INTERNATIONAL STANDARD

ISO/IEC 29192-5

First edition
2016-08-01

Information technology — Security techniques — Lightweight cryptography —

Part 5: Hash-functions

Technologies de l'information — Techniques de sécurité — Cryptographie pour environnements contraints — Partie 5: Fonctions de hachage

Reference number
ISO/IEC 29192-5:2016(E)

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

Contents

Page

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 29192 consists of the following parts, under the general title *Information technology — Security techniques — Lightweight cryptography*:

— *Part 1: General*

— *Part 2: Block ciphers*

— *Part 3: Stream ciphers*

— *Part 4: Mechanisms using asymmetric techniques*

— *Part 5: Hash-functions*

Further parts may follow.

Introduction

This part of ISO/IEC 29192 specifies lightweight hash-functions, which are tailored for implementation in constrained environments.

ISO/IEC 29192-1 specifies the requirements for lightweight cryptography.

A hash-function maps an arbitrary string of bits to a fixed-length string of bits.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this part of ISO/IEC 29192 may involve the use of patents. The ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world.

In this respect, the statements of the holders of these patent rights are registered with the ISO and IEC. Information may be obtained from the following:

Nanyang Technological University - NTUitive Pte Ltd

16 Nanyang Drive, #01-109, Innovation Centre, Singapore 637722

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

Information technology — Security techniques — Lightweight cryptography —

Part 5: Hash-functions

1 Scope

This part of ISO/IEC 29192 specifies three hash-functions suitable for applications requiring lightweight cryptographic implementations.

— PHOTON: a lightweight hash-function with permutation sizes of 100, 144, 196, 256 and 288 bits computing hash-codes of length 80, 128, 160, 224, and 256 bits, respectively.

— SPONGENT: a lightweight hash-function with permutation sizes of 88, 136, 176, 240 and 272 bits computing hash-codes of length 88, 128, 160, 224, and 256 bits, respectively.

— Lesamnta-LW: a lightweight hash-function with permutation size 384 bits computing a hash-code of length 256 bits.

The requirements for lightweight cryptography are given in ISO/IEC 29192-1.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 29192-1, *Information technology — Security techniques — Lightweight cryptography — Part 1: General*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
absorbing phase
input phase of a sponge function

[SOURCE: [4]]

3.2
bitrate
part of the internal state of a sponge function of length r bits

[SOURCE: [4]]

3.3
capacity
part of the internal state of a sponge function of length c bits

[SOURCE: [4]]

3.4
collision resistance
computationally infeasible to find any two distinct inputs which map to the same output of a hash-function

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

3.5
hash-code
string of bits which is the output of a hash-function

Note 1 to entry: The literature on this subject contains a variety of terms that have the same or similar meaning as hash-code. Modification Detection Code, Manipulation Detection Code, digest, hash-result, hash-value and imprint are some examples.

[SOURCE: ISO/IEC 10118-1:—[1)], 2.3]

3.6
hash-function
function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

— it is computationally infeasible to find for a given output, an input which maps to this output;

— it is computationally infeasible to find for a given input, a second input which maps to the same output

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

[SOURCE: ISO/IEC 10118-1:—[1)], 2.4]

3.7
initializing value
value used in defining the starting point of a hash-function

Note 1 to entry: The literature on this subject contains a variety of terms that have the same or similar meaning as initializing value. Initialization vector and starting value are examples.

[SOURCE: ISO/IEC 10118-1:—[1)], 2.5]

3.8
preimage resistance
computationally infeasible to find for a given output of a hash-function, an input which maps to this output

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

3.9
second preimage resistance
computationally infeasible to find for a given input of a hash-function, a second input which maps to the same output

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

3.10
sponge function
mode of operation, based on a fixed-length permutation (or transformation) and a padding rule, which builds a function mapping variable-length input to variable-length output

[SOURCE: [4]]

1) To be published. (Revision of ISO/IEC 10118-1:2000)

3.11
squeezing phase
output phase of a sponge function

[SOURCE: [4]]

4 Symbols

$\{0\}^c$	bit-string containing exactly c zeros
0x	prefix indicating a binary string in hexadecimal notation
||	concatenation of bit strings
$a \leftarrow b$	set variable a to the value of b
$\oplus$	bitwise exclusive-OR operation
c	length of the capacity in bits
hash	n-bit hash-code
IV	t-bit initialization value
m_i	message block i of r bits
n	length of the hash code in bits
r	length of the bitrate in bits
S_i	t-bit internal state at iteration i
t	length of the internal state in bits
$\lceil x \rceil$	the smallest integer greater than or equal to the real number x

5 Lightweight hash-functions optimized for hardware implementations

5.1 General

Clause 5 specifies PHOTON and SPONGENT hash-functions which are optimized for hardware implementations. ISO/IEC 29192-1 shall be referred to for the requirements for lightweight cryptography.

5.2 PHOTON

5.2.1 General

In order to cover a wide spectrum of applications, five different variants of PHOTON[5] are specified. Each variant is defined by its internal permutation size $t = c + r$, where c and r denote the *capacity* and the *bitrate*, respectively. For a fixed permutation size t, the choice of c and r provides a security-efficiency trade-off. PHOTON-t denotes the variant using a t-bit internal permutation.

The five variants are the following:

a) **PHOTON-100** computes an 80-bit hash-code and offers 64-bit preimage resistance, 40-bit second preimage resistance, and 40-bit collision resistance.

b) **PHOTON-144** computes a 128-bit hash-code and offers 112-bit preimage resistance, 64-bit second preimage resistance, and 64-bit collision resistance.

c) **PHOTON-196** computes a 160-bit hash-code and offers 124-bit preimage resistance, 80-bit second preimage resistance, and 80-bit collision resistance.

d) **PHOTON-256** computes a 224-bit hash-code and offers 192-bit preimage, 112-bit second preimage resistance, and 112-bit collision resistance.

e) **PHOTON-288** computes a 256-bit hash-code and offers 224-bit preimage, 128-bit second preimage resistance, and 128-bit collision resistance.

PHOTON-100 does not provide the minimum security strength as required in ISO/IEC 29192-1. It shall not be used as a general purpose hash function. PHOTON-144 does not provide the minimum security strength for collision resistance and second preimage resistance as required in ISO/IEC 29192-1. It shall only be used in applications where collision resistance and second preimage resistance are not required.

5.2.2 PHOTON specific notation

P_t internal permutation, where $t \in \{100,144,196,256,288\}$

z_i the r' leftmost bits of the internal state S

c' length of the capacity in bits during the squeezing phase of PHOTON

d number of rows and columns of the internal state matrix

r' length of the bitrate in bits during the squeezing phase of PHOTON

$S[i,j]$ the s-bit internal state cell located at row i and column j, with $0 \leq i, j < d$

$RC(v)$ round constant of round v

$IC_d(i)$ internal constants of row i

X_r 3-bit or 4-bit internal state of a shift register to generate the round constants $RC(v)$ or the internal constants $IC_d(i)$

$FB()$ feedback function to update the internal state of a shift register

SBOX_{PRE} the 4-bit substitution table (S-box) also used in the block cipher PRESENT[1]

SBOX_{AES} the 8-bit substitution table (S-box) also used in the Advanced Encryption Algorithm[2]

5.2.3 Domain extension algorithm

The message M to hash is first padded by appending a "1" bit and as many zeros (possibly none), such that the total length is a multiple of the bitrate, r, and finally l message blocks $m_0,\ldots, m_{l-1}$ of r bits each can be obtained. The t-bit internal state, S, is initialized by setting it to the value $S_0=IV=\{0\}^{t-24}||n/4||r||r'$, where each value is coded on 8 bits.

NOTE For implementation purposes, each byte is interpreted in big-endian form, that is, the leftmost bit is the most significant bit.

Then, as for the classical sponge strategy, at iteration i the message block m_i is absorbed on the leftmost part of the internal state S_i and then the permutation P_t is applied, i.e.

$$S_{i+1} \leftarrow P_t(S_i \oplus (m_i \| \{0\}^c)).$$

Once all l message blocks have been absorbed, the hash value is built by concatenating the successive r'-bit output blocks z_i until the appropriate output size n is reached:

$$\text{hash} = z_0 \| \ldots \| z_{l'-1}$$

with the rightmost bits truncated if necessary to produce an n-bit hash. More precisely, z_i is the r' leftmost bits of the internal state S_{l+i} and $S_{l+i+1} \leftarrow P_t(S_{l+i})$ for $0 \le i < l'$, where l' denotes the number of squeezing iterations, that is $l' = \lceil n / r' \rceil - 1$. If the hash output size is not a multiple of r', one just truncates $z_{l'-1}$ to n mod r' bits.

5.2.4 Internal permutation

5.2.4.1 General

The internal permutations P_t, where $t \in \{100,144,196,256,288\}$, are applied to an internal state of d^2 elements of s bits each, which can be represented as a $(d \times d)$ matrix. P_t is composed of N_r rounds, each containing four layers as depicted in Figure 1:

a) AddConstants (AC),

b) SubCells (SC),

c) ShiftRows (ShR), and

d) MixColumnsSerial (MCS).

Table 1 shows an overview of the parameters of the different variants of PHOTON.

Table 1 — Overview of parameters of PHOTON

Variant	t	c	r	r'	d	s	N_r	$IC_d(.)$	Irr. polynomial	Z_i coefficients
PHOTON-100	100	80	20	16	5	4	12	[0, 1, 3, 6, 4]	$x^4 + x + 1$	(1, 2, 9, 9, 2)
PHOTON-144	144	128	16	16	6	4	12	[0,1, 3, 7, 6, 4]	$x^4 + x + 1$	(1, 2, 8, 5, 8, 2)
PHOTON-196	196	160	36	36	7	4	12	[0,1, 2, 5, 3, 6, 4]	$x^4 + x + 1$	(1, 4, 6, 1, 1, 6, 4)
PHOTON-256	256	224	32	32	8	4	12	[0,1, 3, 7, 15, 14, 12, 8]	$x^4 + x + 1$	(2, 4, 2, 11, 2, 8, 5, 6)
PHOTON-288	288	256	32	32	6	8	12	[0, 1, 3, 7, 6, 4]	$x^8 + x^4 + x^3 + x + 1$	(2, 3, 1, 2, 1, 4)

NOTE Always a cell size of 4 bits is used, except for the largest version for which 8-bit cells are used, and that the number of rounds is always $N_r = 12$ for all values of t. The output rate r' is always the same as the input rate r, except for PHOTON-100. The internal state cell located at row i and column j is denoted $S[i,j]$ with $0 \le i, j < d$.

Informally, AddConstants simply consists in adding fixed values to the cells of the internal state, while SubCells applies an s-bit S-box to each of them. ShiftRows rotates the position of the cells in each of the rows and MixColumnsSerial linearly mixes all the columns independently.

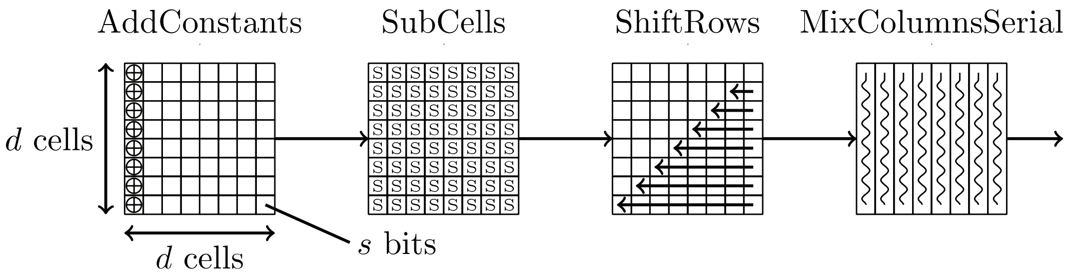

Figure 1 — One round of a PHOTON permutation

5.2.4.2 AddConstants

At round number v (start the counting from 1), first a round constant $RC(v)$ is XORed to each cell $S[i,0]$ of the first column of the internal state. Then, distinct internal constants $IC_d(i)$ are XORed to each cell $S[i,0]$ of the same first column. Overall, for round v it holds that

$$S'\left[i,0\right] \leftarrow S\left[i,0\right] \oplus RC\left(v\right) \oplus IC_d\left(i\right) \textit{ for all } 0 \le i < d.$$

The round constants $RC(v)$ have been generated by a 4-bit linear feedback shift register with maximum cycle length; they are

$$RC\left(v\right) \;=\; \left[1,\ 3,\ 7,\ 14,\ 13,\ 11,\ 6,\ 12,\ 9,\ 2,\ 5,\ 10\right].$$

The internal constants, $IC_d(i)$, depend on the square size d and on the row position i and they have been generated by shift registers with a cycle length of d. For all variants shift registers with $l = 3$ bits are used, except for $d = 8$, where $l = 4$ is used. The internal state of the shift register is denoted with $X_r = (x_{l-1}, ..., x_1, x_0)$, where each $x_i = \{0,1\}$, and the state is initialized with all 0's, that is $X_0 = (0, \ldots, 0, 0)$. Then in each update iteration the new content of the shift register is given by $X_{r+1} \leftarrow (x_{l-2}, ..., x_0, FB(X_r))$, where $FB(X_r)$ is the feedback function. The round constants are computed by $FB(X_r) = x_3$ XNOR x_2, while the feedback functions for the internal constants are shown in Table 2. Constants for all square sizes, round numbers, and row positions are displayed in Table 3 through Table 6.

Table 2 — Feedback functions for internal constants generation

d	**5**	**6**	**7**	**8**
$FB(X_r)$	x_2 NOR x_1	NOT x_2	x_2 XNOR x_0	NOT x_3
$IC_d(.)$	[0, 1, 3, 6, 4]	[0, 1, 3, 7, 6, 4]	[0, 1, 2, 5, 3, 6, 4]	[0, 1, 3, 7, 15, 14, 12, 8]

Table 3 — $RC(v) \oplus IC_d(i)$ for $d = 5$

Round *v* / Row *i*	1	2	3	4	5	6	7	8	9	10	11	12
0	1	3	7	14	13	11	6	12	9	2	5	10
1	0	2	6	15	12	10	7	13	8	3	4	11
2	2	0	4	13	14	8	5	15	10	1	6	9
3	7	5	1	8	11	13	0	10	15	4	3	12
4	5	7	3	10	9	15	2	8	13	6	1	14

