DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 29192-5

ISO/IEC JTC **1**/SC **27** Secretariat: **DIN**

Voting begins on: **2015-07-27**

Voting terminates on: **2015-10-27**

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

ICS: 35.040

Reference number
ISO/IEC DIS 29192-5:2015(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

ISO/IEC 29192-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *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 18033 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:

Patent holder name:

Postal address:

Patent holder name:

Postal address:

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.

ISO/IEC 29192-1 shall be referred for the requirements for lightweight cryptography.

2 Normative references

The following referenced document is indispensable for the application 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 29192-1, *Information technology — Security techniques — Lightweight Cryptography — Part 1: General*

3 Terms and definitions

For the purpose 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
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 10118-1:2000, definition 3.4]

3.5
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 10118-1:2000, definition 3.5]

3.6
initializing value
value used in defining the starting point of a hash-function

[SOURCE: ISO/IEC 10118-1:2000, definition 3.7]

3.7
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]]

3.8
squeezing phase
output phase of a sponge function

[SOURCE: [4]]

4 Symbols

0x	A 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

5 Lightweight hash-functions optimized for hardware implementations

5.1 PHOTON

5.1.1 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. We denote PHOTON-t the variant using a t-bit internal permutation. <- Tatsuta: "we" should not be used.

The five variants are:

a) **PHOTON-100** computes an 80-bit hash-code and offers 64-bit preimage, 40-bit 2nd-preimage, and 40-bit collision security

b) **PHOTON-144** computes a 128-bit hash-code and offers 112-bit preimage, 64-bit 2nd-preimage, and 64-bit collision security

c) **PHOTON-196** computes a 160-bit hash-code and offers 124-bit preimage, 80-bit 2nd-preimage, and 80-bit collision security

d) **PHOTON-256** computes a 224-bit hash-code and offers 192-bit preimage, 112-bit 2nd-preimage, and 112-bit collision security

e) **PHOTON-288** computes a 256-bit hash-code and offers 224-bit preimage, 128-bit 2nd-preimage, and 128-bit collision security

NOTE The first proposal is special in the sense that it is designed for the specific cases where 64-bit preimage security is considered to be sufficient. In contrary, the last proposal provides a high security level of 128-bit collision resistance, thus making it suitable for generic applications.

5.1.2 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 oft he 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]

5.1.3 The 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 we can finally obtain l message blocks $m_0, \ldots, m_{l-1}$ of r bits each. 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. <- Tatsuta: "we" should not be used.

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 we absorb the message block m_i on the leftmost part of the internal state S_i and then apply the permutation P_t, i.e. <- Tatsuta: "we" should not be used.

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

Once all l message blocks have been absorbed, we build the hash value by concatenating the successive r'-bit output blocks z_i until we reach the appropriate output size n: <- Tatsuta: "we" should not be used.

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

with the rightmost bits truncated if necessary to produce an n-bit hash. More precisely, z_i are the r' leftmost bits of the internal state S_{l+i} and we have $S_{l+i+1} \leftarrow P_t(S_{l+i})$ for $0 \leq i < l'$, 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. <- Tatsuta: "we" should not be used.

5.1.4 The internal permutation

5.1.4.1 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 x 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),

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	d	s	N_r	$IC_d(.)$	Irr. polynomial	Z_i coefficients
PHOTON-100	100	5	4	12	[0, 1, 3, 6, 4]	x^4+x+1	(1, 2, 9, 9, 2)
PHOTON-144	144	6	4	12	[0,1, 3, 7, 6, 4]	x^4+x+1	(1, 2, 8, 5, 8, 2)
PHOTON-196	196	7	4	12	[0,1, 2, 5, 3, 6, 4]	x^4+x+1	(1, 4, 6, 1, 1, 6, 4)
PHOTON-256	256	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	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 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.

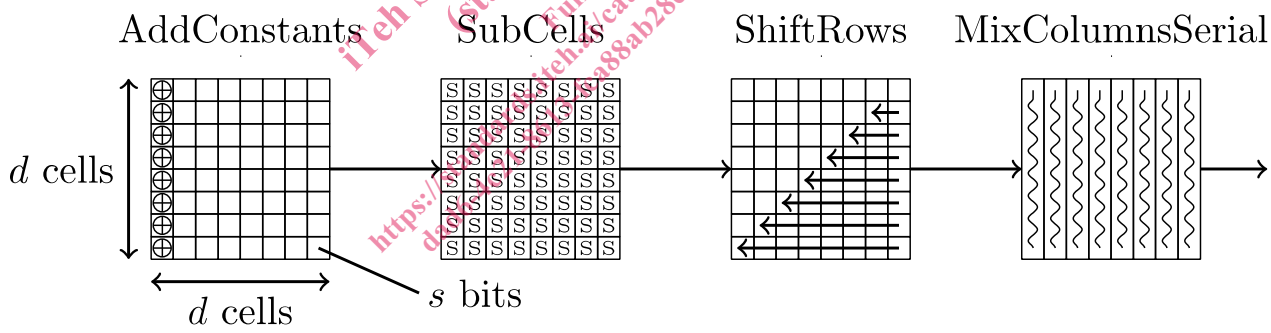

Figure 1 — One round of a PHOTON permutation.

5.1.4.2 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 we have <- Tatsuta: "we" should not be used.

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

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

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