INTERNATIONAL STANDARD

ISO/IEC 18033-4

First edition
2005-07-15

AMENDMENT 1
2009-12-15

Information technology — Security techniques — Encryption algorithms —

Part 4: Stream ciphers

AMENDMENT 1: Rabbit and Decim

iTeh STANDARD PREVIEW
(standards.iteh.ai)

Technologies de l'information — Techniques de sécurité — Algorithmes de chiffrement —

Partie 4: Chiffrements en flot

AMENDEMENT 1: Rabbit et Decim

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/ba968884-4bca-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

Reference number
ISO/IEC 18033-4:2005/Amd.1:2009(E)

© ISO/IEC 2009

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2009

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

 © ISO/IEC 2009 – All rights reserved

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.

Amendment 1 to ISO/IEC 18033-4:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

This Amendment introduces two additional keystream generators for use as stream ciphers: Rabbit and Decim^{v2}.

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-650664f04339/iso-iec-18033-4-2005-amd-1-2009

Rabbit is specified in 7.3, and test vectors are given in A.4.

Decim^{v2} is specified in 7.4, and test vectors are given in A.5.

For all keystream generators, security statements are given in Annex B, and object identifiers are given in Annex C.

© ISO/IEC 2009 – All rights reserved

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

Information technology — Security techniques — Encryption algorithms —

Part 4: Stream ciphers

AMENDMENT 1: Rabbit and Decim

Page 4, Clause 4, immediately before b_i

Add the following:

AND Bitwise logical AND operation.

Page 4, Clause 4, line 21

Replace with the following:

OR Bitwise logical OR operation.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

Page 5, immediately before 4.1

Add the following note:

NOTE Additional variables and notation specific to a given keystream generator are introduced with the algorithm.

Page 23, after 7.2.7

Add the following new subclauses:

7.3 Rabbit keystream generator

Rabbit is a keystream generator which uses a 128-bit secret key K, a 64-bit initialization vector IV, and a 513-bit internal state variable S_i ($i \geq 0$). It outputs a 128-bit keystream block Z_i at every iteration of the function *Strm*.

The 513 bits of the internal state S_i are divided between eight 32-bit state variables $X_0^{(i)}$, ...,$X_7^{(i)}$, eight 32-bit counter variables $C_0^{(i)}$, ...,$C_7^{(i)}$, and one counter carry bit $b^{(i)}$.

The description uses the notation defined in Section 4 of the standard. In addition, a special notation for bit arrays is used to enhance readability: when labeling the bits of a variable A, the least significant bit is denoted by $A^{[0]}$. The notation $A^{[h..g]}$ represents bits h through g of variable A, where bit position h is more significant than bit position g.

NOTE 1 For Rabbit, the maximum recommended amount of keystream produced from a given key K is 2^{64} keystream blocks. This provides a large security margin against cryptanalysis, while at the same time implying no practical limitations on the applicability of the algorithm.

NOTE 2 We refer to [1] for the original proposal of the cipher and to [2] for an overview of its cryptographic security.

© ISO/IEC 2009 – All rights reserved

7.3.1 Additional variables and notation

In the specification of the Rabbit keystream generator, the following specific notation is used:

A	Constant for Rabbit
b	Carry bit for Rabbit
C	Counter variable for Rabbit
g	Subfunction used for Rabbit
X	Inner state variable for Rabbit

In addition, a number of other symbols are used for auxiliary local variables in algorithm descriptions. These symbols occur only within a given function specification and do not have a global meaning. They are thus described in the function declaration.

7.3.2 Initialization function *Init*

In the following, the initialization function *Init* of Rabbit is specified.

INPUT: 128-bit key K, 64-bit initialization vector IV.

OUTPUT: Initial value of the state variable $S_0 = (b^{(0)}, X_0^{(0)}, ..., X_7^{(0)}, C_0^{(0)}, ..., C_7^{(0)})$.

Local variables: counters i, j

iTeh STANDARD PREVIEW
(standards.iteh.ai)

1. Let $K_0 = K^{[15..0]}$, $K_1 = K^{[31..16]}$, ..., and $K_7 = K^{[127..112]}$.

2. Set S_{-9} as follows:

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

   2.1. Set $b^{(-9)} = 0$.

   2.2. For $j = 0, 1, ..., 7$:

      2.2.1. If j is even:

         2.2.1.1. Set $X_j^{(-9)} = K_{(j+1 \bmod 8)} \,||\, K_j$.

         2.2.1.2. Set $C_j^{(-9)} = K_{(j+4 \bmod 8)} \,||\, K_{(j+5 \bmod 8)}$.

      2.2.2. Else:

         2.2.2.1. Set $X_j^{(-9)} = K_{(j+5 \bmod 8)} \,||\, K_{(j+4 \bmod 8)}$.

         2.2.2.2. Set $C_j^{(-9)} = K_j \,||\, K_{(j+1 \bmod 8)}$.

3. Iterate the next-state function *Next* four times:

   3.1. For $i = -8, -7, -6, -5$:

      3.1.1. $S_i = Next\,(S_{i-1})$

4. Set S_{-4} as follows:

   4.1. Modify the counters as follows:

$$C_0^{(-4)} = C_0^{(-5)} \oplus X_4^{(-5)} \oplus IV^{[31..0]} \qquad C_1^{(-4)} = C_1^{(-5)} \oplus X_5^{(-5)} \oplus (IV^{[63..48]} \,||\, IV^{[31..16]})$$
$$C_2^{(-4)} = C_2^{(-5)} \oplus X_6^{(-5)} \oplus IV^{[63..32]} \qquad C_3^{(-4)} = C_3^{(-5)} \oplus X_7^{(-5)} \oplus (IV^{[47..32]} \,||\, IV^{[15..0]})$$
$$C_4^{(-4)} = C_4^{(-5)} \oplus X_0^{(-5)} \oplus IV^{[31..0]} \qquad C_5^{(-4)} = C_5^{(-5)} \oplus X_1^{(-5)} \oplus (IV^{[63..48]} \,||\, IV^{[31..16]})$$
$$C_6^{(-4)} = C_6^{(-5)} \oplus X_2^{(-5)} \oplus IV^{[63..32]} \qquad C_7^{(-4)} = C_7^{(-5)} \oplus X_3^{(-5)} \oplus (IV^{[47..32]} \,||\, IV^{[15..0]})$$

© ISO/IEC 2009 – All rights reserved

4.2. Set $X_0^{(-4)} = X_0^{(-5)}$, ..., $X_7^{(-4)} = X_7^{(-5)}$, $b^{(-4)} = b^{(-5)}$.

5. Iterate the next-state function *Next* four times:

   5.1. For i = -3, -2, -1, 0:

      5.1.1. $S_i = Next\ (S_{i-1})$

6. Output $S_0 = (b^{(0)}, X_0^{(0)}, ..., X_7^{(0)}, C_0^{(0)}, ..., C_7^{(0)})$.

NOTE The IV is mixed into the internal state in steps 4 and 5 of the algorithm. If the application requires frequent re-initialization under the same key, it makes sense to store the internal state after step 3 as master state and to perform only steps 4 through 6 for re-initialization.

7.3.3 Next-state function *Next*

The next-state function *Next* of Rabbit is specified as follows:

INPUT: State variable $S_i = (b^{(i)}, X_0^{(i)}, ..., X_7^{(i)}, C_0^{(i)}, ..., C_7^{(i)})$.

OUTPUT: State variable $S_{i+1} = (b^{(i+1)}, X_0^{(i+1)}, ..., X_7^{(i+1)}, C_0^{(i+1)}, ..., C_7^{(i+1)})$.

Local variables: counter j, 33-bit positive integer *temp*

1. Set constants A_0, ..., A_7 as follows:

   A_0 = 0x4D34D34D A_1 = 0xD34D34D3
   A_2 = 0x34D34D34 A_3 = 0x4D34D34D
   A_4 = 0xD34D34D3 A_5 = 0x34D34D34
   A_6 = 0x4D34D34D A_7 = 0xD34D34D3

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

2. Let $b_0^{(i+1)} = b^{(i)}$

3. For j = 0, 1, ..., 7:

   3.1. Let $temp = C_j^{(i)} + A_j + b_j^{(i+1)}$; this results in a 33-bit value.

   3.2. Let $b_{j+1}^{(i+1)} = temp^{[32]}$.

   3.3. Let $C_j^{(i+1)} = temp^{[31..0]}$.

4. Let $b^{(i+1)} = b_8^{(i+1)}$

5. For j = 0, 1, ..., 7:

   5.1. Let $G_j = g(X_j^{(i)}, C_j^{(i+1)})$. The detailed description of the function g is given in 7.3.5.

6. Modify internal state as follows:

$$
\begin{aligned}
X_0^{(i+1)} &= G_0 +_{32} (G_7 <<<_{32} 16) +_{32} (G_6 <<<_{32} 16)\\
X_1^{(i+1)} &= G_1 +_{32} (G_0 <<<_{32} 8) +_{32} G_7\\
X_2^{(i+1)} &= G_2 +_{32} (G_1 <<<_{32} 16) +_{32} (G_0 <<<_{32} 16)\\
X_3^{(i+1)} &= G_3 +_{32} (G_2 <<<_{32} 8) +_{32} G_1\\
X_4^{(i+1)} &= G_4 +_{32} (G_3 <<<_{32} 16) +_{32} (G_2 <<<_{32} 16)\\
X_5^{(i+1)} &= G_5 +_{32} (G_4 <<<_{32} 8) +_{32} G_3\\
X_6^{(i+1)} &= G_6 +_{32} (G_5 <<<_{32} 16) +_{32} (G_4 <<<_{32} 16)\\
X_7^{(i+1)} &= G_7 +_{32} (G_6 <<<_{32} 8) +_{32} G_5
\end{aligned}
$$

7. Output $S_{i+1} = (b^{(i+1)}, X_0^{(i+1)}, ..., X_7^{(i+1)}, C_0^{(i+1)}, ..., C_7^{(i+1)})$.

© ISO/IEC 2009 – All rights reserved

7.3.4 Keystream function *Strm*

The keystream function *Strm* of Rabbit is specified as follows:

INPUT: State variable $S_i = (b^{(i)}, X_0^{(i)}, ..., X_7^{(i)}, C_0^{(i)}, ..., C_7^{(i)})$.

OUTPUT: Keystream block Z_i.

1. Set Z_i as follows:

$$
\begin{aligned}
Z_i^{[15..0]} &= X_0^{(i)}[15..0] \oplus X_5^{(i)}[31..16] \\
Z_i^{[31..16]} &= X_0^{(i)}[31..16] \oplus X_3^{(i)}[15..0] \\
Z_i^{[47..32]} &= X_2^{(i)}[15..0] \oplus X_7^{(i)}[31..16] \\
Z_i^{[63..48]} &= X_2^{(i)}[31..16] \oplus X_5^{(i)}[15..0] \\
Z_i^{[79..64]} &= X_4^{(i)}[15..0] \oplus X_1^{(i)}[31..16] \\
Z_i^{[95..80]} &= X_4^{(i)}[31..16] \oplus X_7^{(i)}[15..0] \\
Z_i^{[111..96]} &= X_6^{(i)}[15..0] \oplus X_3^{(i)}[31..16] \\
Z_i^{[127..112]} &= X_6^{(i)}[31..16] \oplus X_1^{(i)}[15..0]
\end{aligned}
$$

2. Output Z_i.

7.3.5 Function *g*

The function *g* is specified as follows:

iTeh STANDARD PREVIEW
(standards.iteh.ai)

INPUT: Two 32-bit parameters *u* and *v*.

OUTPUT: 32-bit result $g(u,v)$.

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

Local variables: 64-bit positive integer *temp*

1. Let $temp = (u +_{32} v)^2$; this results in a 64-bit value.

2. Let $g(u,v) = temp^{[31..0]} \oplus temp^{[63..32]}$.

3. Output $g(u,v)$.

7.4 Decim^{v2} keystream generator

Decim^{v2} is a keystream generator which uses an 80-bit secret key *K* and a 64-bit initialization vector *IV*. Decim^{v2} is composed of a 192-bit maximum length linear feedback shift register *A*, filtered by a 14-variable Boolean function *F*. In keystream generation mode, the output of *F* is used to feed a compression block which is a function called *ABSG*, whose output finally passes through a 32-bit long buffer *B* to regulate the keystream output rate.

NOTE 1 See Reference [3] for the theoretical background on the design rationale of Decim^{v2}.

The state variable S_i of Decim^{v2} consists of the 192-bit value $a^{(i)} = (a_0^{(i)}, a_1^{(i)}, ..., a_{191}^{(i)})$ of register *A*, a 3-bit variable $T^{(i)}$ which corresponds to the state of the compression function *ABSG*, the 32 bits $b^{(i)} = (b_0^{(i)}, b_1^{(i)}, ..., b_{31}^{(i)})$ in buffer *B*, and the number $l^{(i)}$ of bits in buffer *B* that are ready to be output.

 © ISO/IEC 2009 – All rights reserved

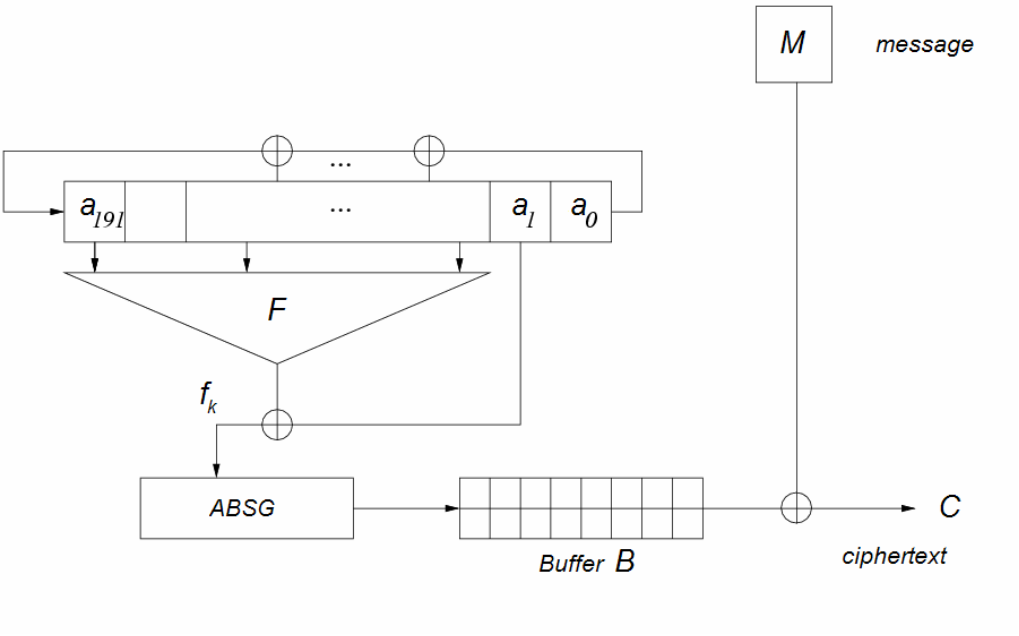

Figure 10 — Schematic drawing of Decim^{v2}.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

The *Init* function, defined in detail in 7.4.2, takes as input the 80-bit key K and the 64-bit initialization vector IV, and produces the initial value of the state variable $S_0 = (a^{(0)}, T^{(0)}, b^{(0)}, I^{(0)})$.

The *Next* function, defined in detail in 7.4.4, takes as an input the value of the state variable $S_i = (a^{(i)}, T^{(i)}, b^{(i)}, I^{(i)})$ and produces as output the next value of the state variable $S_{i+1} = (a^{(i+1)}, T^{(i+1)}, b^{(i+1)}, I^{(i+1)})$. The *Next* function can operate in any of three different modes, depending on whether the iteration performed is part of the initialization of the register, the initialization of the buffer, or the subsequent keystream generation.

The *Strm* function, defined in detail in 7.4.5, takes as an input the value of the state variable $S_i = (a^{(i)}, T^{(i)}, b^{(i)}, I^{(i)})$, and produces as output a keystream bit Z_i.

NOTE 2 The standard output rate of Decim^{v2} is 1/4. Therefore, in order to synchronize the state variable and the keystream output, the *Next* function performs four standard iterations of Decim^{v2} as specified in [3].

NOTE 3 The compression function of Decim^{v2} has a variable output rate, equal to 1/3 on average. Therefore, a buffer mechanism is used to ensure a constant output rate. The differences between the buffer output rate and the compression function output rate, as well as the buffer length, have been chosen to ensure that the buffer always functions as expected with overwhelming probability, as described in Section 7.4.2.

7.4.1 Additional variables and notation

In the specification of the Decim^{V2} keystream generator, the following specific notation is used:

a	Inner state variable for Decim^{v2}
ABSG	Compression function used for Decim^{v2}
b, b'	Inner state variables for Decim^{v2}
B	Buffering function used for Decim^{v2}
F	Linear feedback function used for Decim^{v2}

© ISO/IEC 2009 – All rights reserved

I, I'	Inner state variables for Decim^{v2}
L	Filtering function used for Decim^{v2}
T, T'	Inner state variables for Decim^{v2}
Y	Boolean function used for Decim^{v2}

In addition, a number of other symbols are used for auxiliary local variables in algorithm descriptions. These symbols occur only within a given function specification and do not have a global meaning. They are thus described in the function declaration.

7.4.2 Initialization function *Init*

The Initialization function *Init* is defined as follows.

INPUT: 80-bit key K, 64-bit initialization vector IV.

OUTPUT: Initial value of the state variable $S_0 = (a^{(0)}, T^{(0)}, b^{(0)}, I^{(0)})$.

Local variables: counters i, j

a) Initialize the register with the key K and the initialization vector IV.

1) Set $a_j^{(-256)} = K_j$ for j =0,1,…,79.

2) Set $a_j^{(-256)} = K_{j-80} \oplus IV_{j-80}$ for j =80,81,…,143.

3) Set $a_j^{(-256)} = K_{j-80} \oplus IV_{j-144} \oplus IV_{j-128} \oplus IV_{j-112} \oplus IV_{j-96}$ for j =144,145,…,159.

4) Set $a_j^{(-256)} = IV_{j-160} \oplus IV_{j-128} \oplus 1$ for j =160,161,…,191.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

b) Initialize the buffer and the compression function:

1) Set $T^{(-256)} = 000$.

2) Set $b_j^{(-256)} = 0$ for j =0,1,…,31.

3) Set $I^{(-256)} = 0$.

c) Set $S_{-64} = InitNext^{192}(S_{-256}, \text{LFSR})$.

d) Set i = -64.

e) While $I^{(i)} < 32$ and $i<0$:

1) Set $S_{i+1} = InitNext(S_i, \text{BUFF})$.

2) Set $i=i+1$.

f) Set $S_0 = S_i$.

g) Output S_0.

NOTE Steps d), e) and f) of the Decim^{v2} initialization involve filling the buffer before starting the keystream output. As the output rate of the compression function varies, the number of steps required to fill the buffer may vary. In step e), the *InitNext*(BUFF) function is iterated 64 times at most, which guarantees that the buffer is full with probability more than $1-2^{-97}$. On average, the buffer is full after 24 iterations. If a fixed, constant number of steps in the Init function is needed for implementation, the test $I^{(i)} < 32$ in step e) can be removed.

© ISO/IEC 2009 – All rights reserved

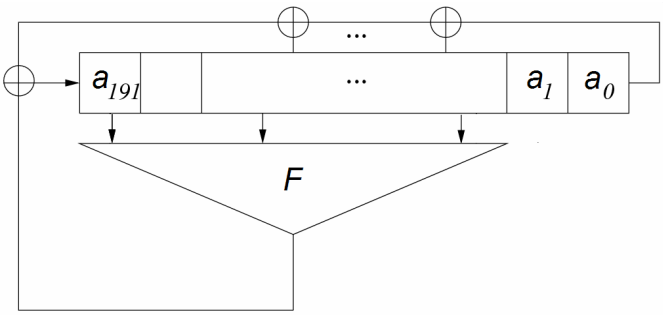

Figure 11 — Initialization mechanism.

7.4.3 Initialization Next-state function *InitNext*

Decim^{v2} has two modes for the *InitNext* function: one mode is used during the initialization of the register A and the second during the initial filling of the buffer.

INPUT: State variable $S_i = (a^{(i)}, T^{(i)}, b^{(i)}, l^{(i)})$, *mode* $\in$ {LFSR, BUFF}.

OUTPUT: Next value of the state variable $S_{i+1} = (a^{(i+1)}, T^{(i+1)}, b^{(i+1)}, l^{(i+1)})$.

Local variables: counters j, k, buffers f_k, r, c,
state buffers $\alpha^{(0)}, \ldots, \alpha^{(4)}, t^{(0)}, \ldots, t^{(4)}, \beta^{(0)}, \ldots, \beta^{(4)}, l^{(0)}, \ldots, l^{(4)}$.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

LFSR mode (execute if mode = LFSR)**:**

a) Update the state of the register A with the following steps:

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

1) Set $\alpha^{(0)} = a^{(i)}$.

2) For k = 0, 1, 2, 3:

   i) Set $f_k = F(\alpha^{(k)})$ and $r = L(\alpha^{(k)}) \oplus f_k$.

   ii) For j =0,1,…,190 set $\alpha_j^{(k+1)} = \alpha_{j+1}^{(k)}$.

   iii) Set $\alpha_{191}^{(k+1)} = r$.

3) Set $a^{(i+1)} = \alpha^{(4)}$.

BUFF mode (execute if mode = BUFF)**:**

a) Update the state of the register A with the following steps:

1) Set $\alpha^{(0)} = a^{(i)}$.

2) For k = 0, 1, 2, 3:

   i) Set $f_k = \alpha_1^{(k)} \oplus F(\alpha^{(k)})$ and $r = L(\alpha^{(k)})$.

   ii) For j =0,1,…,190 set $\alpha_j^{(k+1)} = \alpha_{j+1}^{(k)}$.

   iii) Set $\alpha_{191}^{(k+1)} = r$.

3) Set $a^{(i+1)} = \alpha^{(4)}$.

© ISO/IEC 2009 – All rights reserved

b) Set $\tau^{(0)} = T^{(i)}$, $\beta^{(0)} = b^{(i)}$, $\iota^{(0)} = l^{(i)}$.

c) For k = 0, 1, 2, 3:

1) Update the state of the compression block with the following steps:

i) Set $c = f_k \oplus \tau_0^{(k)}$.

ii) Set $\tau^{(k+1)} = ABSG(\tau^{(k)}, f_k)$.

iii) If $\tau_0^{(k+1)} = 0$, set *output* = TRUE , otherwise set *output* = FALSE.

2) Update the state of the buffer by $(\beta^{(k+1)}, \iota^{(k+1)}) = B(\beta^{(k)}, \iota^{(k)}, output, c)$.

d) Set $T^{(i+1)} = \tau^{(4)}$.

e) Set $b^{(i+1)} = \beta^{(4)}$ and $l^{(i+1)} = \iota^{(4)}$.

7.4.4 Next-state function *Next*

INPUT: State variable $S_i = (a^{(i)}, T^{(i)}, b^{(i)}, l^{(i)})$.

OUTPUT: Next value of the state variable $S_{i+1} = (a^{(i+1)}, T^{(i+1)}, b^{(i+1)}, l^{(i+1)})$.

Local variables: counters j, k, buffers f_k, r, c,
state buffers $\alpha^{(0)}, \ldots, \alpha^{(4)}$, $\tau^{(0)}, \ldots, \tau^{(4)}$, $\beta^{(0)}, \ldots, \beta^{(4)}$, $\iota^{(0)}, \ldots, \iota^{(4)}$.

iTeh STANDARD PREVIEW
(standards.iteh.ai)

a) Update the state of the register A with the following steps:

1) Set $\alpha^{(0)} = a^{(i)}$.

ISO/IEC 18033-4:2005/Amd 1:2009
https://standards.iteh.ai/catalog/standards/sist/be96898d-d1ae-497a-a3ad-659664f04339/iso-iec-18033-4-2005-amd-1-2009

2) For k = 0, 1, 2, 3:

i) Set $f_k = \alpha_1^{(k)} \oplus F(\alpha^{(k)})$ and $r = L(\alpha^{(k)})$.

ii) For j = 0, 1, …,190 set $\alpha_j^{(k+1)} = \alpha_{j+1}^{(k)}$.

iii) Set $\alpha_{191}^{(k+1)} = r$.

3) Set $a^{(i+1)} = \alpha^{(4)}$.

b) Set $\tau^{(0)} = T^{(i)}$, $\beta^{(0)} = b^{(i)}$, $\iota^{(0)} = l^{(i)}$-1.

c) For j = 0, 1, …, $\iota^{(0)}$-1 , set $\beta_j^{(0)} = b_{j+1}^{(i)}$

d) For k = 0, 1, 2, 3:

1) If $\iota^{(0)} = 0$, set $\tau^{(k+1)} = \tau^{(k)}$, *output* = TRUE and $c = f_k$, otherwise update the state of the compression block with the following steps:

i) Set $c = f_k \oplus \tau_0^{(k)}$.

ii) Set $\tau^{(k+1)} = ABSG(\tau^{(k)}, f_k)$.

iii) If $\tau_0^{(k+1)} = 0$, set *output* = TRUE , otherwise set *output* = FALSE.

2) Update the state of the buffer by $(\beta^{(k+1)}, \iota^{(k+1)}) = B(\beta^{(k)}, \iota^{(k)}, output, c)$.

e) Set $T^{(i+1)} = \tau^{(4)}$, $b^{(i+1)} = \beta^{(4)}$ and $l^{(i+1)} = \iota^{(4)}$.

© ISO/IEC 2009 – All rights reserved