Client's public key in APKAS-AMP
<i>HashWS</i>	A hash function used on a Server's public key in APKAS-WSPEKE
<i>HashW2</i>	A hash function used on public keys in APKAS-SRP5 and APKAS-SRP6
<i>HashKC</i>	A hash function used in key confirmation
<i>HashPVD</i>	A hash function used in password verification data generation
<i>HashRE</i>	A hash function used in random element derivation
<i>HashSK</i>	A hash function used on signature private keys used in APKAS-PAKZ

H_u	An octet string representing the hash of a signature private key u used in APKAS-PAKZ
$i_1, i_2, \dots$	Integer values
k	The integer cofactor value equal to the order of the parent group divided by r (part of the domain parameters)
$K_1, K_2, \dots$	Shared secret keys, octet strings derived in a key agreement scheme
Kdf	A key derivation function
M	A message octet string used in APKAS-PAKZ
Mgf	A mask generation function
m_v	A field element multiplier value used in APKAS-SRP6
$Mvcf$	A multiplier value creation function used to compute m_v in APKAS-SRP6
o_{ID}	An octet string parameter used in APKAS-AMP and APKAS-PAKZ
o_π	A selector value octet string derived from a function of π
$o_1, o_2, \dots$	Octet string values
$oLen$	Integer length in octets of the output of a hash function
o_u	An octet string representing private signature key u used in APKAS-PAKZ
$P_1, P_2, \dots$	Key derivation parameter octet strings
q	The size of the field used (part of the domain parameters)
r	The prime order of the desired group for most methods in this document (part of the domain parameters)
$Redp$	An REDP
s	Private key, integer, corresponding to public key w
S_C	A signature used in APKAS-PAKZ key confirmation
$Sign$	A signature generation operation in signature scheme S_s
S_s	A signature scheme used in APKAS-PAKZ
$U_{K\pi}$	An octet string representing a hidden and password-masked private signature key u used in APKAS-PAKZ key confirmation
U_π	An octet string representing a password-masked private signature key u used in APKAS-PAKZ
u	Private key, integer, corresponding to public key v
u_π	A password-limited private key corresponding to v_π , derived from a function of π
$uLen$	The length in octets of an agreed octet string representation of a private key u in signature scheme S_s
v	Public key corresponding to private key u
v_π	A password-limited public key corresponding to u_π , used as password verification data
$Verify$	A signature verification operation in signature scheme S_s
w	Public key corresponding to private key s
w'	Another party's public key
w_C	Client's public key

w_S	Server's public key
x	Field element
y	Field element
z	Shared secret value, field element in $GF(q)$, derived by SVDPs
Z, Z_1, Z_2	Shared secret values, octet strings, in key agreement operations
z_1, z_2	Shared secret field elements
z_g	Established shared secret group element in a PKAS; established secret group element in a PKRS

8.2 Primitives

Subclause 4.4 describes the types of primitives defined in this document. This subclause defines primitives for use in both the DL and EC settings.

8.2.1 KRBP-1

{DL,EC}KRBP-1 is {discrete logarithm, elliptic curve} key retrieval blinding primitive, version 1. KRBP-1 is based on the work of Ford and Kaliski [B16]. It derives a blinded password from a password-derived generator and the Client's private key, using {DL,EC} domain parameters. This primitive is used by the scheme {DL,EC}PKRS-1-CLIENT.

Input:

- The Client's private key integer s
- The Client's password-derived generator group element g_π
- The {DL,EC} domain parameters (including g and r) associated with s and g_π

Assumptions: Private key s and associated domain parameters are valid. g_π is a generator of the desired group of order r .

Output: The computed blinded password w_C , a valid public key.

Operation: The blinded password w_C shall be computed by the following or an equivalent sequence of steps:

- a) Compute group element $w_C = g_\pi^s$
- b) Output w_C as the blinded password

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTE—KRBP-1 performs the inverse function of KRUP-1.

8.2.2 KRPP-1

{DL,EC}KRPP-1 is {discrete logarithm, elliptic curve} key retrieval permutation primitive, version 1. KRPP-1 is based on the work of Ford and Kaliski [B16]. This primitive derives a permuted blinded password from a party's private key and a blinded password using {DL,EC} domain parameters. This primitive is used by the scheme {DL,EC}PKRS-1-SERVER.

Input:

- The Server's own private key integer u
- The Client's blinded password group element w_C
- The {DL,EC} domain parameters (including r) associated with the keys u and w_C

Assumptions: Private key u and associated domain parameters are valid. w_C is an element of the parent group.

Output: The computed permuted blinded password w_S , a valid public key.

Operation: The permuted blinded password w_S shall be computed by the following or an equivalent sequence of steps:

- a) Compute group element $w_S = w_C ^ u$
- b) Output w_S as the permuted blinded password.

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

8.2.3 KRUP-1

{DL,EC}KRUP-1 is {discrete logarithm, elliptic curve} key retrieval unblinding primitive, version 1. KRUP-1 is based on the work of Ford and Kaliski [B16]. This primitive derives a permuted password value from both the Client's own private key and a Server's blinded permuted password value using {DL,EC} domain parameters. This primitive is used by the scheme {DL,EC}PKRS-1-CLIENT.

Input:

- The Client's own private key integer s
- The Server's permuted blinded password group element w_S
- The {DL,EC} domain parameters (including r) associated with the keys s and w_S

Assumptions: Private key s and associated domain parameters are valid. w_S is an element of the parent group.

Output: The permuted password value z_g , an element of the parent group.

Operation: The permuted password value z_g shall be computed by the following or an equivalent sequence of steps:

- a) Compute group element $z_g = w_S ^{(s^{-1} \bmod r)}$
- b) Output z_g as the permuted password.

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTE—KRUP-1 performs the inverse function of KRBP-1.

8.2.4 PEPKGP-AMP-SERVER

{DL,EC}PEPKGP-AMP-SERVER is {discrete logarithm, elliptic curve} password-entangled public-key generation primitive, version AMP for Server. PEPKGP-AMP-SERVER is based on the work of Kwon [B38]. This primitive derives a password-entangled public key from password verification data, the Server's private key, and a Client's public key, using {DL,EC} domain parameters.

This primitive is parameterized by the following choices:

- A hash function *HashWC* (see NOTE), which should be one of the hash functions in 12.1.
- An octet string o_{ID} that may provide additional input to *HashWC*.

The Client and Server shall use the same *HashWC* and o_{ID} parameters with PEPKGP-AMP-SERVER and SVPD-AMP-CLIENT.

Input:

- The Server's private key s
- The password verification data v_{π} , a password-limited public key
- The Client's public key w_C
- The {DL,EC} domain parameters associated with the keys s , v_{π} , and w_C

Assumptions: Private key s , password-limited public key v_{π} , and associated domain parameters are valid; w_C is an element of the parent group.

Output: The derived password-entangled public key value w_S , which is an element of the parent group

Operation: The password-entangled public key value w_S shall be computed by the following or an equivalent sequence of steps:

- a) Compute $o_C = \text{GE2OSP-X}(w_C)$
- b) Compute $o_1 = \text{HashWC}(o_C \parallel o_{ID})$
- c) Compute $i_1 = \text{OS2IP}(o_1)$
- d) Compute $w_S = ((w_C \wedge i_1) * v_{\pi}) \wedge s$
- e) Output w_S

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTE—See D.2.2.1.6 for criteria for selecting *HashWC*.

8.2.5 PEPKGP-PAK

{DL,EC}PEPKGP-PAK is {discrete logarithm, elliptic curve} password-entangled public-key generation primitive, version PAK. It is based on the work of Boyko, MacKenzie, Patel [B8] and MacKenzie [B43]. This primitive derives a password-entangled public key from the party's private key and password value, using {DL,EC} domain parameters. This primitive is used by the schemes {DL,EC}BPKAS-PAK-CLIENT, {DL,EC}BPKAS-PPK-{CLIENT,SERVER} and {DL,EC}APKAS-PAKZ-CLIENT.

Input:

- The party's own private key s .

- The password-based mask group element π_m
- The {DL,EC} domain parameters (including g and r) associated with key s and group element π_m

Assumptions: Private key s and domain parameters are valid; group element π_m generates the desired group of order r .

Output: The derived password-entangled public key w

Operation: The password-entangled public key w shall be computed by the following or an equivalent sequence of steps:

- Compute a group element $w = (g \wedge s) * \pi_m$
- Output w as the password-entangled public key

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

8.2.6 PEPKGP-SPEKE

{DL,EC}PEPKGP-SPEKE is {discrete logarithm, elliptic curve} password-entangled public-key generation primitive, version SPEKE. It is based on the work of Jablon [B31]. This primitive derives a password-entangled public key from the party's private key and password value. This primitive is used in schemes BPKAS-SPEKE and APKAS-BSPEKE2.

Input:

- A private key integer s
- A password-derived generator group element g_π
- The {DL,EC} domain parameters associated with g_π and s

Assumptions: Private key s and associated input domain parameters are valid. g_π generates the desired group of order r .

Output: The derived password-entangled public key w

Operation: The public key w shall be computed by the following or an equivalent sequence of steps:

- Compute $w = g_\pi \wedge s$
- Output w

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

8.2.7 [DL] PEPKGP-SRP3-SERVER, PEPKGP-SRP6-SERVER

DLPEPKGP-{SRP3,SRP6}-SERVER is discrete logarithm password-entangled public-key generation primitive, version {SRP3,SRP6} for Server. This primitive derives a password-entangled public key from the Server's private key and password verification data using DL domain parameters. This primitive is used in DLAPKAS-{SRP3,SRP6}-SERVER. APKAS-SRP3 is based on the work of Wu [B57] and IETF RFC 2945 [B24]. APKAS-SRP6 is based on the work of Wu [B57] and [B56].

For SRP6 only: This primitive is parameterized by the following choices:

- A multiplier value creation function $Mvcf$, which should be MVCF-DP (see 12.5.1). The Client shall use the same $Mvcf$ and its associated parameters with DLSVDP-SRP6-CLIENT.

Input:

- The Server's own private key s
- The password verification data v_π , a password-limited public key
- The DL domain parameters (including g_{q-1} and q) associated with the key s and verification data v_π

Assumptions: Private key s , password-limited public key v_π , and DL domain parameters are valid; g_{q-1} is of order $q-1$.

Output: The computed password-entangled public-key value w_S , which is a field element in $GF(q)$

Operation: The password-entangled public-key value w_S shall be computed by the following or an equivalent sequence of steps:

- Compute field element $x = \exp(g_{q-1}, s)$
- Compute field element m_v as follows:
 - For SRP3 only: Let $m_v = 1$
 - For SRP6 only: Compute m_v using $Mvcf$
- Compute $w_S = (v_\pi \times m_v + x)$
- Output w_S

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

8.2.8 [EC] PEPKGP-SRP5-SERVER

ECPEPKGP-SRP5-SERVER is elliptic curve password-entangled public-key generation primitive, version SRP5. SRP5 is based on the work of Wang [B55] and Wu [B57]. This primitive derives a password-entangled public key from the Server's private key and password verification data, using EC domain parameters. This primitive is used by the scheme ECAPKAS-SRP5-SERVER.

This primitive is parameterized by the following choice:

- A REDP function $Redp$, which should be ECREDP-1 or ECREDP-2. The Client shall use the same $Redp$ parameter with SVDP-SRP5-CLIENT.

Input:

- The Server's own private key s
- The password verification data v_π , a password-limited public key
- The EC domain parameters (including g) associated with key s and public key v_π

Assumptions: Private key s , public key v_π , and associated domain parameters are valid.

Output: The derived password-entangled public key w_S

Operation: The password-entangled public key w_S shall be computed by the following or an equivalent sequence of steps:

- a) Compute octet string $o_1 = \text{GE2OSP-X}(v_\pi)$
- b) Compute group element $e_1 = \text{Redp}(o_1)$
- c) Compute group element $w_S = (g \wedge s) * e_1$
- d) Output w_S as the password-entangled public key

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

8.2.9 PKGP-DH

{DL,EC}PKGP-DH is {discrete logarithm, elliptic curve} public-key generation primitive, Diffie-Hellman version. PKGP-DH is based on the work of Diffie and Hellman [B10] and IEEE Std 1363-2000. This primitive derives a {DL,EC} public key from the party's {DL,EC} private key using {DL,EC} domain parameters. This primitive is used by the schemes {DL,EC}BPKAS-PAK-SERVER, {DL,EC}APKAS-AMP-CLIENT and ECAPKAS-SRP5-CLIENT.

Input:

- The party's own private key integer s
- The {DL,EC} domain parameters (including g and r) associated with the key s

Assumptions: Private key s and associated domain parameters are valid.

Output: The computed public key w

Operation: The public key w shall be computed by the following or an equivalent sequence of steps:

- a) Compute public-key group element $w = g \wedge s$
- b) Output w as the public key

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

8.2.10 [DL] PKGP-SRP-CLIENT

DLPKGP-SRP-CLIENT is discrete logarithm public-key generation primitive, version SRP for Client. It is based on the work of Wu [B57] and IETF RFC 2945 [B24]. This primitive derives a public key from the Client's private key using DL domain parameters. This primitive is used by the schemes DLAPKAS-{SRP3,SRP6}-CLIENT.

Input:

- The Client's own private key integer s
- The DL domain parameters (including q and g_{q-1}) associated with the key s

Assumptions: Private key s and DL domain parameters are valid; g_{q-1} is of order $q-1$.

Output: The computed public key w_C , which is a non-zero field element of $GF(q)$

Operation: The public key w_C shall be computed by the following or an equivalent sequence of steps:

- a) Compute a DL public-key field element $w_C = \exp(g_{q-1}, s)$
- b) Output w_C

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTE—DLPKGP-SRP-CLIENT is similar to DLPKGP-DH, except that DLPKGP-DH uses the generator g of order r , instead of g_{q-1} .

8.2.11 PVDGP-AMP

{DL,EC}PVDGP-AMP is {discrete logarithm, elliptic curve} password verification data generation primitive, version AMP. PVDGP-AMP is based on the work of Kwon [B38]. This primitive derives password verification data and associated values from a party's password value, using {DL,EC} domain parameters. This primitive derives the password-limited private key used in {DL,EC}APKAS-AMP-CLIENT and derives the password verification data used in {DL,EC}APKAS-AMP-SERVER.

This primitive is parameterized by the following choices:

- A hash function *HashPVD* (see NOTE), which should be one of the hash functions in 12.1 or MGF1 (see 12.2.1)

Input:

- The password-based octet string π
- The {DL,EC} domain parameters (including g and r) associated with π

Assumptions: Domain parameters are valid.

Output: The password-limited private key u_π and associated password verification data consisting of password-limited public key v_π

Operation: The password-limited private key u_π and verification data v_π shall be computed by the following or an equivalent sequence of steps:

- a) Compute $o_\pi = \text{HashPVD}(\pi)$
- b) Compute $u_\pi = \text{OS2IP}(o_\pi) \bmod r$
- c) Compute $v_\pi = g^{u_\pi}$
- d) Output u_π and verification data v_π

Conformance region recommendation: A conformance region should include limitations for any input values as discussed in Annex B.

NOTE—See D.2.1.17 for discussion of appropriate choices for *HashPVD*.

8.2.12 PVDGP-BSPEKE2

{DL,EC}PVDGP-BSPEKE2 is {discrete logarithm, elliptic curve} password verification data generation primitive, version BSPEKE2. It is based on the work of Jablon [B27] and [B28]. This primitive derives password verification data and associated values from a party's password, using {DL,EC} domain parameters. This primitive derives the password-limited private key used in {DL,EC}APKAS-BSPEKE2-CLIENT and {DL,EC}APKAS-WSPEKE-CLIENT and derives the associated password verification data used in {DL,EC}BPKAS-BSPEKE2-SERVER and {DL,EC}BPKAS-WSPEKE-SERVER.