INTERNATIONAL STANDARD

ISO/IEC 18033-7

First edition
2022-04

Information security — Encryption algorithms —

Part 7:
Tweakable block ciphers

Sécurité de l'information — Algorithmes de chiffrement —

Partie 7: Chiffrements par blocs paramétrables

Reference number
ISO/IEC 18033-7:2022(E)

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.

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 or www.iec.ch/members_experts/refdocs).

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) or the IEC list of patent declarations received (see patents.iec.ch).

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

A list of all parts in the ISO/IEC 18033 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document specifies tweakable block ciphers. A tweakable block cipher is a family of permutations parametrized by a secret key value and a public tweak value.

Information security — Encryption algorithms —

Part 7:
Tweakable block ciphers

1 Scope

This document specifies tweakable block ciphers. A tweakable block cipher is a family of *n*-bit permutations parametrized by a secret key value and a public tweak value. Such primitives are generic tools that can be used as building blocks to construct cryptographic schemes such as encryption, Message Authentication Codes, authenticated encryption, etc.

A total of five different tweakable block ciphers are defined. They are categorized in Table 1.

Table 1 — Tweakable block ciphers specified

Block length	Tweakey length	Algorithm name
128 bits	256 bits	Deoxys-TBC-256
128 bits	384 bits	Deoxys-TBC-384
64 bits	192 bits	Skinny-64/192
128 bits	256 bits	Skinny-128/256
128 bits	384 bits	Skinny-128/384

2 Normative references

There are no normative references in this document.

3 Terms and definitions

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

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

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

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

3.1
block
string of bits of a defined length

[SOURCE: ISO/IEC 18033-1:2021 3.5]

3.2
ciphertext
data which has been transformed to hide its information content

[SOURCE: ISO/IEC 18033-1:2021, 3.7]

3.3
encryption algorithm
process which transforms plaintext into ciphertext

[SOURCE: ISO/IEC 18033-1:2021, 3.12]

3.4
key
sequence of symbols that controls the operation of a cryptographic transformation (e.g. encryption, decryption)

[SOURCE: ISO/IEC 11770-1:2010, 2.12, modified – the list of cryptographic mechanisms is removed]

3.5
plaintext
unencrypted information

[SOURCE: ISO/IEC 18033-1:2021, 3.20]

3.6
tweak
non-secret sequence of symbols that controls the operation of a cryptographic transformation (e.g. encryption, decryption)

3.7
tweakable block cipher
symmetric encryption system with the property that the encryption algorithm operates on a block of plaintext, i.e. a string of bits of a defined length, and a *tweakey* (3.8) to yield a block of ciphertext

3.8
tweakey
sequence of symbols that controls the operation of a cryptographic transformation (e.g. encryption, decryption)

Note 1 to entry: The tweakey is the concatenation of the key and the tweak inputs.

4 Symbols

k	key bit-length for a tweakable block cipher
Nr	the number of rounds of the tweakable block cipher
n	plaintext/ciphertext bit-length for a tweakable block cipher
t	tweak bit-length for a tweakable block cipher
$a \leftarrow b$	replaces the value of the variable a with the value of the variable b
$\|\|$	concatenation of bit-strings
$\oplus$	bitwise exclusive-OR operation
M	diffusion matrix of the tweakable block cipher
X	n-bit internal state of the tweakable block cipher
GF(i)	finite field of i elements
$\mathbb{K}$	base field as GF(2^8), defined by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$

λ sub-tweakey value

ρ table of rotation values for the ShiftRows / ShiftRowsInv functions of Deoxys-TBC and for the ShiftRowsRight / ShiftRowsRightInv of Skinny

h byte permutation in the tweakey schedule algorithm of Deoxys-TBC

P_T cell permutation in the tweakey schedule algorithm of Skinny

$[i, \dots , j]$ sequence of integers starting from i included, ending at j included, with a step of 1

5 Requirements on the usage of tweakable block ciphers

Both Deoxys-TBC and Skinny ciphers propose a tweakey input that can be utilized as key and/or tweak material, up to the user needs. Therefore, the user can freely choose which part of the tweakey is dedicated to key and/or tweak material. However, whatever the combination of key/tweak size chosen by the user, it shall be such that the key size is at least 128 bits.

In general, the tweak may be made public and a user can repeat the same (tweak,key) combination without causing a security degradation. Some use-cases may require stricter conditions to meet the user's security requirements, and these additional conditions shall always be satisfied.

NOTE Modes of operation offering beyond-birthday security are an example for requiring stricter conditions as they often fail if the same tweak is repeated under the same key.

Skinny-64/192 version shall only be used to instantiate security algorithms guaranteeing an upper bound on the adversarial advantage that remains meaningful as long as the adversary processes less than 2^{64} data blocks.

This document describes the Skinny and Deoxys-TBC configuration where the least-significant portion of the tweakey input is loaded with the tweak and the most-significant portion of the tweakey input is loaded with the key material, i.e. tweakey = key || tweak. Keying material shall never be reused across instances with differing tweak sizes.

Annex A provides numerical examples of Deoxys-TBC-256, Deoxys-TBC-384, Skinny-64/192, Skinny-128/256 and Skinny-128/384. Annex B defines the object identifiers which shall be used to identify the algorithms specified in this document.

6 Deoxys-TBC

6.1 Deoxys-TBC versions

The Deoxys-TBC algorithm (originally published in Reference [4], slightly modified for improved performances in Reference [5], the latter being the version described in this document) is a tweakable block cipher. Deoxys-TBC operates on a plaintext block of 16 bytes numbered from most-significant to least-significant byte [0,…,15]. The internal state X of the cipher is a (4×4) matrix of bytes, initialized from the plaintext block of 16 bytes as follows:

$$X = \begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix}.$$

This document defines two versions of Deoxys-TBC. For Deoxys-TBC-256 the tweakey is of size 256 bits and consists of a key of size $k \geq 128$ and a tweak of size $t = 256\text{-}k$. For Deoxys-TBC-384 the tweakey is of size 384 bits and consists of a key of size $k \geq 128$ and a tweak of size $t = 384\text{-}k$.

6.2 Deoxys-TBC encryption

The number of rounds *Nr* is 14 for Deoxys-TBC-256 and 16 for Deoxys-TBC-384. One round of Deoxys-TBC encryption, similar to a round in the Advanced Encryption Standard (AES)[3], has the following four transformations applied to the internal state in the order specified below:

— **AddSubTweakey(X, λ)**: bitwise exclusive-or (XOR) the 128-bit round sub-tweakey λ (see 6.4) to the internal state X. This function is applied one more time at the end of the last round.

— **SubBytes(X)**: apply the 8-bit AES Sbox S to each of the 16 bytes of the internal state X. The description of this Sbox in hexadecimal notation is presented in Table 2.

Table 2 — The AES Sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

For example, for an input value `53`, then the output value would be determined by the intersection of the row with index '`5`' and the column with index '`3`', which would give `ed`.

— **ShiftRows(X):** rotate the 4-byte i-th row of the internal state X to the left by $\rho[i]$ positions, where $\rho = (0, 1, 2, 3)$.

— **MixBytes(X):** multiply each column of the internal state X by the (4×4) AES maximum distance separable (MDS) matrix M (given below, coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from GF(2)[x]) in $\mathbb{K}$, where $\mathbb{K}$ denotes the base field as GF(2^8) defined by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

$$M = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

The composition MixBytes(ShiftRows(SubBytes(X))) is an unkeyed AES round operating on a state X and is denoted AES_R. The encryption with Deoxys-TBC of a 128-bit plaintext P outputs a 128-bit ciphertext C. Denoting the initial internal state by X_0 and the internal state after round i as X_i, a pseudo-code of the algorithm is as follows:

$X_0 \leftarrow P$

$X_{i+1} \leftarrow$ AES_R(AddSubTweakey(X_i , λ_i)) for i in [0, … , Nr-1]

$C \leftarrow$ AddSubTweakey(X_{Nr} , λ_{Nr})

6.3 Deoxys-TBC decryption

For the decryption, at each round the following four transformations are applied to the internal state in the following order:

— **AddSubTweakey(*X*, *λ*):** XOR the 128-bit round sub-tweakey λ (See 6.4) to the internal state X. This function is applied one more time at the end of the last round.

— **MixBytesInv(*X*):** multiply each column of the internal state X by the (4×4) AES MDS matrix M^{-1} (given below, coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from GF(2)[x]) in $\mathbb{K}$, where $\mathbb{K}$ denotes the base field as GF(2^8) defined by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

$$M^{-1} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix}.$$

— **ShiftRowsInv(*X*)**: rotate the 4-byte i-th row of the internal state X to the right by $\rho[i]$ positions, where ρ = (0, 1, 2, 3).

— **SubBytesInv(*X*)**: apply the 8-bit inverse AES Sbox S_inv to each of the 16 bytes of the internal state X. The description of this Sbox in hexadecimal notation is presented in Table 3.

Table 3 — The AES inverse Sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

For example, for an input value `ed`, then the output value would be determined by the intersection of the row with index '`e`' and the column with index '`d`', which would give `53`.

The composition SubBytesInv(ShiftRowsInv(MixBytesInv(X))) is an unkeyed AES inverse round operating on a state X and is denoted AES_R_Inv. The decryption with Deoxys-TBC of a 128-bit ciphertext C outputs a 128-bit plaintext P. Denoting the initial internal state by X_0 and the internal state after round i as X_i , a pseudo-code of the algorithm is as follows:

$X_0 \leftarrow C$

$X_{i+1} \leftarrow$ AES_R_Inv(AddSubTweakey(X_i , $\lambda_{Nr\ -\ i}$)) for i in [0, ... , Nr-1]

$P \leftarrow$ AddSubTweakey(X_{Nr} , λ_0)

6.4 Deoxys-TBC tweakey schedule

The input tweakey is denoted as T and is divided into words of 128 bits. More precisely, in Deoxys-TBC-256, the size of T is 256 bits with the most-significant 128 bits of T denoted W_2, the second most-significant W_1. For Deoxys-TBC-384, the size of T is 384 bits, with the most-significant 128 bits of T denoted W_3, the second most-significant W_2 and the third most-significant W_1. Finally, λ_i denotes the sub-tweakey (a 128-bit word) that is added to the internal state at round i of the cipher via the AddSubTweakey operation. For Deoxys-TBC-256, a sub-tweakey for round i is defined as (see Figure 1):

$$\lambda_i = \sigma_i^1 \oplus \sigma_i^2 \oplus RC_i,$$

whereas for the case of Deoxys-TBC-384 it is defined as (see Figure 2):

$$\lambda_i = \sigma_i^1 \oplus \sigma_i^2 \oplus \sigma_i^3 \oplus RC_i.$$

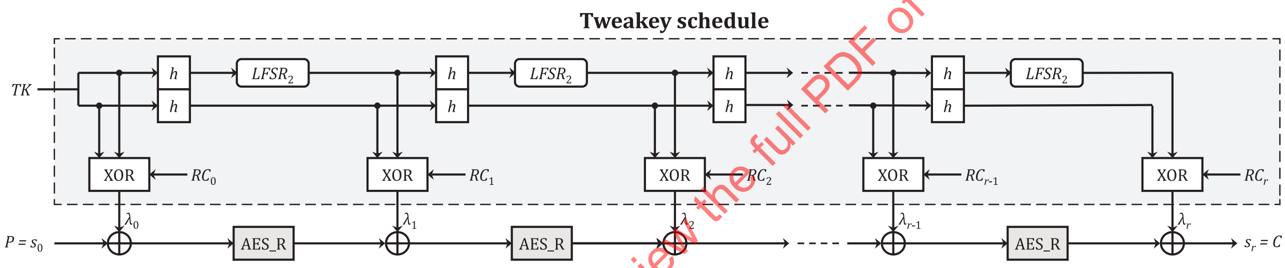

Figure 1 — Deoxys-TBC-256 with its tweakey schedule

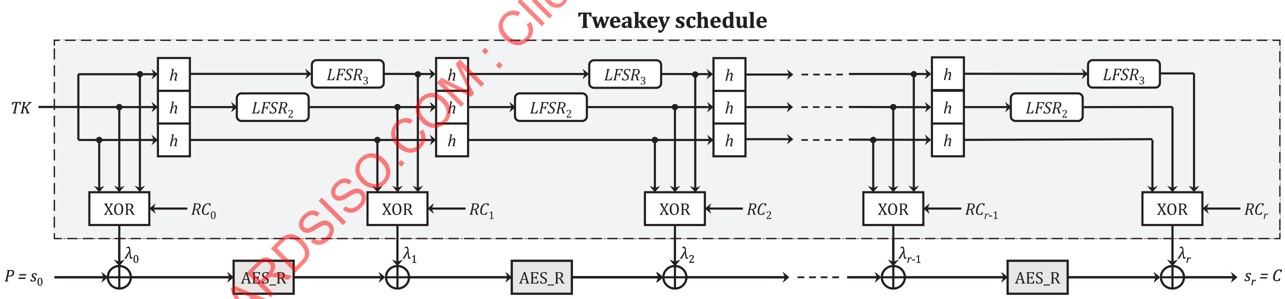

Figure 2 — Deoxys-TBC-384 with its tweakey schedule

The 128-bit words σ_i^1 , σ_i^2 , σ_i^3 are outputs of the tweakey schedule algorithm, initialized with $\sigma_0^1 = W_1$ and $\sigma_0^2 = W_2$ for Deoxys-TBC-256 and with $\sigma_0^1 = W_1$, $\sigma_0^2 = W_2$ and $\sigma_0^3 = W_3$ for Deoxys-TBC-384.

A 16-byte word can be numbered from most-significant to least-significant byte as [0,...,15]. Then, represented as a (4×4) matrix of bytes, the function h permutes the bytes of a word as follows:

$$\begin{bmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 5 & 9 & 13 \\ 6 & 10 & 14 & 2 \\ 11 & 15 & 3 & 7 \\ 12 & 0 & 4 & 8 \end{bmatrix}.$$

The tweakey schedule algorithm then uses two Linear-Feedback Shift Registers (LFSR) and is defined as:

$$\sigma_{i+1}^{1} = h(\sigma_i^{1}),$$

$$\sigma_{i+1}^{2} = h(\,LFSR_2(\sigma_i^{2})),$$

$$\sigma_{i+1}^{3} = h(\,LFSR_3(\sigma_i^{3})),\ \text{in the case of Deoxys-TBC-384,}$$

where the $LFSR_2$ and $LFSR_3$ functions are the application of an LFSR to each of the 16 bytes of a tweakey 128-bit word. More precisely, the two LFSRs used are given in Table 4 (x_0 stands for the most-significant bit of the byte).

Table 4 — LFSRs used in Deoxys-TBC

$\boldsymbol{LFSR_2}$	$(x_0 \,\|\, x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4 \,\|\, x_5 \,\|\, x_6 \,\|\, x_7) \rightarrow (x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4 \,\|\, x_5 \,\|\, x_6 \,\|\, x_7 \,\|\, x_0 \oplus x_2)$
$\boldsymbol{LFSR_3}$	$(x_0 \,\|\, x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4 \,\|\, x_5 \,\|\, x_6 \,\|\, x_7) \rightarrow (x_7 \oplus x_1 \,\|\, x_0 \,\|\, x_1 \,\|\, x_2 \,\|\, x_3 \,\|\, x_4 \,\|\, x_5 \,\|\, x_6)$

Finally, RC_i are the tweakey schedule round constants, and are defined as:

$$RC_i = \begin{bmatrix} 1 & RCON[i] & 0 & 0 \\ 2 & RCON[i] & 0 & 0 \\ 4 & RCON[i] & 0 & 0 \\ 8 & RCON[i] & 0 & 0 \end{bmatrix},$$

where $RCON[i]$ denotes the i + 15-th key schedule constant of the AES. These constants are also given in hexadecimal notation in Table 5.

Table 5 — Constants used in Deoxys-TBC

Rounds	*RCON*[*i*] Constants
i = 0 to i = 16	`2F,5E,BC,63,C6,97,35,6A,D4,B3,7D,FA,EF,C5,91,39,72`

7 Skinny

7.1 Skinny versions

The Skinny algorithm [6] is a tweakable block cipher. Skinny operates on blocks of 16 nibbles or bytes (a total of 64 bits or 128 bits) numbered from most-significant to least-significant as [0,...,15]. The internal state X of the cipher is a (4×4) matrix of nibbles or bytes, initialized from the block of 16 nibbles or bytes as follows:

$$X = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}.$$

This document defines three versions of Skinny-n/t:

— Skinny-64/192 for 64-bit blocks and 192-bit tweakey. The tweakey consists of a key of size $k \geq 128$ and a tweak of size $t = 192\text{-}k$.

— Skinny-128/256 for 128-bit blocks and 256-bit tweakey. The tweakey consists of a key of size $k \geq 128$ and a tweak of size $t = 256\text{-}k$.

— Skinny-128/384 for 128-bit blocks and 384-bit tweakey. The tweakey consists of a key of size $k \geq 128$ and a tweak of size $t = 384\text{-}k$.

7.2 Skinny encryption

The number of rounds Nr is 40 for Skinny-64/192, 48 for Skinny-128/256 and 56 for Skinny-128/384. The internal state of Skinny is a matrix (4×4) of cells, where each cell is of 4-bits when n=64 and 8 bits when n=128. $X[i,j]$ denotes the cell located at row i and column j of the matrix.

One round of Skinny encryption has the following five transformations applied to the internal state in this order (see Figure 3):

— SubCells

— AddConstants

— AddSubTweakey

— ShiftRowsRight

— MixColumns

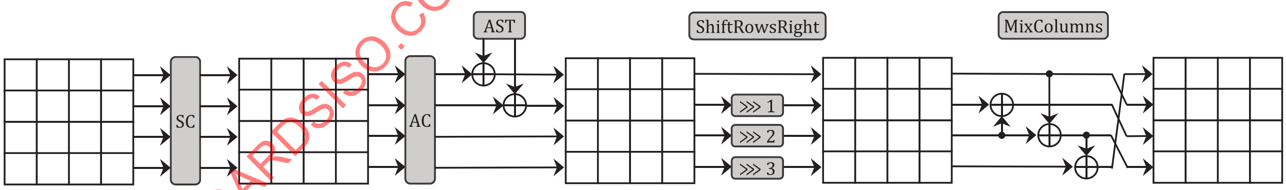

Figure 3 — One round of Skinny

— **SubCells(X):** an Sbox is applied to each cell $X[i,j]$ of the cipher internal state X (a 4-bit Sbox when n=64, an 8-bit Sbox when n=128). The description of these Sboxes in hexadecimal notation is presented in Tables 6 and 7.

Table 6 — The Skinny 4-bit Sbox

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
c	6	9	0	1	a	2	b	3	8	5	d	4	e	7	f

Table 7 — The Skinny 8-bit Sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	65	4c	6a	42	4b	63	43	6b	55	75	5a	7a	53	73	5b	7b
1	35	8c	3a	81	89	33	80	3b	95	25	98	2a	90	23	99	2b
2	e5	cc	e8	c1	c9	e0	c0	e9	d5	f5	d8	f8	d0	f0	d9	f9
3	a5	1c	a8	12	1b	a0	13	a9	05	b5	0a	b8	03	b0	0b	b9
4	32	88	3c	85	8d	34	84	3d	91	22	9c	2c	94	24	9d	2d
5	62	4a	6c	45	4d	64	44	6d	52	72	5c	7c	54	74	5d	7d
6	a1	1a	ac	15	1d	a4	14	ad	02	b1	0c	bc	04	b4	0d	bd
7	e1	c8	ec	c5	cd	e4	c4	ed	d1	f1	dc	fc	d4	f4	dd	fd
8	36	8e	38	82	8b	30	83	39	96	26	9a	28	93	20	9b	29
9	66	4e	68	41	49	60	40	69	56	76	58	78	50	70	59	79
a	a6	1e	aa	11	19	a3	10	ab	06	b6	08	ba	00	b3	09	bb
b	e6	ce	ea	c2	cb	e3	c3	eb	d6	f6	da	fa	d3	f3	db	fb
c	31	8a	3e	86	8f	37	87	3f	92	21	9e	2e	97	27	9f	2f
d	61	48	6e	46	4f	67	47	6f	51	71	5e	7e	57	77	5f	7f
e	a2	18	ae	16	1f	a7	17	af	01	b2	0e	be	07	b7	0f	bf
f	e2	ca	ee	c6	cf	e7	c7	ef	d2	f2	de	fe	d7	f7	df	ff

For example, for an input value `53`, then the output value for the 8-bit Sbox would be determined by the intersection of the row with index '`5`' and the column with index '`3`', which would give `45`.

— **AddConstants(X, i):** a 6-bit LFSR, whose state is denoted (rc_5, rc_4, rc_3, rc_2, rc_1, rc_0) with rc_0 being the least-significant bit, is used to generate round constants. Its update function is defined as: $(rc_5||rc_4||rc_3||rc_2||rc_1||rc_0) \rightarrow (rc_4||rc_3||rc_2||rc_1||rc_0||rc_5 \oplus rc_4 \oplus 1)$.

The six bits are initialized to zero and updated before use in a given round. The bits from the LFSR are arranged into a (4×4) matrix of cells (only the first column of the state is affected by the LFSR bits), depending on the size of internal state:

$$\begin{bmatrix} c_0 & 0 & 0 & 0 \\ c_1 & 0 & 0 & 0 \\ c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

with $c_0 = rc_3||rc_2||rc_1||rc_0$, $c_1 = 0||0||rc_5||rc_4$, $c_2 = 2$ when n=64, and $c_0 = 0||0||0||0||rc_3||rc_2||rc_1||rc_0$, $c_1 = 0||0||0||0||0||0||rc_5||rc_4$, $c_2 = 2$ when n=128.

The round constants are combined with the internal state X, respecting matrix positioning, using bitwise exclusive-or. Precomputed values for the round constants can be found in Table 8.

Table 8 — Round constant used in Skinny

Rounds	Constants
i = 0 to i = 15	`01,03,07,0F,1F,3E,3D,3B,37,2F,1E,3C,39,33,27,0E`
i = 16 to i = 31	`1D,3A,35,2B,16,2C,18,30,21,02,05,0B,17,2E,1C,38`
i = 32 to i = 47	`31,23,06,0D,1B,36,2D,1A,34,29,12,24,08,11,22,04`
i = 48 to i = 55	`09,13,26,0C,19,32,25,0A`

— **AddSubTweakey(X, λ):** XOR the first and second rows of the n-bit round sub-tweakey λ (see 7.4) to the internal state X, respecting the matrix positioning. More formally, for i = {0, 1} and j = {0, 1, 2, 3}: $X[i,j] = X[i,j] \oplus \lambda[i,j]$.

— **ShiftRowsRight(X):** rotate the 4-cell i-th row of the internal state X to the right by $\rho[i]$ positions, where ρ = (0, 1, 2, 3).

— **MixColumns(X):** multiply each column of the cipher internal state matrix X by the following binary matrix M:

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

In other words, for an input column vector of cells [a, b, c, d], the output column vector of cells is [$a \oplus c \oplus d$, a, $b \oplus c$, $a \oplus c$].

The composition MixColumns(ShiftRowsRight(AddSubTweakey(AddConstants(SubCells(X),i),λ))) is a Skinny round i operating on a state X with a sub-tweakey λ and is denoted Skinny_R(X, λ, i). The encryption with Skinny of an n-bit plaintext P outputs an n-bit ciphertext C. Denoting the initial internal state by X_0 and the internal state after round i as X_i, a pseudo-code of the algorithm is as follows:

$X_0 \leftarrow P$

$X_{i+1} \leftarrow$ Skinny_R(X_i , λ_i , i) for i in [0, ... , Nr-1]

$C \leftarrow X_{Nr}$

The final value of the internal state matrix provides the ciphertext with cells being unpacked in the same way as the packing in the initialization.

7.3 Skinny decryption

For the decryption of Skinny, at each round the following five transformations are applied to the internal state in this order:

— MixColumnsInv

— ShiftRowsRightInv

— AddSubTweakey

— AddConstants

— SubCellsInv

— **MixColumnsInv(X):** multiply each column of the cipher internal state matrix X by the following binary matrix M^{-1}:

$$M^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}.$$

In other words, for an input column vector of cells [a, b, c, d], the output column vector of cells is [b, $b \oplus c \oplus d$, $b \oplus d$, $a \oplus d$].

— **ShiftRowsRightInv(X):** rotate the 4-cell i-th row of the internal state X to the left by $\rho[i]$ positions, where ρ = (0, 1, 2, 3).

— **AddSubTweakey(X, λ):** XOR the first and second rows of the n-bit round sub-tweakey λ (see 7.4) to the internal state X, respecting the matrix positioning. More formally, for i = {0, 1} and j = {0, 1, 2, 3}: $X[i,j] = X[i,j] \oplus \lambda[i,j]$.

— **AddConstants(X, i):** identical to the encryption function.

— **SubCellsInv(X):** an Sbox is applied to each cell $X[i,j]$ of the cipher internal state X (a 4-bit Sbox when n=64, an 8-bit Sbox when n=128). The description of these Sboxes in hexadecimal notation is presented in Tables 9 and 10.

Table 9 — The Skinny inverse 4-bit Sbox

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
3	4	6	8	c	a	1	e	9	2	5	7	0	b	d	f

Table 10 — The Skinny inverse 8-bit Sbox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	ac	e8	68	3c	6c	38	a8	ec	aa	ae	3a	3e	6a	6e	ea	ee
1	a6	a3	33	36	66	63	e3	e6	e1	a4	61	34	31	64	a1	e4
2	8d	c9	49	1d	4d	19	89	cd	8b	8f	1b	1f	4b	4f	cb	cf
3	85	c0	40	15	45	10	80	c5	82	87	12	17	42	47	c2	c7
4	96	93	03	06	56	53	d3	d6	d1	94	51	04	01	54	91	d4
5	9c	d8	58	0c	5c	08	98	dc	9a	9e	0a	0e	5a	5e	da	de
6	95	d0	50	05	55	00	90	d5	92	97	02	07	52	57	d2	d7
7	9d	d9	59	0d	5d	09	99	dd	9b	9f	0b	0f	5b	5f	db	df
8	16	13	83	86	46	43	c3	c6	41	14	c1	84	11	44	81	c4
9	1c	48	c8	8c	4c	18	88	cc	1a	1e	8a	8e	4a	4e	ca	ce
a	35	60	e0	a5	65	30	a0	e5	32	37	a2	a7	62	67	e2	e7
b	3d	69	e9	ad	6d	39	a9	ed	3b	3f	ab	af	6b	6f	eb	ef
c	26	23	b3	b6	76	73	f3	f6	71	24	f1	b4	21	74	b1	f4
d	2c	78	f8	bc	7c	28	b8	fc	2a	2e	ba	be	7a	7e	fa	fe
e	25	70	f0	b5	75	20	b0	f5	22	27	b2	b7	72	77	f2	f7
f	2d	79	f9	bd	7d	29	b9	fd	2b	2f	bb	bf	7b	7f	fb	ff

For example, for an input value 45, then the output value for the 8-bit inverse Sbox would be determined by the intersection of the row with index '4' and the column with index '5', which would give 53.

The composition SubCellsInv(AddConstants(AddSubTweakey(ShiftRowsRightInv(MixColumnsInv(X)), λ),i)) is a Skinny round i operating on a state X with a sub-tweakey λ and is denoted Skinny_R_Inv(X, λ, i). The decryption with Skinny of an n-bit ciphertext C outputs an n-bit plaintext P. Denoting the initial internal state by X_0 and the internal state after round i as X_i , a pseudo-code of the algorithm is as follows:

$X_0 \leftarrow C$

$X_{i+1} \leftarrow$ Skinny_R_Inv(X_i , $\lambda_{Nr\text{-}1\text{-}i}$, Nr-1-i) for i in [0, … , Nr-1]

$P \leftarrow X_{Nr}$

The final value of the internal state matrix provides the plaintext with cells being unpacked in the same way as the packing in the initialization.

7.4 Skinny tweakey schedule

The input tweakey is denoted as T and is divided into words of n bits. More precisely, in Skinny-64/192, the size of T is 192 bits with the first most-significant 64 bits of T denoted W_3, the second most-significant W_2 and the third most-significant W_1. For Skinny-128/256, the size of T is 256 bits, with the first most-significant 128 bits of T denoted W_2 and the second most-significant W_1. For Skinny-128/384, the size of T is 384 bits, with the first most-significant 128 bits of T denoted W_3, the second most-significant W_2 and the third most-significant W_1. Finally, λ_i denotes the sub-tweakey (an n-bit word) created at round i, half of which is added to the internal state of the cipher via the AddSubTweakey operation. For Skinny-128/256 a sub-tweakey for round i is defined as:

$$\lambda_i = \sigma_i^1 \oplus \sigma_i^2 ,$$

whereas for the case of Skinny-64/192 and Skinny-128/384 it is defined as:

$$\lambda_i = \sigma_i^1 \oplus \sigma_i^2 \oplus \sigma_i^3 .$$

The n-bit words σ_i^1, σ_i^2, σ_i^3 are outputs of the tweakey schedule algorithm, initialized with $\sigma_0^1 = W_1$ and $\sigma_0^2 = W_2$ for Skinny-128/256 and with $\sigma_0^1 = W_1$, $\sigma_0^2 = W_2$ and $\sigma_0^3 = W_3$ for Skinny-64/192 and Skinny-128/384.

A 16-nibble (or 16-byte) word can be numbered from most-significant to least-significant nibble (or byte) as [0,...,15]. Then, represented as a (4×4) matrix of nibbles (or bytes), the function P_T permutes the nibbles (or bytes) of a word as follows:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \rightarrow \begin{bmatrix} 9 & 15 & 8 & 13 \\ 10 & 14 & 12 & 11 \\ 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{bmatrix}.$$

The tweakey schedule algorithm uses the cell permutation P_T and Linear-Feedback Shift Registers (LFSR). It is defined as:

$$\sigma_{i+1}^1 = P_T(\sigma_i^1),$$

$$\sigma_{i+1}^2 = P_T(LFSR_2(\sigma_i^2)),$$

$$\sigma_{i+1}^3 = P_T(LFSR_3(\sigma_i^3)), \text{ in the case of Skinny-64/192 and Skinny-128/384,}$$

where the $LFSR_2$ and $LFSR_3$ functions are the application of an LFSR to each cell of the two first rows of its n-bit cell matrix input (see Figure 4).

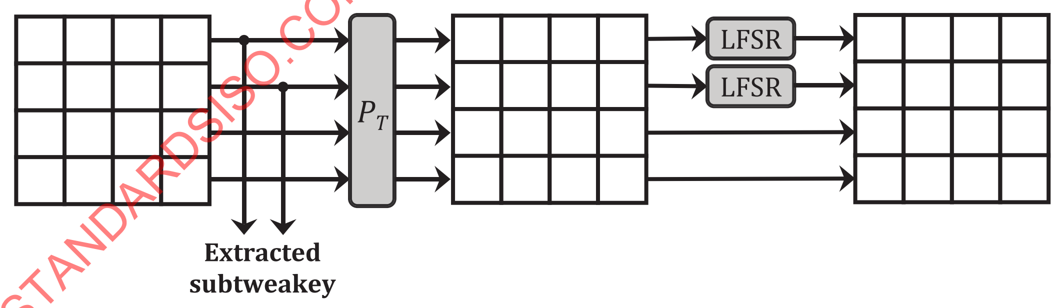

Figure 4 — Tweakey schedule of Skinny

In the case of Skinny-64/192, $LFSR_2$ is using $LFSR_2^4$ and $LFSR_3$ is using $LFSR_3^4$. In the case of Skinny-128/256, $LFSR_2$ is using $LFSR_2^8$ and $LFSR_3$ is not utilized. In the case of Skinny-128/384, $LFSR_2$ is using $LFSR_2^8$ and $LFSR_3$ is using $LFSR_3^8$. These LFSRs are given in Table 11 (x_0 stands for the most-significant bit of the cell).

Table 11 — LFSRs used in Skinny

$LFSR_2^4$	$(x_0 \| x_1 \| x_2 \| x_3) \rightarrow (x_1 \| x_2 \| x_3 \| x_1 \oplus x_0)$
$LFSR_2^8$	$(x_0 \| x_1 \| x_2 \| x_3 \| x_4 \| x_5 \| x_6 \| x_7) \rightarrow (x_1 \| x_2 \| x_3 \| x_4 \| x_5 \| x_6 \| x_7 \| x_0 \oplus x_2)$
$LFSR_3^4$	$(x_0 \| x_1 \| x_2 \| x_3) \rightarrow (x_3 \oplus x_0 \| x_0 \| x_1 \| x_2)$
$LFSR_3^8$	$(x_0 \| x_1 \| x_2 \| x_3 \| x_4 \| x_5 \| x_6 \| x_7) \rightarrow (x_7 \oplus x_1 \| x_0 \| x_1 \| x_2 \| x_3 \| x_4 \| x_5 \| x_6)$

Annex A
(informative)

Numerical examples

A.1 Introduction

This annex provides numerical examples for Deoxys-TBC and Skinny for each block/tweakey length in hexadecimal notation.

A.2 Deoxys-TBC

A.2.1 Deoxys-TBC-256

Tweakey:	Key:	101112131415161718191a1b1c1d1e1f
	Tweak:	02021222324252627000000000000000
Plaintext:	1857d4edf080e8e2c83aa9e794ebf90d	
Ciphertext:	f86ecad0d69d2c573cdeee96c90f37ac	

A.2.2 Deoxys-TBC-384

Tweakey:	Key:	101112131415161718191a1b1c1d1e1f
	Tweak:	202122232425262728292a2b2c2d2e2f
		00010203040506070000000000000000
Plaintext:	d18db1b44ad16fe5623ccd73c250c272	
Ciphertext:	e94c5c6df7c19474bbdd292baa2555fd	

A.3 Skinny

A.3.1 Skinny-64/192

Tweakey:	Key:	ed00c85b120d6861
		8753e24bfd908f60
	Tweak:	b2dbb41b422dfcd0
Plaintext:	530c61d35e8663c3	
Ciphertext:	dd2cf1a8f330303c	

A.3.2 Skinny-128/256

Tweakey:	Key:	009cec81605d4ac1d2ae9e3085d7a1f3