INTERNATIONAL STANDARD

ISO/IEC 18367

First edition
2016-12-15

Information technology — Security techniques — Cryptographic algorithms and security mechanisms conformance testing

Technologie de l'information — Techniques de sécurité — Essais de conformité des algorithmes cryptographiques et des mécanismes de sécurité

Reference number
ISO/IEC 18367:2016(E)

Documents
Document Sharing Hub
© ISO/IEC 2016

Full document details: ISO IEC 18367-2016



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2016, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Documents
Document Sharing Hub

© ISO/IEC 2016 – All rights reserved

Full document details: ISO IEC 18367-2016

Contents

Page

Foreword ..... **v**
Introduction ..... **vi**
1 Scope **1**
2 Normative references **1**
3 Terms and definitions **1**
4 Symbols and abbreviated terms **6**
5 Objectives **7**
6 Types of cryptographic algorithms and security mechanisms from a conformance testing perspective **8**
6.1 General 8
6.2 Asymmetric key algorithms 8
6.3 Digital signature 8
6.4 Digital signature with message recovery 8
6.5 Hashing algorithms 8
6.6 Key establishment mechanisms 8
6.7 Lightweight cryptography 9
6.8 Message authentication algorithms 9
6.9 Random bit generator algorithms 9
6.9.1 Deterministic random bit generator algorithms 9
6.9.2 Non-deterministic random bit generator algorithms 9
6.10 Symmetric key algorithms 10
6.10.1 Block cipher symmetric key algorithms 10
6.10.2 Stream cipher symmetric key algorithms 10
7 Conformance testing methodologies **10**
7.1 Overview 10
7.2 Black box testing 11
7.2.1 General 11
7.2.2 Known-answer test vectors 11
7.2.3 Multi-block message testing 11
7.2.4 Monte Carlo or statistical testing 11
7.3 Glass box or white box testing 11
7.3.1 Source code inspection 11
7.3.2 Binary analysis 11
8 Levels of conformance testing **12**
8.1 Introduction 12
8.2 Level of basic conformance testing 12
8.3 Level of moderate conformance 12
9 Conformance testing guidelines **12**
9.1 General guidelines 12
9.1.1 Identification 12
9.1.2 Guidelines for black box testing 13
9.1.3 Guidelines for white box testing 13
9.2 Guidelines specific to encryption algorithms 16
9.2.1 Identification of encryption algorithms 16
9.2.2 Selecting a set of conformance test items 17
9.2.3 Guidelines for each conformance test item 18
9.3 Guidelines specific to digital signature algorithms 29
9.3.1 Identification of digital signature algorithms 29
9.3.2 Selecting a set of conformance test items 29
9.3.3 Guidelines for each conformance test item 29
9.4 Guidelines specific to hashing algorithms 30

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

9.4.1 Identification of hashing algorithms 30
9.4.2 Selecting a set of conformance test items 31
9.4.3 Guidelines for each conformance test item 31
9.5 Guidelines specific to MAC algorithms 33
9.5.1 Identification of MAC algorithms 33
9.5.2 Selecting a set of conformance test items 34
9.5.3 Guidelines for each conformance test item 34
9.6 Guidelines specific to RBG algorithms 35
9.6.1 Identification of RBG algorithms 35
9.6.2 Selecting a set of conformance test items 35
9.6.3 Guidelines for each conformance test item 35
9.7 Guidelines specific to key establishment mechanisms 36
9.7.1 Identification of key establishment mechanisms 36
9.7.2 Selecting a set of conformance test items 36
9.7.3 Guidelines for each conformance test item 37
9.8 Guidelines specific to key derivation function 39
9.8.1 Identification of key derivation function 39
9.8.2 Selecting a set of conformance test items 39
9.8.3 Guidelines for each conformance test item 39
9.9 Guidelines specific to prime number generation 40
9.9.1 Identification of prime number generation 40
9.9.2 Selecting a set of conformance test items 40
9.9.3 Guidelines for each conformance test item 41

10 Conformance testing 41
10.1 Level of conformance testing 41
10.2 Symmetric key cryptographic algorithms 42
10.2.1 n-bit block cipher 42
10.3 Asymmetric key cryptographic algorithms 43
10.3.1 Digital Signature Algorithm (DSA) 43
10.3.2 RSA 47
10.3.3 Elliptic Curve Digital Signature Algorithm (ECDSA) 49
10.4 Dedicated hashing algorithms 51
10.4.1 General 51
10.4.2 Black box testing 51
10.4.3 White box testing 51
10.5 Message Authentication Codes (MAC) 51
10.5.1 Black box testing 51
10.5.2 White box testing 52
10.6 Authenticated encryption 53
10.6.1 Black box testing 53
10.6.2 White box testing 54
10.7 Deterministic Random Bit Generation algorithms 54
10.7.1 DRBG based on ISO/IEC 18031 54
10.8 Key agreement 58
10.8.1 Black box testing 58
10.8.2 White box testing 61
10.9 Key Derivation Functions (KDF) 62
10.9.1 Black box testing 62
10.9.2 White box testing 63

Annex A (informative) **Common mistakes in cryptographic algorithm implementations 64**

Annex B (informative) **Examples of known-answer test vectors 65**

Bibliography 66

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 27, *IT Security techniques.*

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

Introduction

This document describes cryptographic algorithms and security mechanisms conformance testing methods.

The purpose of this document is to address conformance testing methods of cryptographic algorithms and security mechanisms implemented in a cryptographic module. This will allow a complete security evaluation of both the cryptographic module and the implemented cryptographic algorithms and security mechanisms.

This document is related to ISO/IEC 19790 and ISO/IEC 24759. ISO/IEC 19790 specifies the security requirements for cryptographic modules. At a minimum, a cryptographic module implements at least one approved security function (i.e., cryptographic algorithm or security mechanism). ISO/IEC 24759 addresses the test requirements for each of the security requirements in ISO/IEC 19790. However, ISO/IEC 24759 does not address test methods for cryptographic algorithms and security mechanisms conformance testing.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

Information technology — Security techniques — Cryptographic algorithms and security mechanisms conformance testing

1 Scope

This document gives guidelines for cryptographic algorithms and security mechanisms conformance testing methods.

Conformance testing assures that an implementation of a cryptographic algorithm or security mechanism is correct whether implemented in hardware, software or firmware. It also confirms that it runs correctly in a specific operating environment. Testing can consist of known-answer or Monte Carlo testing, or a combination of test methods. Testing can be performed on the actual implementation or modelled in a simulation environment.

This document does not include the efficiency of the algorithms or security mechanisms nor the intrinsic performance. This document focuses on the correctness of the implementation.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14888-3:2016, *Information technology — Security techniques — Digital signatures with appendix*

ISO/IEC 19790:2012, *Information technology — Security techniques — Security requirements for cryptographic modules*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19790 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1
approval authority
any national or international organisation/authority mandated to approve and/or evaluate security functions

Note 1 to entry: An approval authority in the context of this definition evaluates and approves security functions based on their cryptographic or mathematical merits but is not the testing entity which would test for conformance to this document.

[SOURCE: ISO/IEC 19790:2012, 3.4]

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

3.2
approved mode of operation
set of services which includes non-security relevant services and at least one service that utilizes an approved security function or process

Note 1 to entry: Not to be confused with a specific mode of an approved security function, e.g. Cipher Block Chaining (CBC) mode.

Note 2 to entry: Non-approved security functions or processes are excluded.

[SOURCE: ISO/IEC 19790:2012, 3.7]

3.3
approved security function
security function (e.g. cryptographic algorithm) approved by an approval authority

3.4
black box
idealized mechanism that accepts inputs and produces outputs, but is designed such that an observer cannot see inside the box or determine exactly what is happening inside that box

Note 1 to entry: This term can be contrasted with *glass box* (3.12).

[SOURCE: ISO/IEC 18031:2011, 3.6]

3.5
critical security parameter
CSP
security-related information whose disclosure or modification can compromise the security of a cryptographic module

EXAMPLE Secret and private cryptographic keys, authentication data such as passwords, PINs, certificates or other trust anchors.

[SOURCE: ISO/IEC 19790:2012, 3.18, modified]

3.6
cryptographic algorithm
well-defined computational procedure that takes variable inputs, which may include cryptographic keys, and produces an output

[SOURCE: ISO/IEC 19790:2012, 3.20]

3.7
cryptographic algorithm boundary
boundary encompassing the complete cryptographic algorithm implementation

3.8
cryptographic boundary
explicitly defined continuous perimeter that establishes the physical and/or logical bounds of a cryptographic module and contains all the hardware, software, and/or firmware components of a cryptographic module

[SOURCE: ISO/IEC 19790:2012, 3.21, modified]

3.9
firmware
executable code of a cryptographic module that is stored in hardware within the cryptographic boundary and cannot be dynamically written or modified during execution while operating in a non-modifiable or limited operational environment

EXAMPLE Storage hardware can include but is not limited to PROM, EEPROM, FLASH, solid state memory, hard drives, etc.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

[SOURCE: ISO/IEC 19790:2012, 3.45]

3.10
functional specification
high-level description of the ports and interfaces visible to the operator and high-level description of the behaviour of the IUT

Note 1 to entry: Here, the IUT means a cryptographic algorithm implementation (see 3.13).

[SOURCE: ISO/IEC 19790:2012, 3.47, modified]

3.11
functional testing
testing of the IUT functionality as defined by the *functional specification* (3.10)

[SOURCE: ISO/IEC 19790:2012, 3.48, modified]

3.12
glass box
idealized mechanism that accepts inputs and produces outputs and is designed such that an observer can see inside and determine exactly what is going on

Note 1 to entry: This term can be contrasted with black box (3.4).

[SOURCE: ISO/IEC 18031:2011, 3.14]

3.13
implementation under test
IUT
implementation which is tested for conformance to the selected cryptographic algorithm or security mechanism standard

3.14
independent verification test
test verifying the conformance for algorithms where their outputs are non-deterministic (or randomized) for defined input vectors in an independent way, instead of literally following the algorithm steps

Note 1 to entry: The signature generation function of DSA involves per-message secret number internally, and therefore, the resultant signature cannot be derived from the input vectors in a deterministic way without the knowledge of per-message secret number. Here, the signature verification function can be used to verify the relation between the public key, message and resultant signature, without knowing the per-message secret number.

Note 2 to entry: Independent verification tests can involve the reference implementation of the inverse function of the selected cryptographic algorithm, e.g. using the digital signature verification function to verify the corresponding signature generation function.

Note 3 to entry: In contrast to the other KATs, the independent verification test does not prepare expected values in advance.

EXAMPLE Applying Miller-Rabin primality test to verifying prime number generation functions, independent of the implementation details of the IUT.

3.15
key agreement
process of establishing a shared secret key between entities in such a way that neither of them can predetermine the value of that key

[SOURCE: ISO/IEC 11770-3:2015, 3.18, modified]

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

3.16
key derivation function
KDF
function that outputs one or more shared secrets, for use as keys, given shared secrets and other mutually known parameters as input

[SOURCE: ISO/IEC 11770-3:2015, 3.22]

3.17
key derivation key
key that is used as input to the key expansion function to derive other keys

3.18
key establishment
process of making available a shared secret key to one or more entities, where the process includes key agreement and key transport

[SOURCE: ISO/IEC 11770-3:2015, 3.23]

3.19
key expansion function
function which takes as input a number of parameters, at least one of which is a MAC algorithm key, and which gives as output keys appropriate for the intended algorithm and application, and which has the property that it is computationally infeasible to deduce either the output without prior knowledge of the secret input or the secret input from the output

3.20
key extraction function
function which takes as input a number of parameters, at least one of which is secret, which gives as output a MAC algorithm key for use as input to a key expansion function, and which has the property that it is computationally infeasible to deduce either the output without prior knowledge of the secret input or the secret input from the output

3.21
keying material
data necessary to establish and maintain cryptographic keying relationships

EXAMPLE Keys, initialization values.

[SOURCE: ISO/IEC 11770-1:2010, 2.27]

3.22
key management
administration and use of the generation, registration, certification, deregistration, distribution, installation, storage, archiving, revocation, derivation and destruction of keying material in accordance with a security policy

[SOURCE: ISO/IEC 19790:2012, 3.64]

3.23
key transport
process of transferring a key from one entity to another entity, suitably protected

[SOURCE: ISO/IEC 11770-3:2015, 3.25]

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

3.24
known-answer test
KAT
method of testing a deterministic mechanism where a given input is processed by the mechanism and the resulting output is then compared to a corresponding known value

Note 1 to entry: The known-answer tests are designed to test the conformance of the implementation under test (IUT) to the various specifications of the referenced cryptographic algorithm.

Note 2 to entry: A known-answer test is considered as a kind of black box testing.

[SOURCE: ISO/IEC 18031:2011, 3.21, modified]

3.25
Monte Carlo test
MCT
subset of known-answer test utilising randomly generated input vectors and the corresponding known output result, designed to pseudo exhaust the presence of flaws by exercising the entire IUT in a manner that cannot be detected with the controlled input vectors

Note 1 to entry: The types of implementation flaws which can be detected by Monte Carlo tests include pointer problems, insufficient allocation of space, improper error handling and incorrect behaviour of the IUT.

3.26
multi-block message test
MMT
set of tests designed to test the ability of the implementation to process multi-block messages which will require the chaining of information from one block to the next

3.27
public security parameter
PSP
security-related public information whose modification can compromise the security of a cryptographic module

EXAMPLE Public cryptographic keys, public key certificates, self-signed certificates, trust anchors, one-time passwords associated with a counter and internally held date and time.

[SOURCE: ISO/IEC 19790:2012, 3.99, modified]

3.28
random bit generator
RBG
device or algorithm that outputs a sequence of bits that appears to be statistically independent and unbiased

[SOURCE: ISO/IEC 18031:2011, 3.29]

3.29
salt
random data item produced by the signing entity during the generation of message representative

Note 1 to entry: Also known as the randomizer in ISO/IEC 14888-3.

Note 2 to entry: Also known as the per-message secret number in Reference [16].

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

3.30
security function
cryptographic algorithms together with modes of operation, such as block ciphers, stream ciphers, symmetric or asymmetric key algorithms, message authentication codes, hash functions or other security functions, random bit generators, entity authentication and SSP generation and establishment all approved either by ISO/IEC or an approval authority

[SOURCE: ISO/IEC 19790:2012, 3.106]

3.31
sensitive security parameter
SSP
critical security parameter (CSP) or public security parameter (PSP)

3.32
shared secret key
key which is shared with all the active entities via a key establishment mechanism for multiple entities

Note 1 to entry: Also known as "shared secret."

[SOURCE: ISO/IEC 11770-5:2011, 3.28, modified]

3.33
simulation
exercise of source code (e.g. VHDL code) prior to physical entry into the module (e.g. an FPGA or custom ASIC)

Note 1 to entry: The behaviour of the source code within the simulator can be logically identical when placed into the module or instantiated into logic gates. However, many other variables exist that can alter the actual behaviour (e.g., path delays, transformation errors, noise, environmental, etc.). It is not guaranteed that the actual behaviour of the IUT is identical, as many other variables cannot be identified with certainty.

3.34
zeroisation
method of destruction of stored data and unprotected SSPs to prevent retrieval and reuse

[SOURCE: ISO/IEC 19790:2012, 3.134]

4 Symbols and abbreviated terms

AES	advanced encryption standard
ASN.1	abstract syntax notation one
CBC	cipher block chaining mode of operation
CCM	counter mode with cipher block chaining-message authentication code
CFB	cipher feedback mode of operation
CMAC	cipher-based message authentication code
CSP	critical security parameter
DER	distinguished encoding rules
DRBG	deterministic random bit generator
DSA	digital signature algorithm

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

EAL	evaluation assurance level
ECB	electronic codebook mode of operation
ECDSA	elliptic-curve digital signature algorithm
FFC	finite field cryptography
GCM	galois/counter mode
GMAC	GCM message authentication code
HMAC	keyed-hash message authentication code
IUT	implementation under test
KAS	key agreement scheme
KAT	known-answer test
KC	key confirmation
KDF	key derivation function
MAC	message authentication code
MCT	Monte Carlo test
MMT	multi-block message test
OFB	output feedback mode of operation
PKCS	public key cryptography standards
PSS	probabilistic signature scheme
RBG	random bit generator
RSA	algorithm developed by Rivest, Shamir and Adleman
SHA	secure hash algorithm
TDEA	triple data encryption algorithm

$\lceil X \rceil$	ceiling: the smallest integer greater than or equal to X. For example, $\lceil 5 \rceil = 5$ and $\lceil 5.3 \rceil = 6$
$X \oplus Y$	bitwise exclusive-or (also bitwise addition mod 2) of two bit strings X and Y of the same length
X || Y	concatenation of two bit strings X and Y in that order
x mod n	unique remainder r, $0 \leq r \leq n-1$, when integer x is divided by n. For example, 23 mod 7 = 2

5 Objectives

The requirements specified in this document are derived from the following objectives for cryptographic algorithm implementations to

— provide assurance that the cryptographic algorithm implementation adheres to the specifications detailed in the associated cryptographic standard, and

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

— detect implementation non-conformities made by implementers by testing the algorithm's specifications, components, features and/or functionality for correctness and completeness.

6 Types of cryptographic algorithms and security mechanisms from a conformance testing perspective

6.1 General

This document will address approved security functions from a conformance testing point perspective. In particular, this document will address those defined in Clause 2. It will include within its scope the associated security mechanisms of the cryptographic algorithms or the security mechanisms. The considered implementations can be software, firmware, hardware or a combination thereof.

6.2 Asymmetric key algorithms

This subclause describes the different types of asymmetric key algorithms.

Asymmetric key algorithms consist of asymmetric key cryptographic primitive(s) and supporting functions. Some of asymmetric key algorithms are non-deterministic, due to salt or internally generated random numbers. The implementation of asymmetric key algorithms would contain more conditional branches than symmetric key algorithms. In considering these aspects of asymmetric key algorithms, the known-answer test (see 7.2.2) and/or independent verification test (see 3.14) are applicable.

In addition to these testing methodologies, other conformance testing methodologies (e.g. source code inspection) are also applicable.

6.3 Digital signature

This subclause describes the different types of digital signature algorithms.

The same perspective as asymmetric key algorithms is still applicable to digital signature algorithms.

6.4 Digital signature with message recovery

This subclause describes the different types of digital signature with message recovery algorithms.

The same perspective as asymmetric key algorithms is still applicable to digital signature with message recovery.

6.5 Hashing algorithms

This subclause describes the different types of hashing algorithms.

Hashing algorithms will be dedicated hash functions, functions based on block cipher algorithms or functions based on modular arithmetic. In considering the nature of hashing algorithms, the known-answer test (see 7.2.2) and Monte Carlo test (see 7.2.4) are applicable. Other conformance testing methodologies (e.g. source code inspection) are also applicable.

6.6 Key establishment mechanisms

This subclause describes the different types of key establishment mechanisms.

Key establishment mechanisms consist of asymmetric/symmetric key cryptographic primitive(s) and supporting functions. Supporting functions can be hashing algorithms, random bit generation, asymmetric key pair generation function, and public key validation function.

In considering the complex nature of key establishment mechanisms, the known-answer test (see 7.2.2) and independent verification test (see 3.14) are applicable. As a prerequisite for this conformance

© ISO/IEC 2016 – All rights reserved
AVA Documents Document Sharing Hub
Full document details: ISO IEC 18367-2016

testing, underlying algorithm implementations have passed the conformance testing elsewhere in this document.

Note that some input parameters are transmitted through a communication channel. It should be tested through the conformance testing that an IUT has an ability to distinguish valid parameters with invalid parameters.

In addition to these testing methodologies, other conformance testing methodologies (e.g. source code inspection) are also applicable.

6.7 Lightweight cryptography

This subclause describes the different types of lightweight cryptography algorithms.

Lightweight cryptography includes asymmetric key algorithms, block ciphers and stream ciphers. So, the applicable perspective is the same as one for asymmetric key algorithms, block ciphers and stream ciphers.

6.8 Message authentication algorithms

This subclause describes the different types of message authentication algorithms.

Message authentication algorithms can consist of underlying algorithms (e.g. block cipher algorithms, hash algorithms).

In considering this aspect of message authentication algorithms, the known-answer test (see 7.2.2) is applicable. As a prerequisite for this conformance testing, underlying algorithm implementations have passed the conformance testing elsewhere in this document.

In addition to this testing methodology, other conformance testing methodologies (e.g. source code inspection) are applicable.

6.9 Random bit generator algorithms

6.9.1 Deterministic random bit generator algorithms

This subclause describes the different types of deterministic random bit generator algorithms.

Deterministic random bit generators can consist of underlying algorithms (e.g. block cipher algorithms, hash algorithms).

In considering this aspect of deterministic random bit generator, the known-answer test (see 7.2.2) is applicable. As a prerequisite for this conformance testing, underlying algorithm implementations have passed the conformance testing elsewhere in this document.

In addition to this testing methodology, other conformance testing methodologies (e.g. source code inspection) are also applicable.

6.9.2 Non-deterministic random bit generator algorithms

This subclause describes the different types of non-deterministic random bit generator algorithms.

Currently, there is no standard specification of NRBG in ISO/IEC 18031, so the associated conformance testing methodologies are not specified in this document.

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

6.10 Symmetric key algorithms

6.10.1 Block cipher symmetric key algorithms

This subclause describes the different types of block cipher symmetric key algorithms.

Block cipher symmetric key algorithms support fixed input block and multiple input blocks with block cipher modes of operation. Block cipher symmetric key algorithms are used for various purposes and, therefore, the correctness of the algorithm implementations is important. In considering these aspects, known-answer tests (see 7.2.2), multi-block message test (see 7.2.3) and Monte Carlo test (see 7.2.4) are applicable. In addition to these testing methodologies, other conformance testing methodologies are also applicable.

6.10.2 Stream cipher symmetric key algorithms

This subclause describes the different types of stream cipher symmetric key algorithms.

Stream cipher symmetric key algorithms are similar to block cipher symmetric key algorithms in their structures.

In considering this aspect of stream cipher algorithms, known-answer tests (see 7.2.2) and Monte Carlo test (see 7.2.4) are applicable.

In addition to these testing methodologies, other conformance testing methodologies are also applicable.

7 Conformance testing methodologies

7.1 Overview

For cryptographic algorithms which do not introduce randomness, known-answer test(s), Monte Carlo and multi-block message tests shall be applied. For cryptographic algorithms which utilize internally generated random numbers to generate output, independent verification test(s) shall be applied.

EXAMPLE Digital signature scheme 2 in ISO/IEC 9796-2.

Note that neither the known-answer test, Monte Carlo test, multi-block message test, nor the independent verification test is universal. In order to complement these tests, source code review may be applied.

Conformance testing provides tests to determine the correctness of the algorithm implementation and implementation errors including pointer problems, insufficient allocation of space, improper error handling and incorrect behaviour of the algorithm implementation. The validation tests are designed to assist in the detection of accidental implementation errors and are not designed to detect intentional attempts to misrepresent conformance. Conformance testing should not be interpreted as an evaluation or endorsement of overall product security. Conformance testing shall utilize statistical sampling (i.e. only a small number of the possible cases are tested); hence, the successful conformance testing of a device does not imply 100 % correctness of conformance.

NOTE 1 Implementation errors in hardware implementations include timing delay.

NOTE 2 Implementation errors in software/firmware implementations include pointer problems, insufficient allocation of space, improper error handling and incorrect behaviour of the algorithm implementation.

The different types of testing methodologies are described in 7.2 and 7.3.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

7.2 Black box testing

7.2.1 General

Black box validation testing is where the functionality of an implementation is examined without peering into the implementation's actual code or internal workings. The given inputs exercise the implementation to assure the specifications of the standard are implemented correctly. This is determined by comparing the output produced by the implementation under test (IUT) with the expected outputs. Black box testing can be used when a reference implementation or simulator is available.

7.2.2 Known-answer test vectors

The known-answer tests are designed to verify the individual components of the specifications of the cryptographic algorithm. The tests exercise each bit of every component of the algorithm implementation. Known-answer tests use specific pre-determined input test vectors and the computation of the expected results to target the specific functionality of a security function.

7.2.3 Multi-block message testing

The Multi-block message test (MMT) is designed to test the ability of an implementation to process multi-block messages, which may require chaining of information from one block to the next.

7.2.4 Monte Carlo or statistical testing

Monte Carlo testing is a statistical method which is designed to exercise the entire implementation of the cryptographic algorithm being implemented. The purpose is to detect the presence of flaws in the IUT that were not detected with the controlled input of the known-answer test. The Monte Carlo test does not guarantee ultimate reliability of the IUT that implements the algorithm (e.g. hardware failure, software corruption, etc.). A predetermined number of pseudorandom values are used as input values to test the algorithmic implementation. Using these values, the IUT is exercised through the complete implementation. The results are then compared with the expected values.

7.3 Glass box or white box testing

7.3.1 Source code inspection

7.3.1.1 Overview

In contrast to black box testing such as KAT, source code inspection is capable of wide application, which will be useful in verifying the internal behaviour and rare case handling. However, it needs knowledge of description language and reviewing time. So, the source code inspection should be complementary to other testing approaches.

7.3.2 Binary analysis

Even if the source code itself is consistent with the specification of a selected algorithm, the behaviour of resultant binary (or executable code) might be different from the original source code, e.g. due to the software building tools. In such a case, binary analysis should be performed in order to ensure the conformance. In the case of software or firmware implementations, this can be performed by inspecting the generated assembly output to assure it is consistent with the original source code (e.g. in Mixed Mode).

EXAMPLE Compiler, code interpreter.

NOTE Some cryptographic algorithm specifications include a step to zeroise secret information. However, such zeroisation steps might be deleted from the resultant binary due to the optimization or decision made by a compiler.

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

8 Levels of conformance testing

8.1 Introduction

The overview of two levels of conformance testing is described in 8.2 and 8.3.

8.2 Level of basic conformance testing

In this level, the conformity is tested by exercising IUTs from outside the cryptographic algorithm boundary. This level will commensurate with security level 1 in ISO/IEC 19790 or EAL3 in ISO/IEC 15408.

The tester is required to test the IUT by considering implementation errors listed in 7.1.

8.3 Level of moderate conformance

This level enhances the level of basic conformance by enforcing informal security policy. In this level, the conformity is tested by inspecting the IUTs also from inside the cryptographic algorithm boundary with focus on the security policy enforcement. This level will commensurate with security levels 2 and 3 in ISO/IEC 19790 or EAL4 in ISO/IEC 15408.

NOTE There can exist a level of conformance in which the formal security policy or formal verification is applied. However, this document does not define conformance tests covering this level.

9 Conformance testing guidelines

9.1 General guidelines

9.1.1 Identification

The tester shall identify the following information on the IUT:

a) tested platform or configuration;

EXAMPLE 1 Operational environment for software, including test harness.

EXAMPLE 2 When the virtualization technology is used for the operational environment, configuration parameters for the virtual machine.

The vendor shall identify the following information on the IUT:

b) reference to the specification of cryptographic algorithm implemented;

c) security function(s);

EXAMPLE 3 Digital signature generation function.

EXAMPLE 4 Encryption function of block ciphers.

EXAMPLE 5 Two-step key derivation function.

d) supporting functions;

EXAMPLE 6 Hash functions in digital signatures.

EXAMPLE 7 Entropy conditioning functions in random bit generators.

EXAMPLE 8 Public key validation function in key establishment scheme.

EXAMPLE 9 Key derivation functions in key establishment scheme.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

1) If the conformance testing for the supporting function is available, the tester shall verify that the supporting function implementation has passed the conformance testing.

e) supported parameters range;

EXAMPLE 10 Bit length of key of block cipher.

EXAMPLE 11 Bit length of key of MAC.

EXAMPLE 12 Bit length of input message for hash functions.

EXAMPLE 13 Bit length of input message to be encrypted using asymmetric ciphers.

f) data types and representation of parameters;

EXAMPLE 14 Most significant octet first.

EXAMPLE 15 Point compression of elliptic curve points.

1) The data types and format of input and output parameters shall be precise.

g) architecture or functional decomposition.

EXAMPLE 16 A digital signature algorithm implementation using a hashing algorithm implemented in firmware and a modular arithmetic coprocessor implemented in hardware.

EXAMPLE 17 An AES implementation using cryptographic supporting instructions.

9.1.2 Guidelines for black box testing

Some cryptographic algorithms use multiple-precision integer. Assuming that the multiple-precision integer is represented by octets, the carry bit handling will occur normally once in 256 test vectors. This also applies to the implementation of a counter. For such an implementation, at least 512 test vectors should be used to perform black box testing.

9.1.3 Guidelines for white box testing

9.1.3.1 Guidelines for source code inspection

9.1.3.1.1 General

Under the condition that the IUT is tested by black box testing and found to be functionally conformant, source code inspection may be applied to claim conformance of one level higher.

Source code inspection should be used to check the following aspects:

a) security feature which cannot be tested easily from external;

1) zeroisation,

EXAMPLE Zeroisation of shared secret key, ephemeral key, or subkeys

2) call of underlying cryptographic algorithm(s),

NOTE 1 SHA can be an underlying algorithm of HMAC.

NOTE 2 Random number generation becomes an underlying algorithm for DSA, ECDSA, and RSASSA-PSS signature generation function.

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

NOTE 3 If the IUT includes two or more implementations of the same cryptographic algorithm, it would be difficult to identify which implementation is used by black box testing from external. If such implementations are used as underlying algorithms, source code inspection enables us to identify which implementation is used. This also assures the chain of tested algorithm implementations.

3) health tests,

NOTE 4 Health tests are also called self-tests (see ISO/IEC 19790:2012, 3.108).

NOTE 5 There are health test requirements in ISO/IEC 18031.

b) branch condition;

c) status output;

d) source code which is rarely executed, but security relevant.

The tester should review the selected cryptographic algorithm specification to check whether the above examples are applicable to the IUT or not. If one or more examples are found to be applicable, the tester should apply source code inspection to verify that the IUT is conformant to the selected cryptographic algorithm specification. The tester shall not be misguided by source code annotation, if any.

Some IUTs support only limited input parameters range and/or limited functions, which are narrower than that of the original cryptographic algorithm specification. Such limited input parameters range and/or functions shall be recognised by the validation authority. The tester shall perform source code inspection by considering the claimed input parameters range, and verify that the source code is the correct embodiment of the cryptographic algorithm.

In the source code, many variables will be introduced. The tester should identify conversion between variables in the cryptographic algorithm specification and variables in the source code.

9.1.3.1.2 Security features which cannot be tested easily

9.1.3.1.2.1 Zeroisation

In general, there are several types of memory to which IUTs access, and IUTs can generate some copies of critical security parameters and/or their derivatives in the course of their operations.

The tester shall verify that all of the instances of CSPs and their derivatives are zeroised in both cases

a) when the cryptographic algorithm is completed successfully if they are no longer used, and

b) when the cryptographic algorithm terminates with an error indicator.

In order to perform source code inspection for zeroisation, the tester should identify types of memory on which any copies of critical security parameters or its derivatives reside. The tester should identify registers, variables, data structures, objects and/or pieces of memory which store any CSP.

EXAMPLE 1 Registers, cache and RAM.

EXAMPLE 2 The internal state of RBG.

NOTE 1 In ISO/IEC 19790:2012, 7.9.7, it is required to zeroise temporary SSPs when they are no longer needed, for security level 2 and higher.

The tester shall identify applied zeroisation techniques to the IUT. If the IUT is to be used in a cryptographic module, the tester shall verify that the applied zeroisation techniques meet sensitive security parameter zeroisation requirements in ISO/IEC 19790:2012, 7.9.7.

NOTE 2 The zeroisation requirements differ by security levels in ISO/IEC 19790.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

9.1.3.1.2.2 Call of underlying cryptographic algorithms

Some cryptographic algorithm specifications need underlying algorithms. The tester shall identify the underlying cryptographic algorithms by reviewing the cryptographic algorithm specification. The tester shall also identify the underlying cryptographic algorithm implementations by reviewing the source code. The tester shall verify, by performing source code inspection, that the underlying cryptographic algorithms implementation utilized meets the requirements of the cryptographic algorithm specification.

9.1.3.1.2.3 Health tests

Some cryptographic algorithm specifications request IUTs to implement and perform health tests.

The tester shall identify health test requirements from the selected specification and perform the source code inspection, in order to verify that the required health tests are really implemented. Some of the health test requirements can be purposely designed to test a) a specific part of cryptographic algorithm and/or b) a specific ability. Through the identification of selected cryptographic algorithms using guidelines in Clause 9, the tester shall verify that the heath test requirements are met in contrast to those selected cryptographic algorithm configurations.

However, the invocation of heath tests might not be a matter of IUTs, but the matter of cryptographic modules (see ISO/IEC 19790:2012, 7.10.3.2) or of security architecture under ISO/IEC 15408 (see ADV_ARC.1-5 in ISO/IEC 18045). For the latter case, through the cryptographic module testing, the tester should confirm that those heath tests are really invoked as specified in the selected cryptographic algorithm specification.

9.1.3.1.3 Branch condition

There will be branch conditions in cryptographic algorithm specifications. The tester shall identify targeted branch conditions by reviewing not only the cryptographic algorithm specification but also the source code.

NOTE 1 IUTs can include implementation-dependent branch conditions which are not identified in the cryptographic algorithm specification.

EXAMPLE Carry bit handling.

The branch condition will be expressed using a comparison operator. Note that the branch condition is not always directly translated into source code. The tester shall verify that the source code is arithmetically equivalent to the identified branch condition.

NOTE 2 Branch conditions are not always directly translated into source code. For example, if the statement is expressed as (A > 127) by integer A, another statement (A ≥ 128) is arithmetically equivalent.

9.1.3.1.4 Status output

Some cryptographic algorithms define one or more status output values, e.g. composite or probably prime for primality test. However, such status output might not be observed in the black box testing. The tester should perform source code inspection to verify that each distinct status output in the specification is mapped to the different value in the IUT.

NOTE 1 ADV_FSP.4-8 in ISO/IEC 18045.

NOTE 2 ADV_TDS.3-10 in ISO/IEC 18045.

NOTE 3 ADV_IMP.1-3 in ISO/IEC 18045.

9.1.3.1.5 Source code rarely executed

Currently, there are no dedicated guidelines for source code that is rarely executed. However, guidelines in 9.1.3.1.4 should be followed if a status output is included in the source code that is rarely executed.

© ISO/IEC 2016 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: ISO IEC 18367-2016

9.1.3.2 Guidelines for binary analysis

Under the condition that the source code of the IUT is inspected to be conformant, binary analysis should be performed to confirm that the final implementation representation corresponds to the original source code.

9.2 Guidelines specific to encryption algorithms

9.2.1 Identification of encryption algorithms

9.2.1.1 Identification of block ciphers

In addition to the identification information in 9.1.1, the vendor shall identify the following information on the IUT:

a) bit length of key supported;

b) bit length of input block supported.

In many cases, implementations of symmetric key algorithms support a block cipher mode of operation in order to process multiple blocks.

If the IUT supports a block cipher mode of operation, the vendor shall identify the following information on the IUT:

a) block cipher mode of operation;

b) bit length of initialization vector;

c) maximum number of blocks supported.

If a counter mode is supported by the IUT, the vendor shall identify the following information on the IUT:

d) how the counter is constructed, including its format

9.2.1.2 Identification of stream ciphers

The vendor shall identify the following information on the implementation of stream cipher:

a) reference to the specification of stream cipher;

b) bit length of key supported;

c) bit length of initialization vector, if any.

9.2.1.3 Identification of asymmetric ciphers

In addition to the identification information in 9.1.1, the vendor shall identify the following information on the IUT:

a) consistency between security function, supporting functions and supported parameters;

b) bit length of each public key component;

c) bit length of each private key component.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO IEC 18367-2016

9.2.2 Selecting a set of conformance test items

9.2.2.1 Block ciphers

A block cipher sometimes becomes the underlying algorithm of higher-level algorithms, e.g. MAC and RBG. In this sense, implementations of symmetric key algorithms should be extensively tested.

Taking into account the above situation, the following five basic black box test items can be applicable to block ciphers:

a) KAT-Text, which supplies various plaintexts (or ciphertexts) to the IUT and tests if the resultant ciphertexts (or plaintexts) match the expected values, while using a fixed key;

b) KAT-Key, which uses various keys in the IUT and tests if the resultant ciphertexts (or plaintexts) match the expected values, while using fixed plaintexts (or ciphertexts);

c) KAT-Sbox, which utilizes all of the entries in the Sbox and tests if the resultant ciphertexts (or plaintexts) match the expected values;

d) MMT;

e) MCT.

In addition to the above test items, the following optional white box tests can be applicable to symmetric key algorithms:

f) source code inspection;

g) binary analysis.

9.2.2.2 Stream ciphers

The construction of stream cipher algorithms will be different from that of block ciphers; therefore, the same test items as block ciphers cannot always be applicable to stream ciphers.

Taking into account the above situation, the following two basic black box test items can be applicable to stream ciphers:

a) KAT-Key, which uses various keys in the IUT and tests if the resultant ciphertexts (or plaintexts) match the expected values, while using fixed plaintexts (or ciphertexts);

b) MCT.

In addition to the above test items, the following optional white box tests can be applicable to symmetric key algorithms:

c) source code inspection;

d) binary analysis.

9.2.2.3 Asymmetric ciphers

Before selecting a set of conformance test items, the tester shall verify that the conformance to the underlying algorithms in asymmetric ciphers is already met. Therefore, it is assumed that the underlying algorithms are extensively tested. The following KAT or independent verification test should be selected:

a) Random test, which uses various information needed by the IUT and tests:

   1) if the resultant ciphertext matches the expected value for the encryption function, or that the resultant ciphertext can be decrypted by a reference implementation,

Documents
Document Sharing Hub

© ISO/IEC 2016 – All rights reserved

Full document details: ISO IEC 18367-2016