INTERNATIONAL STANDARD

ISO/IEC 18033-4

Second edition
2011-12-15

Information technology — Security techniques — Encryption algorithms —

Part 4: Stream ciphers

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

Partie 4: Chiffrements en flot

Reference number
ISO/IEC 18033-4:2011(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. 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 18033-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 18033-4:2005), which has been technically revised. It also incorporates the Amendment ISO/IEC 18033-4:2005/Amd.1:2009.

ISO/IEC 18033 consists of the following parts, under the general title *Information technology — Security techniques — Encryption algorithms*:

— *Part 1: General*

— *Part 2: Asymmetric ciphers*

— *Part 3: Block ciphers*

— *Part 4: Stream ciphers*

Introduction

This part of ISO/IEC 18033 includes stream cipher algorithms. A stream cipher is an encryption mechanism that uses a keystream to encrypt a plaintext in a bitwise or a block-wise manner. There are two types of stream ciphers: a synchronous stream cipher, in which the keystream is generated from only the secret key (and an initialization vector) and a self-synchronizing stream cipher, in which the keystream is generated from the secret key and some past ciphertexts (and an initialization vector). This part of ISO/IEC 18033 describes both pseudorandom number generators for producing keystream and output functions to combine a keystream with plaintext.

This part of ISO/IEC 18033 includes two output functions:

— Binary-additive output function; and

— MULTI-S01 output function.

This part of ISO/IEC 18033 includes five dedicated keystream generators:

— MUGI keystream generator;

— SNOW 2.0 keystream generator;

— Rabbit keystream generator;

— Decim^{v2} keystream generator; and

— KCipher-2 (K2) keystream generator.

Information technology — Security techniques — Encryption algorithms —

Part 4: Stream ciphers

1 Scope

This part of ISO/IEC 18033 specifies

a) output functions to combine a keystream with plaintext,

b) keystream generators for producing keystream, and

c) object identifiers assigned to dedicated keystream generators in accordance with ISO/IEC 9834.

NOTE 1 The list of assigned object identifiers is given in Annex A.

NOTE 2 Any change to the specification of these algorithms resulting in a change of functional behaviour will result in a change of the object identifier assigned to the algorithms concerned.

2 Normative references

The following referenced documents are 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 18033-1, *Information technology — Security techniques — Encryption algorithms — Part 1: General*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18033-1 and the following apply.

3.1
big-endian
method of storage of multi-byte numbers with the most significant bytes at the lowest memory addresses

[ISO/IEC 10118-1:2000]

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

[ISO/IEC 10116:2006]

3.3
confidentiality
property that information is not made available or disclosed to unauthorized individuals, entities, or processes

3.4
data integrity
property that data has not been altered or destroyed in an unauthorized manner

[ISO/IEC 9797-1:2011]

3.5
decryption
reversal of a corresponding encryption

[ISO/IEC 10116:2006]

3.6
encryption
reversible transformation of data by a cryptographic algorithm to produce ciphertext, i.e., to hide the information content of the data

[ISO/IEC 9797-1:2011]

3.7
initialization value
value used in defining the starting point of an encryption process

3.8
key
sequence of symbols that controls the operation of a cryptographic transformation (e.g., encryption, decryption, cryptographic check function computation, signature generation, or signature verification)

[ISO/IEC 11770-1:2010]

3.9
keystream function
function that takes as input, the current state of the keystream generator and (optionally) part of the previously generated ciphertext, and gives as output the next part of the keystream

3.10
keystream generator
state-based process (i.e., a finite state machine) that takes as input, a key, an initialization vector, and if necessary the ciphertext, and gives as output a keystream (i.e., a sequence of bits or blocks of bits) of arbitrary length

3.11
***n*-bit block cipher**
block cipher with the property that plaintext blocks and ciphertext blocks are n bits in length

[ISO/IEC 10116:2006]

3.12
next-state function
function that takes as input, the current state of the keystream generator and (optionally) part of the previously generated ciphertext, and gives as output a new state for the keystream generator

3.13
output function
function that combines the keystream and the plaintext to produce the ciphertext

NOTE This function is often bitwise XOR.

3.14
padding
appending extra bits to a data string

[ISO/IEC 10118-1:2000]

3.15
plaintext
unencrypted information

[ISO/IEC 9797-1:2011]

3.16
secret key
key used with symmetric cryptographic techniques by a specified set of entities

[ISO/IEC 11770-3:2008]

3.17
state
current internal state of a keystream generator

4 Symbols and abbreviated terms

4.1 Symbols

$0x$	Prefix for hexadecimal values.
$0^{(n)}$	n-bit variable where 0 is assigned to every bit.
AND	Bitwise logical AND operation.
$Am^{(i)}[Y]$	The Y-th bit of the register $Am^{(i)}$ in KCipher-2 (K2).
a_i	Variables in an internal state of a keystream generator.
b_i	Variables in an internal state of a keystream generator.
CFB	Cipher FeedBack mode of a block cipher.
CTR	Counter mode of a block cipher.
C_i	Ciphertext block.
D_i	64-bit constants used for MUGI.
e_K	Symmetric block cipher encryption function using secret key K.
F	Subfunction used for MUGI.
FSM	Subfunction used for SNOW 2.0.
$GF(2^n)$	Finite field of exactly 2^n elements.
$GF(2^n)[x]$	The polynomial ring over the finite field $GF(2^n)$.

Init	Function which generates the initial internal state of a keystream generator.
IV	Initialization vector.
IK	Internal key used for KCipher-2 (K2).
K	Key.
M	Subfunction used for MUGI.
Next	Next-state function of a keystream generator.
NLF	Nonlinear function used for KCipher-2 (K2).
n	Block length.
OFB	Output FeedBack mode of a block cipher.
OR	Bitwise logical OR operation.
Out	Output function combining keystream and plaintext in order to generate ciphertext.
P	Plaintext.
P_i	Plaintext block.
R	Additional input to Out.
S_R	Subfunction used for MUGI.
Strm	Keystream function of a keystream generator.
SUB	Lookup table used for MUGI and SNOW 2.0.
Sub_{K2}	Subfunction used for KCipher-2 (K2).
S_i	Internal state of a keystream generator.

NOTE During normal operation of the cipher, i will increase monotonically starting from zero. However, during initialization of the ciphers, it is convenient from a notational point of view to let i take negative values and define the starting state S_0 in terms of values of S_i for i <0.

T	Subfunction used for SNOW 2.0.
Z	Keystream.
Z_i	Keystream block.
α_{MUL}	Lookup table used for SNOW 2.0.
$\alpha_{\text{MUL}0}$	Lookup table with index 0 used for KCipher-2 (K2).
$\alpha_{\text{MUL}1}$	Lookup table with index 1 used for KCipher-2 (K2).
$\alpha_{\text{MUL}2}$	Lookup table with index 2 used for KCipher-2 (K2).

α_{MUL3}	Lookup table with index 3 used for KCipher-2 (K2).
$\alpha_{\mathrm{inv\_MUL}}$	Inverse lookup table used for SNOW 2.0.
ρ_1	Subfunction used for MUGI.
λ_1	Subfunction used for MUGI.
$\lceil x \rceil$	The smallest integer greater than or equal to the real number x.
$\neg x$	Bitwise complement operation.
$\bullet$	Polynomial multiplication.
$\Vert$	Bit concatenation.
$+_m$	Integer addition modulo 2^m.
$\oplus$	Bitwise XOR (eXclusive OR) operation.
$\otimes$	Operation of multiplication of elements in the finite field $GF(2^n)$.

EXAMPLE $C = A \otimes B$: In this operation, the finite field is represented as a selected irreducible polynomial $F(x)$ of degree n with binary coefficients, the n-bit blocks $A = (a_0,a_1,\dots,a_{n-1})$ and $B = (b_0,b_1,\dots,b_{n-1})$ (where the a_i and b_i are bits) are represented as the polynomials, $A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1$ and $B(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \dots + b_0$ respectively, then let $C(x) = A(x) \bullet B(x) \bmod F(x)$, i.e., $C(x)$ is the polynomial of degree at most $n-1$ obtained by multiplying $A(x)$ and $B(x)$, dividing the result by $F(x)$, and then taking the remainder. If $C(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \dots + c_0$ (where the c_i are bits) then let C be the n-bit block $(c_0,c_1,\dots,c_{n-1})$.

$\boxplus$	Modular addition operation
$<<_n t$	t-bit left shift in an n-bit register.
$>>_n t$	t-bit right shift in an n-bit register.
$<<<_n t$	t-bit left circular rotation in an n-bit register.
$>>>_n t$	t-bit right circular rotation in an n-bit register.

4.2 Functions

4.2.1 Left-truncation of bits

The operation of selecting the j leftmost bits of an array A=($a_0,a_1,\dots,a_{m-1}$) to generate a j-bit array is written

$$(j \sim A)=(a_0,a_1,\dots,a_{j-1})$$

This operation is defined only when $1 \le j \le m$.

See ISO/IEC 10116:2006.

4.2.2 Shift operation

The operation *Shift* is defined as follows: Given an n-bit variable X and a k-bit variable V where $1 \leq k \leq n$, the effect of the shift function Shift is to produce the n-bit variable

$$Shift_k(X \mid V) = (x_k, x_{k+1}, \ldots, x_{n-1}, v_0, v_1, \ldots, v_{k-1}) \qquad (k < n)$$

$$Shift_k(X \mid V) = (v_0, v_1, \ldots, v_{k-1}) \qquad (k = n)$$

The effect is to shift the bits of array X left by k places, discarding $x_0, x_1, \ldots, x_{k-1}$ and to place the array V in the rightmost k places of X. When $k = n$ the effect is to totally replace X by V.

See ISO/IEC 10116:2006.

4.2.3 Variable *I*(*k*)

The variable $I(k)$ is a k-bit variable where 1 is assigned to every bit.

5 Framework for stream ciphers

This clause contains a high-level description of a framework for the stream ciphers specified in this part of ISO/IEC 18033. A detailed description of the general model for a stream cipher is provided in Clause 6. A stream cipher specified in this part of ISO/IEC 18033 is defined by the specification of the following processes:

a) The keystream generator, which may be either

— a Synchronous keystream generator, or

— a Self-synchronizing keystream generator.

NOTE 1 Block cipher modes of operation are methods by which a block cipher can be used to construct a keystream generator. These modes are standardised in ISO/IEC 10116, and the meaning of the functions used in the specification is defined in 6.2.1 and 6.2.2.

NOTE 2 Block ciphers are defined in this part of ISO/IEC 18033.

b) The output function, which may be either

— the Binary-additive output function, or

— the MULTI-S01 output function.

6 General models for stream ciphers

6.1 Keystream generators

6.1.1 Synchronous keystream generators

A synchronous keystream generator is a finite-state machine. It is defined by:

a) An initialization function, *Init*, which takes as input a key K and an initialization vector IV, and outputs an initial state S_0 for the keystream generator. The initialization vector should be chosen so that no two messages are ever encrypted using the same key and the same IV.

b) A next-state function, *Next*, which takes as input the current state of the keystream generator S_i, and outputs the next state of the keystream generator S_{i+1}.

c) A keystream function, *Strm*, which takes as input a state of the keystream generator S_i, and outputs a keystream block Z_i.

When the synchronous keystream generator is first initialized, it will enter an initial state S_0 defined by:

$S_0 = Init(IV, K)$.

On demand the synchronous keystream generator will, for i=0,1,...:

a) Output a keystream block $Z_i = Strm(S_i, K)$.

b) Update the state of the machine $S_{i+1} = Next(S_i, K)$.

Therefore to define a synchronous keystream generator it is only necessary to specify the functions *Init*, *Next* and *Strm*, including the lengths and alphabets of the key, the initialization vector, the state, and the output block.

6.1.2 Self-synchronizing keystream generators

Generation of a keystream for a self-synchronizing stream cipher is dependent only on previous ciphertexts, the key, and the initialization vector. A general model for a keystream generator for a self-synchronizing stream cipher is now defined:

a) An initialization function, *Init*, which takes as input a key K and an initialization vector IV and outputs an internal input for the keystream generator S and r dummy ciphertext blocks $C_{-1}, C_{-2}, \ldots, C_{-r}$.

b) A keystream function, *Strm*, that takes as input S and r ciphertext blocks $C_{i-1}, C_{i-2}, \ldots, C_{i-r}$, and outputs a keystream block Z_i.

To define a self-synchronizing keystream generator it is only necessary to specify the number of feedback blocks r and the functions *Init* and *Strm*.

NOTE A self-synchronizing stream cipher differs from a synchronous stream cipher in that the keystream depends only on previous ciphertext, the initialization vector and the key, i.e., the keystream generator operates in a stateless fashion. As a result, a decryptor for such a cipher can recover from loss of synchronization after receiving sufficient ciphertext blocks. This also means that the method of keystream generation is dependent upon the selected output function *Out*, which is typically the bitwise XOR operation.

6.2 Output functions

6.2.1 General model of output function

6.2 specifies two stream cipher output functions, i.e., techniques to be used in a stream cipher to combine a keystream with plaintext to derive ciphertext.

An output function for a synchronous or a self-synchronizing stream cipher is a function *Out* that combines a plaintext block P_i, a keystream block Z_i, and some other input R if necessary to give a ciphertext block C_i ($i \geq 0$). A general model for a stream cipher output function is now defined:

Encryption of a plaintext block P_i by a keystream block Z_i is given by:

$C_i = Out(P_i, Z_i, R)$,

and decryption of a ciphertext block C_i by a keystream block Z_i is given by:

$$P_i = Out^{-1}(C_i, Z_i, R).$$

The output function shall satisfy that for any keystream block Z_i, plaintext block P_i, and other input R,

$$P_i = Out^{-1}(Out\,(P_i, Z_i, R), Z_i, R).$$

6.2.2 Binary-additive output function

A binary-additive stream cipher is a stream cipher in which the keystream, plaintext, and ciphertext blocks are strings of binary digits, and the operation to combine plaintext with keystream is bitwise XOR. The operation *Out* takes two inputs and does not use any additional information R for calculation. Let n to be the bit length of P_i. This function is specified by

$$Out(P_i, Z_i, R) = P_i \oplus Z_i.$$

The operation *Out* $^{-1}$ is specified by

$$Out^{-1}(C_i, Z_i, R) = C_i \oplus Z_i.$$

NOTE The binary-additive stream cipher does not provide any integrity protection for encrypted data. If data integrity is required, either the MULTI-S01 output function or a separate integrity mechanism should be used, such as a MAC, i.e., a Message Authentication Code (such mechanisms are specified in ISO/IEC 9797).

6.2.3 MULTI-S01 output function

a) General model of MULTI-S01

MULTI-S01 is an output function for a synchronous stream cipher that supports both data confidentiality and data integrity. The MULTI-S01 encryption operation is suitable for use in an online environment. However, the decryption operation of MULTI-S01 can only be performed in an offline situation, as the integrity check is only performed after receiving all the ciphertext blocks. MULTI-S01 has a security parameter n. The computation of Out depends on the choice of a field $GF(2^n)$, i.e., on the choice of an irreducible polynomial over $GF(2)$ of degree n. The MULTI-S01 function only accepts messages whose length is a multiple of n. To encrypt messages whose length is not a multiple of n, a padding mechanism *Pad*(M) is required.

NOTE 1 The redundancy R is generated in such a way that the sender and the receiver share it. R can be a fixed public value like 0x00...0.

b) The encryption function *Out*(P, Z, R)

Input: $n \cdot u$ -bit plaintext P, keystream $Z = (Z_0, Z_1, ...)$, where Z_i are n-bit blocks, n-bit redundancy R.

Output: Ciphertext C.

1) Let t be the lowest value of i ($i \geq 0$) such that $Z_i \neq 0^{(n)}$.

2) Let $(P_0, P_1, ..., P_{u-1}) = P$, where P_i is an n-bit block.

3) Set $P_u = Z_{t+u+3}$.

4) Set $P_{u+1} = R$.

5) For each P_i, do the following calculations (for $i = 0, 1, \ldots, u + 1$):

— Let $W_i = P_i \oplus Z_{t+i+1}$.

— Let $X_i = Z_t \otimes W_i$ (in $GF(2^n)$).

— Let $C_i = X_i \oplus W_{i-1}$, where W_{i-1} is the W value of the previous block $i - 1$, and $W_{-1} = 0^{(n)}$.

— Set $C = C_0 \,||\, C_1 \,||\, \ldots \,||\, C_{u+1}$.

— Output C.

Figure 1 shows the block diagram of *Out* function.

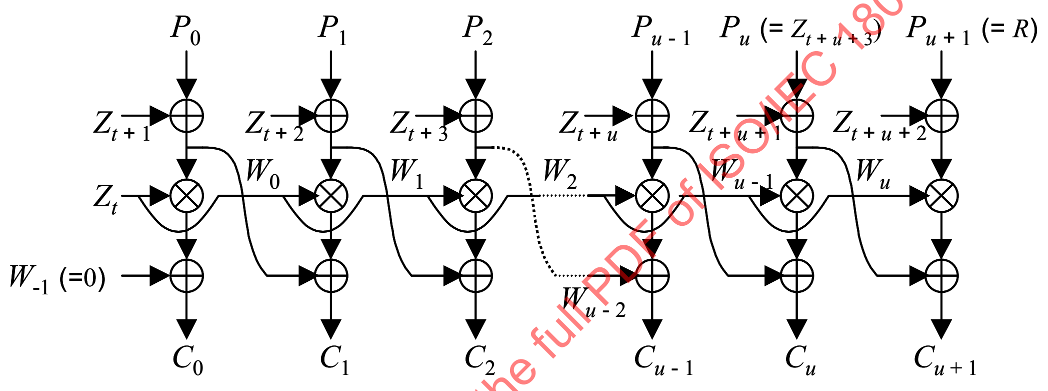

Figure 1 — *Out* function of MULTI-S01 mode

NOTE 2 The irreducible polynomial used to define multiplication in the field depends on n. For instance, in the case of $n = 64$ and 128, the irreducible polynomial $x^{64} + x^4 + x^3 + x + 1$ and $x^{128} + x^7 + x^2 + x + 1$ can be used.

c) The decryption function $Out^{-1}(P, Z, R)$

Input: $n \cdot v$-bit ciphertext C, keystream Z, n-bit redundancy R.

Output: Plaintext P or "*reject*".

1) Let t be the lowest value of i ($i \geq 0$) such that $Z_i \neq 0^{(n)}$.

2) Let $(C_0, C_1, \ldots, C_{v-1}) = C$, where C_i is an n-bit block.

3) For each C_i, do the following calculations (for $i = 0, 1, \ldots, v - 1$):

— Let $X_i = C_i \oplus W_{i-1}$, where $W_{-1} = 0^{(n)}$.

— Let $W_i = Z_t^{-1} \otimes X_i$ (in $GF(2^n)$).

— Let $P_i = W_i \oplus Z_{t+i+1}$.

4) If $P_{v-2} = Z_{t+v+1}$ and $P_{v-1} = R$, output $P = P_0 \,||\, P_1 \,||\ldots||\, P_{v-3}$ as plaintext. Otherwise, output the special symbol meaning "*reject*" without any text.

Figure 2 shows the block diagram of *Out*$^{-1}$ function.

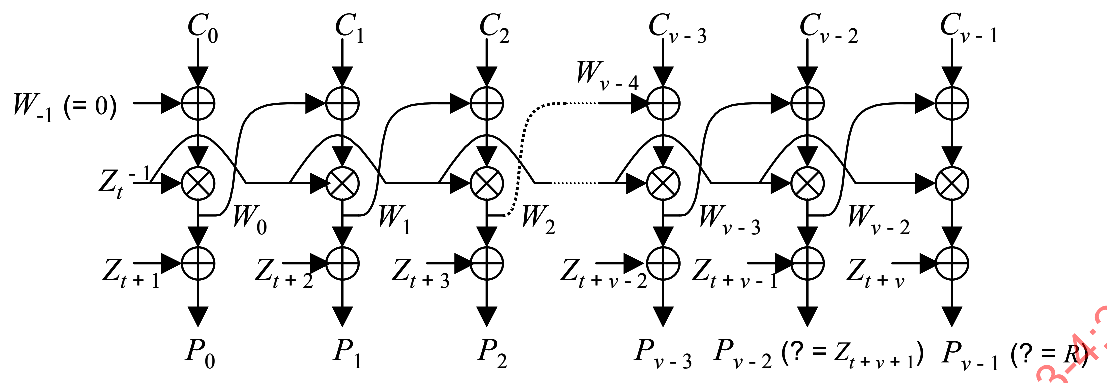

Figure 2 — *Out*$^{-1}$ function of MULTI-S01 mode

d) Padding mechanism *Pad*(M)

Only when lengths of input messages are not multiples of n, the following padding mechanism *Pad*(M) is excecuted:

Input: ($n\ v + c$)-bit string M, where v is a non-negative integer and $0 \leq c < n$.

Output: Padded plaintext P.

1) Pad a bit string "1" at the end of the message.

2) Pad ($n - c - 1$)-bit string $0^{(n-c-1)}$ to the string generated by step a).

3) Output the whole data string in length of ($n\ v + n$) bits.

NOTE 3 If the length of the message is a multiple of n in an environment where the length is not certain to be so, this padding mechanism is recommended.

NOTE 4 In order to unpad the message, remove consecutive 0 bits at the end of the data, and remove another bit "1".

7 Constructing keystream generators from block ciphers

7.1 Block cipher modes for a synchronous keystream generator

7.1.1 The OFB (Output FeedBack) mode and the CTR (Counter) mode

Subclause 7.1 specifies two n-bit block cipher modes for a synchronous keystream generator. They are the OFB (Output FeedBack) mode and the CTR (Counter) mode of an n-bit block cipher e_K.

7.1.2 OFB mode

The OFB mode is defined by one parameter r, $1 \leq r \leq n$, which is the size of a plaintext and ciphertext block.

The initialization vector IV is an n-bit string. IV shall be generated differently for two encryptions with the same key K. The functions *Init*, *Next* and *Strm* are specified as follows:

— *Init*(IV, K) = IV.

— *Next*(S_i, K) = $e_K(S_i)$.

— *Strm*(S_i) = ($r \sim S_i$).

NOTE *Init*(IV, K) = IV, is equivalent to $S_0 = IV$.

In case of the OFB mode, the binary-additive output function defined in 6.2.2 is used. Figure 3 shows the block diagram of a keystream generator based on CFB mode.

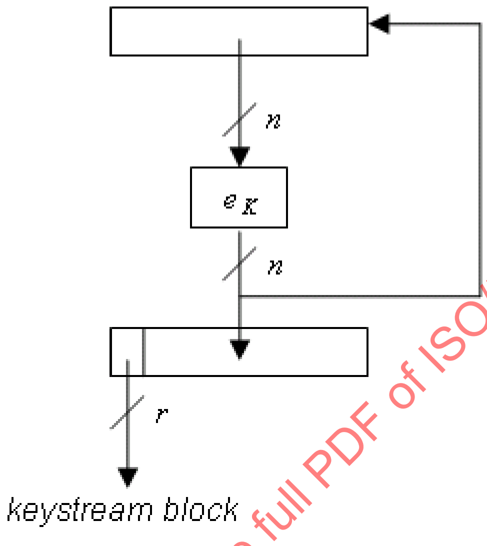

Figure 3 — Keystream generation based on OFB mode

7.1.3 CTR mode

The CTR mode is defined by one parameter r, $1 \le r \le n$, which is the size of a plaintext and ciphertext block.

The initialization vector IV is an n-bit string. It shall be assured that $S_i \neq S_j$' for two keystreams S_0, S_1, S_2, ... and S_0', S_1', S_2', ... generated with the same key K. The functions *Init*, *Next* and *Strm* are specified as follows:

— *Init*(IV, K) = IV.

— *Next*(S_i, K) = $S_i + 1 \bmod 2^n$.

— *Strm*(S_i, K) = ($r \sim e_K(S_i)$).

NOTE *Init*(IV, K) = IV, is equivalent to $S_0 = IV$.

In case of the CTR mode, the binary-additive output function defined in 6.2.2 is used. Figure 4 shows the block diagram of a keystream generator based on CFB mode.

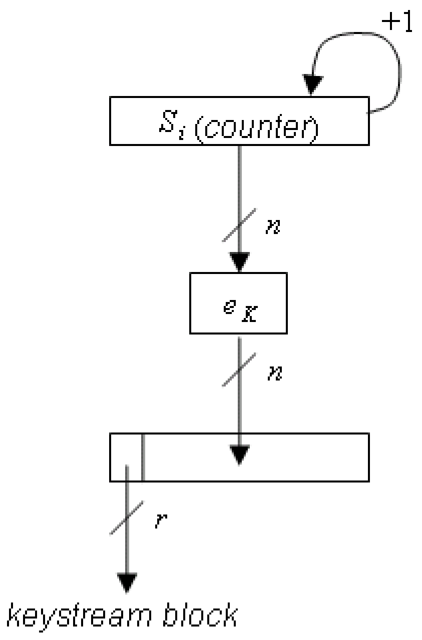

Figure 4 — Keystream generation based on CTR mode

7.2 Block cipher mode for a self-synchronizing keystream generator

7.2.1 Introduction to the CFB mode

The CFB mode of an n-bit block cipher is a self-synchronizing stream cipher.

7.2.2 CFB mode

The CFB (Cipher FeedBack) mode is defined by three parameters, i.e., the size j of feedback buffer S_i, where $n \le j \le 1024n$, the size b of feedback variable, where $1 \le b \le n$ and the size r of the output block, where $1 \le r \le b$.

NOTE 1 The value b-r shall be small compared to b.

The initialization vector IV shall be a randomly generated j-bit string and also shall be generated differently for two encryptions with the same key K. The functions *Init*, *Next* and *Strm* are specified as follows:

— *Init*(IV, K) = IV.

— *Next*(S) = *Shift*$_b$(S | *Shift*$_r$(I (b)| C_i)).

— *Strm*(S, K) = ($r \sim e_K$ (($n \sim S$))).

NOTE 2 *Init*(IV, K) = IV, is equivalent to $S_0 = IV$.

In case of the CFB mode, the binary-additive output function defined in 6.2.2 is used. Figure 5 shows the block diagram of a keystream generator based on CFB mode.

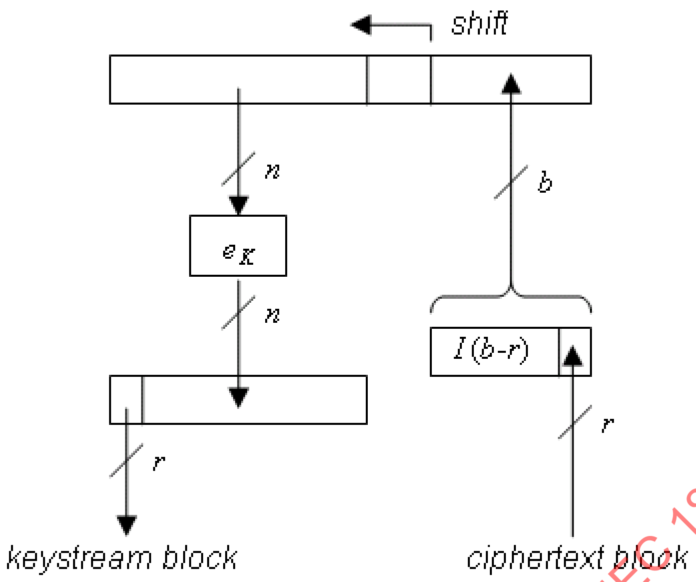

Figure 5 — Keystream generation based on CFB mode

8 Dedicated keystream generators

8.1 MUGI keystream generator

8.1.1 Introduction to MUGI

MUGI is a keystream generator which uses a 128-bit secret key K, a 128-bit initialization vector IV, and a state variable S_i $(i \geq 0)$ consisting of 19 64-bit blocks (note that the term block is used through the specification of MUGI for a 64-bit block), and outputs a keystream block Z_i at every iteration of the function *Strm*.

NOTE This keystream generator is originally proposed in [17].

The state variable S_i is sub-divided into a combination of a 3-block variable:

$$a^{(i)} = (a_0^{(i)}, a_1^{(i)}, a_2^{(i)}),$$

where $a_j^{(i)}$ is a block (for $j = 0, 1, 2$), and a 16-block variable:

$$b^{(i)} = (b_0^{(i)}, b_1^{(i)}, \ldots, b_{15}^{(i)}),$$

where $b_j^{(i)}$ is a block (for $j = 0, 1, \ldots, 15$).

The *Init* function, defined in detail in 8.1.2, takes as input the 128-bit key K and the 128-bit initializing vector IV, and produces the initial value of the state variable $S_0 = (a^{(0)}, b^{(0)})$.

The *Next* function, defined in detail in 8.1.3, takes as input the 19-block state variable $S_i = (a^{(i)}, b^{(i)})$ and produces as output the next value of the state variable $S_{i+1} = (a^{(i+1)}, b^{(i+1)})$.

The *Strm* function, defined in detail in 8.1.4, takes as input the 19-block state variable $S_i = (a^{(i)}, b^{(i)})$ and produces as output the keystream block Z_i.

Note that the *Next* function is defined in terms of the functions ρ_1 and λ_1 which are defined in 8.1.5 and 8.1.6, respectively. The function ρ_1 is defined in terms of a function *F* which is defined in 8.1.7

There are three constants used in MUGI, D_0 in the initialization function Init, and D_1, D_2 in ρ_1. These are given by:

D_0 = `0x6A09E667F3BCC908`,

D_1 = `0xBB67AE8584CAA73B`,

D_2 = `0x3C6EF372FE94F82B`.

8.1.2 Initialization function *Init*

The initialization of MUGI is divided into eight steps. The left- and right-half blocks of K are denoted by K_0 and K_1 respectively. IV_0 and IV_1 are defined in the same manner. The initialization function Init is as follows:

Input: 128-bit key K, 128-bit initialization vector IV.

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

a) Set the key K into the part of the state variable $a^{(-49)}$ as follows:

— Set $(K_0, K_1) = K$, where K_i is 64 bits for $i = 0,1$.

— Set $a_0^{(-49)} = K_0$.

— Set $a_1^{(-49)} = K_1$.

— Set $a_2^{(-49)} = (K_0 <<<_{64} 7) \oplus (K_1 >>>_{64} 7) \oplus D_0$.

D_0 in the above equation is a constant (see 8.1).

b) For $i = -49, -48, ..., -34$, set $a^{(i+1)} = \rho_1(a^{(i)}, 0^{(64)}, 0^{(64)})$. For the description of ρ_1 see 8.1.5.

c) For $i = 0, 1, ..., 15$, set $b_{15-i}^{(-16)} = a_0^{(i-48)}$.

d) Add the initialization vector IV into the state a as follows:

— Set $IV_0 \,||\, IV_1 = IV$, where IV_i is a block.

— Set $a_0^{(-32)} = a_0^{(-33)} \oplus IV_0$.

— Set $a_1^{(-32)} = a_1^{(-33)} \oplus IV_1$.

— Set $a_2^{(-32)} = a_2^{(-33)} \oplus (IV_0 <<<_{64} 7) \oplus (IV_1 >>>_{64} 7) \oplus D_0$.

e) For $i = -32, -31, ..., -17$, set $a^{(i+1)} = \rho_1(a^{(i)}, 0^{(64)}, 0^{(64)})$.

f) Set $S_{-16} = (a^{(-16)}, b^{(-16)})$.

g) Iterate the update function *Next* 16 times:

Set S_0 = *Next*^{16} (S_{-16}),

where *Next*^{16} stands for 16 iterations of the next-state function *Next*.

h) Output S_0.

8.1.3 Next-state function *Next*

The next-state function of MUGI is described as a combination of ρ_1 and λ_1 The next-state function *Next* of MUGI is as follows:

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

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

- Set $a^{(i+1)} = \rho_1(a^{(i)}, b_4^{(i)}, b_{10}^{(i)})$. The detailed description of the function ρ_1 is given in 8.1.5.
- Set $b^{(i+1)} = \lambda_1(b^{(i)}, a_0^{(i)})$. The detailed description of the function λ_1 is given in 8.1.6.
- Set $S_{i+1} = (a^{(i+1)}, b^{(i+1)})$.
- Output S_{i+1}.

8.1.4 Keystream function *Strm*

The keystream function *Strm* is as follows:

Input: State variable S_i.

Output: Keystream block Z_i.

- Set $Z_i = a_2^{(i)}$.
- Output Z_i.

8.1.5 Function ρ_1

The function ρ_1 is as follows:

Input: State variable $a^{(i)}$, two 64-bit parameters w_1, w_2.

Output: The next value of the state variable $a^{(i+1)}$.

- Set $a_0^{(i+1)} = a_1^{(i)}$.
- Set $a_1^{(i+1)} = a_2^{(i)} \oplus F(a_1^{(i)}, w_1) \oplus D_1$.
- Set $a_2^{(i+1)} = a_0^{(i)} \oplus F(a_1^{(i)}, (w_2 <<<_{64} 17)) \oplus D_2$.
- Output $a^{(i+1)}$.

D_1, D_2 are constants (see 8.1 for details).

Figure 6 shows the block diagram of the function ρ_1. The detailed description of the function *F* is given in 8.1.7.

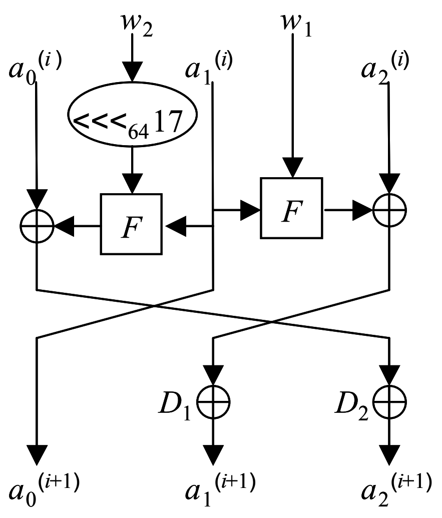

Figure 6 — ρ_1 function of MUGI

8.1.6 Function λ_1

The function λ_1 is as follows:

Input: State variable $b^{(i)}$, 64-bit parameter a'.

Output: The next value of the state variable $b^{(i+1)}$.

— Set $b_j^{(i+1)} = b_{j-1}^{(i)}$ for $j \neq 0, 4, 10$.

— Set $b_0^{(i+1)} = b_{15}^{(i)} \oplus a'$.

— Set $b_4^{(i+1)} = b_3^{(i)} \oplus b_7^{(i)}$.

— Set $b_{10}^{(i+1)} = b_9^{(i)} \oplus (b_{13}^{(i)} <<<_{64} 32)$.

Output $b^{(i+1)}$.

8.1.7 Function *F*

Function *F* uses operations over the finite field $GF(2^8)$. In the polynomial representation, $GF(2^8)$ is realized as $GF(2)[x] / f(x)$, where $f(x)$ is an irreducible polynomial of degree 8 defined over $GF(2)$. The MUGI keystream generator uses the following irreducible polynomial:

$$f(x) = x^8 + x^4 + x^3 + x + 1.$$

The function *F* is the composition of a key addition (the data addition from the part of state variable b), a non-linear transformation using the function S_R, a linear transformation using the matrix M and byte shuffling (see Figure 7).

Let us denote the input and the output to the *F* function as X and Y respectively. Then the function *F* is as follows:

Input: Two 64-bit strings X and T.

Output: 64-bit string Y.

— $X' = X \oplus T$.

— Set $(X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7) = X'$, where X_i is an 8-bit string.

— Set $P_i = S_R(X_i)$ for $i = 0, 1, ..., 7$.

— Set $P_L = P_0 \| P_1 \| P_2 \| P_3$.

— Set $P_R = P_4 \| P_5 \| P_6 \| P_7$.

— Set $Q_L = M(P_L)$.

— Set $Q_R = M(P_R)$.

— Set $(Q_0, Q_1, Q_2, Q_3) = Q_L$.

— Set $(Q_4, Q_5, Q_6, Q_7) = Q_R$.

— Set $Y = Q_4 \| Q_5 \| Q_2 \| Q_3 \| Q_0 \| Q_1 \| Q_6 \| Q_7$.

— Output Y.

Figure 7 shows the block diagram of the function F.

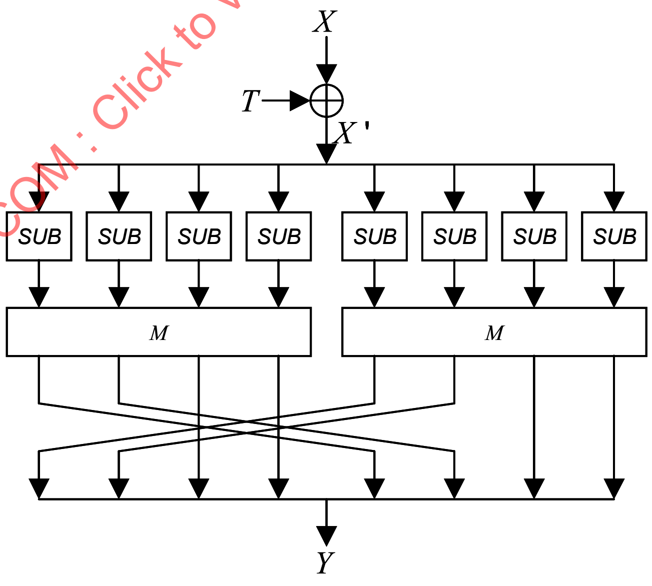

Figure 7 — function *F* of MUGI

8.1.8 Function S_R

The function S_R is the internal function of F. The function S_R can be described by using a substitution table. In this case, the function S_R is as follows:

Input: 8-bit string x.

Output: 8-bit string y.

— Set y = *SUB* $[x]$.

— Output y.

The *SUB* used in function S_R is a substitution as follows:

```
SUB [256] = {

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16}
```

8.1.9 Function *M*

The function *M* is the internal function of the *F* function. The function *M* is as follows:

Input: 32-bit string X.

Output: 32-bit string Y.

— Set $(x_0, x_1, x_2, x_3) = X$, where x_i is an 8-bit string and an element of $GF(2^8)$.

— Set

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \texttt{0x02} & \texttt{0x03} & \texttt{0x01} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x03} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x01} & \texttt{0x02} & \texttt{0x03} \\ \texttt{0x03} & \texttt{0x01} & \texttt{0x01} & \texttt{0x02} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

where 0x01, 0x02, and 0x03 are the hexadecimal expressions of the elements of $GF(2^8)$.

— Set $Y = y_0 \,||\, y_1 \,||\, y_2 \,||\, y_3$.

— Output Y.

8.2 SNOW 2.0 keystream generator

8.2.1 Introduction to SNOW 2.0

SNOW 2.0, in the sequel simply denoted SNOW, is a keystream generator which uses as input a 128 or 256-bit secret key K, and a 128-bit initialization vector IV. These are used to initiate a state variable S_i $(i \geq 0)$

consisting of eighteen $n = 32$ bit blocks. Bit/byte order is big-endian, i.e., if the key and initialization vector are given as a sequence of bits/bytes, the first/leftmost bit/byte is the most significant of the corresponding data. For every iteration of the *Strm* function, a 32-bit keystrem Z_i is produced as output.

SNOW's state variable S_i consists of two components. First, 16 32-bit variables:

$$a^{(i)} = (a_{15}^{(i)}, a_{14}^{(i)},\ldots, a_0^{(i)}),$$

implements a linear feedback shift register (LFSR). Secondly, 2 32-bit variables:

$$b^{(i)} = (b_2^{(i)}, b_1^{(i)}),$$

maintains the state of a finite state machine (FSM). SNOW is best understood with reference to Figure 8, which shows a snapshot, at time i, omitting the time dependence variable (i) from the notation.

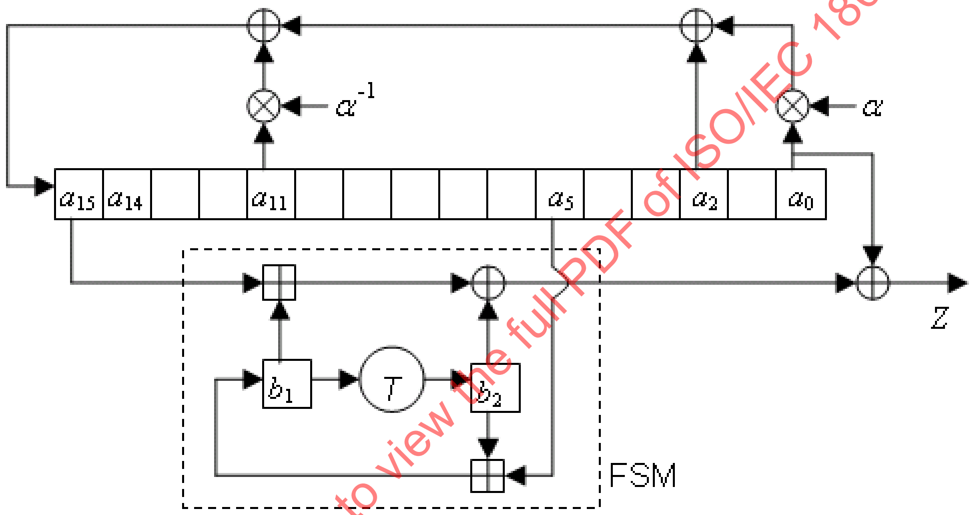

Figure 8 — Schematic drawing of SNOW

SNOW operation is defined by:

The *Init* function, defined in detail in 8.2.2, takes as input the 128 or 256-bit key K and the 128-bit IV, and produces the initial value of the state variable $S_0 = (a^{(0)}, b^{(0)})$.

The *Next* function, defined in detail in 8.2.3, takes as input the 18 32-bit state variable $S_i = (a^{(i)}, b^{(i)})$ and produces as output the next value of the state variable $S_{i+1} = (a^{(i+1)}, b^{(i+1)})$. The *Next* function runs in two modes, depending on whether the iteration performed is part of the initialization, or, of the normal mode of generating output, see below.

The *Strm* function, defined in detail in 8.2.4, takes as input the 18 32-bit state variable $S_i = (a^{(i)}, b^{(i)})$ and produces as output the 32-bit keystream Z_i.

NOTE 1 For SNOW, the maximum recommended amount of keystream produced from a given (K,IV) is 2^{50} 32 bits. This bound has been selected to provide good security margin against cryptanalysis, and implies no practical limitation in applicability of the algorithm.

NOTE 2 The paper [10] is referred for theoretical background on the design rationale for SNOW.

8.2.2 Initialization function *Init*

The Initialization function *Init* is as follows.

Input: 128- or 256-bit key K, 128-bit initialization vector IV.

Output: Initial value of state variable $S_0 = (a^{(0)}, b^{(0)})$.

a) Initialize the registers by the key information

— For a 128-bit key, set $(K_3, K_2, K_1, K_0) = K$, $a_{15\text{-}j}^{(-34)} = a_{15\text{-}j\text{-}8}^{(-34)} = K_{3\text{-}j}$, and $a_{15\text{-}j\text{-}4}^{(-34)} = a_{15\text{-}j\text{-}12}^{(-34)} = \neg K_{3\text{-}j}$ for $j = 0, 1, 2, 3$.

— For a 256-bit key, set $(K_7, K_6, \ldots, K_0) = K$, $a_{15\text{-}j}^{(-34)} = K_{7\text{-}j}$, and $a_{15\text{-}j\text{-}8}^{(-34)} = \neg(K_{7\text{-}j})$ for $j = 0, 1, \ldots, 7$.

b) Set $S_{-33} = (a^{(-33)}, b^{(-33)})$ by:

— Set $(IV_3, IV_2, IV_1, IV_0) = IV$.

— Set $a_i^{(-33)} = a_i^{(-34)}$ for $i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 13, 14$.

— Set $a_{15}^{(-33)} = a_{15}^{(-34)} \oplus IV_0$; $a_{12}^{(-33)} = a12^{(-34)} \oplus IV_1$; $a_{10}^{(-33)} = a_{10}^{(-34)} \oplus IV_2$; $a_9^{(-33)} = a_9^{(-34)} \oplus IV_3$.

— Set $b_1^{(-33)} = b_2^{(-33)} = 0^{(32)}$.

c) Set $S_{-1} = Next^{32}(S_{-33}, \text{INIT})$, where $Next^{32}$ denotes 32 iterations of the *Next* function.

d) $S_0 = Next(S_{-1})$.

e) Output S_0.

8.2.3 Next-state function *Next*

SNOW has two modes for *Next* function.

Input: State variable $S_i = (a^{(i)}, b^{(i)})$, mode = {INIT, null}.

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

a) Set $b_2^{(i+1)} = T(b_1^{(i)})$.

b) Set $b_1^{(i+1)} = b_2^{(i)} +_{32} a_5^{(i)}$.

c) For $j = 0, 1, \ldots, 14$ set $a_j^{(i+1)} = a_{j+1}^{(i)}$

d) If INIT mode, set $a_{15}^{(i+1)} = (a_0^{(i)} \otimes \alpha) \oplus a_2^{(i)} \oplus (a_{11}^{(i)} \oplus \alpha^{-1}) \oplus FSM(a_{15}^{(i)}, b_1^{(i)}, b_2^{(i)})$. Otherwise, set $a_{15}^{(i+1)} = (a_0^{(l)} \otimes \alpha) \oplus a_2^{(i)} \oplus (a_{11}^{(i)} \otimes \alpha^{-1})$.

e) $S_{i+1} = (a^{(i+1)}, b^{(i+1)})$.

f) Output S_{i+1} .

The description of the *T* function and the finite field arithmetic involving the fixed element α refers to 8.2.5 and 8.2.6, respectively.

NOTE Figure 9 shows the block diagram of the INIT mode of *Next* function.

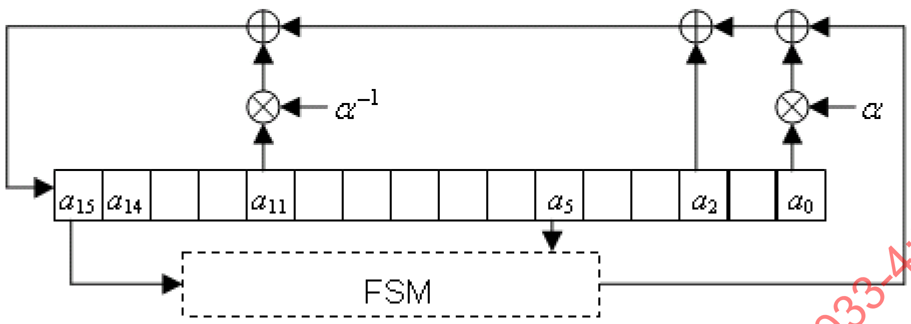

Figure 9 — INIT mode of *Next* function

The definition of the *FSM* function refers to 8.2.8.

8.2.4 Keystream function *Strm*

The keystream function *Strm* is as follows:

Input: State variable S_i .

Output: 32-bit keystream Z_i.

a) Set $Z_i = FSM\ (a_{15}{}^{(i)}, b_1{}^{(i)}, b_2{}^{(i)}\ ) \oplus a_0{}^{(i)}$.

b) Output Z_i.

8.2.5 Function *T*

The *T* function is a substitution, specifically a permutation of $GF(2^{32})$, based on components from the Advanced Encryption Standard (AES), ISO/IEC 18033-3. To this end, the finite field $GF(2^8)$ is used, which is viewed as $GF(2)$[x] modulo the irreducible polynomial

$$f(x) = x^8 + x^4 + x^3 + x + 1.$$

and the polynomial ring $GF(2^8)$ [y] modulo ($y^4 + 1$).

Input: 32-bit string w.

Output: 32-bit string $q = T(w)$.

a) Set $(w_3, w_2, w_1, w_0) = w$, where each w_j is 8 bit.

b) For $j = 0, 1, 2, 3$ set $t_j = SUB\ [w_j]$.

c) Let $t(y)$ be the polynomial $t(y) = t_3\, y^3 + t_2\, y^2 + t_1\, y + t_0$ in $GF(2^8)$ [y], where t_j is interpreted as an element of $GF(2^8)$ in the natural way: $t_j = t_{j,7}\, x^7 + \ldots + t_{j,1}\, x + t_{j,0}$, $t_{j,k}$ in $GF(2)$.

d) set $q(y) = c(y) \bullet t(y)$ modulo $(y^4 + 1)$, where $c(y) = (x+1)y^3 + y^2 + y + x$ in $GF(2^8)$ [y].

e) associate the 32-bit string $q = (q_3, q_2, q_1, q_0)$ with the result of the above, $q(y) = q_3 y^3 + q_2 y^2 + q_1 y + q_0$.

f) Output q.

Note that in step c), the two polynomials are multiplied where coefficient-by-coefficient operations are carried out in $GF(2^8)$ as defined by $f(x)$ above. The result is then reduced modulo $y^4 + 1$.

NOTE 1 The AES S-box can be found in 8.1.8.

NOTE 2 The paper [10] is referred for details of this (and other) optimisation(s).

8.2.6 Multiplication of α in finite field arithmetic

Input: 32-bit string w, representing an element of $GF(2^{32})$.

Output: 32-bit string w', representing $\alpha \otimes w$ in $GF(2^{32})$.

a) Set $w' = (w <<_{32} 8) \oplus \alpha_{\mathrm{MUL}}[w >>_{32} 24]$.

b) Output w'.

The function α_{MUL} is defined in the following:

```
α_MUL [256] = {
0x00000000,0xE19FCF13,0x6B973726,0x8A08F835,0xD6876E4C,0x3718A15F,0xBD10596A,0x5C8F9679,
0x05A7DC98,0xE438138B,0x6E30EBBE,0x8FAF24AD,0xD320B2D4,0x32BF7DC7,0xB8B785F2,0x59284AE1,
0x0AE71199,0xEB78DE8A,0x617026BF,0x80EFE9AC,0xDC607FD5,0x3DFFB0C6,0xB7F748F3,0x566887E0,
0x0F40CD01,0xEEDF0212,0x64D7FA27,0x85483534,0xD9C7A34D,0x38586C5E,0xB250946B,0x53CF5B78,
0x1467229B,0xF5F8ED88,0x7FF015BD,0x9E6FDAAE,0xC2E04CD7,0x237F83C4,0xA9777BF1,0x48E8B4E2,
0x11C0FE03,0xF05F3110,0x7A57C925,0x9BC80636,0xC747904F,0x26D85F5C,0xACD0A769,0x4D4F687A,
0x1E803302,0xFF1FFC11,0x75170424,0x9488CB37,0xC8075D4E,0x2998925D,0xA3906A68,0x420FA57B,
0x1B27EF9A,0xFAB82089,0x70B0D8BC,0x912F17AF,0xCDA081D6,0x2C3F4EC5,0xA637B6F0,0x47A879E3,
0x28CE449F,0xC9518B8C,0x435973B9,0xA2C6BCAA,0xFE492AD3,0x1FD6E5C0,0x95DE1DF5,0x7441D2E6,
0x2D699807,0xCCF65714,0x46FEAF21,0xA7616032,0xFBEEF64B,0x1A713958,0x9079C16D,0x71E60E7E,
0x22295506,0xC3B69A15,0x49BE6220,0xA821AD33,0xF4AE3B4A,0x1531F459,0x9F390C6C,0x7EA6C37F,
0x278E899E,0xC611468D,0x4C19BEB8,0xAD8671AB,0xF109E7D2,0x109628C1,0x9A9ED0F4,0x7B011FE7,
0x3CA96604,0xDD36A917,0x573E5122,0xB6A19E31,0xEA2E0848,0x0BB1C75B,0x81B93F6E,0x6026F07D,
0x390EBA9C,0xD891758F,0x52998DBA,0xB30642A9,0xEF89D4D0,0x0E161BC3,0x841EE3F6,0x65812CE5,
0x364E779D,0xD7D1B88E,0x5DD940BB,0xBC468FA8,0xE0C919D1,0x0156D6C2,0x8B5E2EF7,0x6AC1E1E4,
0x33E9AB05,0xD2766416,0x587E9C23,0xB9E15330,0xE56EC549,0x04F10A5A,0x8EF9F26F,0x6F663D7C,
0x50358897,0xB1AA4784,0x3BA2BFB1,0xDA3D70A2,0x86B2E6DB,0x672D29C8,0xED25D1FD,0x0CBA1EEE,
0x5592540F,0xB40D9B1C,0x3E056329,0xDF9AAC3A,0x83153A43,0x628AF550,0xE8820D65,0x091DC276,
0x5AD2990E,0xBB4D561D,0x3145AE28,0xD0DA613B,0x8C55F742,0x6DCA3851,0xE7C2C064,0x065D0F77,
0x5F754596,0xBEEA8A85,0x34E272B0,0xD57DBDA3,0x89F22BDA,0x686DE4C9,0xE2651CFC,0x03FAD3EF,
0x4452AA0C,0xA5CD651F,0x2FC59D2A,0xCE5A5239,0x92D5C440,0x734A0B53,0xF942F366,0x18DD3C75,
0x41F57694,0xA06AB987,0x2A6241B2,0xCBFD8EA1,0x977218D8,0x76EDD7CB,0xFCE52FFE,0x1D7AE0ED,
0x4EB5BB95,0xAF2A7486,0x25228CB3,0xC4BD43A0,0x9832D5D9,0x79AD1ACA,0xF3A5E2FF,0x123A2DEC,
0x4B12670D,0xAA8DA81E,0x2085502B,0xC11A9F38,0x9D950941,0x7C0AC652,0xF6023E67,0x179DF174,
0x78FBCC08,0x9964031B,0x136CFB2E,0xF2F3343D,0xAE7CA244,0x4FE36D57,0xC5EB9562,0x24745A71,
0x7D5C1090,0x9CC3DF83,0x16CB27B6,0xF754E8A5,0xABDB7EDC,0x4A44B1CF,0xC04C49FA,0x21D386E9,
0x721CDD91,0x93831282,0x198BEAB7,0xF81425A4,0xA49BB3DD,0x45047CCE,0xCF0C84FB,0x2E934BE8,
0x77BB0109,0x9624CE1A,0x1C2C362F,0xFDB3F93C,0xA13C6F45,0x40A3A056,0xCAAB5863,0x2B349770,
0x6C9CEE93,0x8D032180,0x070BD9B5,0xE69416A6,0xBA1B80DF,0x5B844FCC,0xD18CB7F9,0x301378EA,
0x693B320B,0x88A4FD18,0x02AC052D,0xE333CA3E,0xBFBC5C47,0x5E239354,0xD42B6B61,0x35B4A472,
0x667BFF0A,0x87E43019,0x0DECC82C,0xEC73073F,0xB0FC9146,0x51635E55,0xDB6BA660,0x3AF46973,
0x63DC2392,0x8243EC81,0x084B14B4,0xE9D4DBA7,0xB55B4DDE,0x54C482CD,0xDECC7AF8,0x3F53B5EB};
```

8.2.7 Multiplication of α^{-1} in finite field arithmetic

Input: 32-bit string y, representing an element of $GF(2^{32})$.

Output: 32-bit string y, representing $\alpha^{-1} \otimes y$ in $GF(2^{32})$.

a) Set $y' = (y >>_{32} 8) \otimes \alpha_{\mathrm{inv\_MUL}}[y \bmod 256]$.

b) Output y'.

The function $\alpha_{\text{inv\_MUL}}$ is defined in the following:

$\alpha_{\text{inv\_MUL}}$ [256]= {
```
0x00000000,0x180F40CD,0x301E8033,0x2811C0FE,0x603CA966,0x7833E9AB,0x50222955,0x482D6998,
0xC078FBCC,0xD877BB01,0xF0667BFF,0xE8693B32,0xA04452AA,0xB84B1267,0x905AD299,0x88559254,
0x29F05F31,0x31FF1FFC,0x19EEDF02,0x01E19FCF,0x49CCF657,0x51C3B69A,0x79D27664,0x61DD36A9,
0xE988A4FD,0xF187E430,0xD99624CE,0xC1996403,0x89B40D9B,0x91BB4D56,0xB9AA8DA8,0xA1A5CD65,
0x5249BE62,0x4A46FEAF,0x62573E51,0x7A587E9C,0x32751704,0x2A7A57C9,0x026B9737,0x1A64D7FA,
0x923145AE,0x8A3E0563,0xA22FC59D,0xBA208550,0xF20DECC8,0xEA02AC05,0xC2136CFB,0xDA1C2C36,
0x7BB9E153,0x63B6A19E,0x4BA76160,0x53A821AD,0x1B854835,0x038A08F8,0x2B9BC806,0x339488CB,
0xBBC11A9F,0xA3CE5A52,0x8BDF9AAC,0x93D0DA61,0xDBFDB3F9,0xC3F2F334,0xEBE333CA,0xF3EC7307,
0xA492D5C4,0xBC9D9509,0x948C55F7,0x8C83153A,0xC4AE7CA2,0xDCA13C6F,0xF4B0FC91,0xECBFBC5C,
0x64EA2E08,0x7CE56EC5,0x54F4AE3B,0x4CFBEEF6,0x04D6876E,0x1CD9C7A3,0x34C8075D,0x2CC74790,
0x8D628AF5,0x956DCA38,0xBD7C0AC6,0xA5734A0B,0xED5E2393,0xF551635E,0xDD40A3A0,0xC54FE36D,
0x4D1A7139,0x551531F4,0x7D04F10A,0x650BB1C7,0x2D26D85F,0x35299892,0x1D38586C,0x053718A1,
0xF6DB6BA6,0xEED42B6B,0xC6C5EB95,0xDECAAB58,0x96E7C2C0,0x8EE8820D,0xA6F942F3,0xBEF6023E,
0x36A3906A,0x2EACD0A7,0x06BD1059,0x1EB25094,0x569F390C,0x4E9079C1,0x6681B93F,0x7E8EF9F2,
0xDF2B3497,0xC724745A,0xEF35B4A4,0xF73AF469,0xBF179DF1,0xA718DD3C,0x8F091DC2,0x97065D0F,
0x1F53CF5B,0x075C8F96,0x2F4D4F68,0x37420FA5,0x7F6F663D,0x676026F0,0x4F71E60E,0x577EA6C3,
0xE18D0321,0xF98243EC,0xD1938312,0xC99CC3DF,0x81B1AA47,0x99BEEA8A,0xB1AF2A74,0xA9A06AB9,
0x21F5F8ED,0x39FAB820,0x11EB78DE,0x09E43813,0x41C9518B,0x59C61146,0x71D7D1B8,0x69D89175,
0xC87D5C10,0xD0721CDD,0xF863DC23,0xE06C9CEE,0xA841F576,0xB04EB5BB,0x985F7545,0x80503588,
0x0805A7DC,0x100AE711,0x381B27EF,0x20146722,0x68390EBA,0x70364E77,0x58278E89,0x4028CE44,
0xB3C4BD43,0xABCBFD8E,0x83DA3D70,0x9BD57DBD,0xD3F81425,0xCBF754E8,0xE3E69416,0xFBE9D4DB,
0x73BC468F,0x6BB30642,0x43A2C6BC,0x5BAD8671,0x1380EFE9,0x0B8FAF24,0x239E6FDA,0x3B912F17,
0x9A34E272,0x823BA2BF,0xAA2A6241,0xB225228C,0xFA084B14,0xE2070BD9,0xCA16CB27,0xD2198BEA,
0x5A4C19BE,0x42435973,0x6A52998D,0x725DD940,0x3A70B0D8,0x227FF015,0x0A6E30EB,0x12617026,
0x451FD6E5,0x5D109628,0x750156D6,0x6D0E161B,0x25237F83,0x3D2C3F4E,0x153DFFB0,0x0D32BF7D,
0x85672D29,0x9D686DE4,0xB579AD1A,0xAD76EDD7,0xE55B844F,0xFD54C482,0xD545047C,0xCD4A44B1,
0x6CEF89D4,0x74E0C919,0x5CF109E7,0x44FE492A,0x0CD320B2,0x14DC607F,0x3CCDA081,0x24C2E04C,
0xAC977218,0xB49832D5,0x9C89F22B,0x8486B2E6,0xCCABDB7E,0xD4A49BB3,0xFCB55B4D,0xE4BA1B80,
0x17566887,0x0F59284A,0x2748E8B4,0x3F47A879,0x776AC1E1,0x6F65812C,0x477441D2,0x5F7B011F,
0xD72E934B,0xCF21D386,0xE7301378,0xFF3F53B5,0xB7123A2D,0xAF1D7AE0,0x870CBA1E,0x9F03FAD3,
0x3EA637B6,0x26A9777B,0x0EB8B785,0x16B7F748,0x5E9A9ED0,0x4695DE1D,0x6E841EE3,0x768B5E2E,
0xFEDECC7A,0xE6D18CB7,0xCEC04C49,0xD6CF0C84,0x9EE2651C,0x86ED25D1,0xAEFCE52F,0xB6F3A5E2};
```

8.2.8 Function *FSM* (x, y, z)

Input: Three 32-bit strings, x, y, and z.

Output: 32-bit string q.

a) Set $q = (x +_{32} y) \oplus z$.

b) Output q.

8.3 Rabbit keystream generator

8.3.1 Introduction to Rabbit

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 laid out in Clause 4 of this part of ISO/IEC 18033. 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 The paper [8] is referred for the original proposal of the cipher and the paper [9] is referred for an overview of its cryptographic security.

8.3.2 Additional variables and notation

For the Rabbit keystream generator, the following notation is added:

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.

8.3.3 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 $S_0 = (b^{(0)}, X_0^{(0)}, \ldots, X_7^{(0)}, C_0^{(0)}, \ldots, C_7^{(0)})$.

Local variables: counters i, j

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

b) Set S_{-9} as follows:

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

2) For $j = 0, 1, \ldots, 7$:

— If j is even, set $X_j^{(-9)} = K_{(j+1 \bmod 8)} \| K_j$ and $C_j^{(-9)} = K_{(j+4 \bmod 8)} \| K_{(j+5 \bmod 8)}$.

— Else, set $X_j^{(-9)} = K_{(j+5 \bmod 8)} \| K_{(j+4 \bmod 8)}$ and $C_j^{(-9)} = K_j \| K_{(j+1 \bmod 8)}$.

c) Iterate the next-state function *Next* four times: set $S_i = Next\,(S_{i-1})$ for $i = -8, -7, -6, -5$.

d) Set S_{-4} as follows:

1) Modify the counters as follows:

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

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

e) Iterate the next-state function Next four times: set $S_i = Next\,(S_{i-1})$ for $i = -3, -2, -1, 0$.

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

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

8.3.4 Next-state function *Next*

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

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

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

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

a) Set constants $A_0, \ldots, A_7$ 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`

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

c) For $j = 0, 1, \ldots, 7$:

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

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

— Let $C_j^{(i+1)} = temp^{[31..0]}$.

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

e) For $j = 0, 1, \ldots, 7$, let $G_j = g(X_j^{(i)}, C_j^{(i+1)})$, where the function g is given in 8.3.6.

f) Modify internal state as follows:

$$X_0^{(i+1)} = G_0 +_{32} (G_7 <<<_{32} 16) +_{32} (G_6 <<<_{32} 16)$$

$$X_1^{(i+1)} = G_1 +_{32} (G_0 <<<_{32} 8) +_{32} G_7$$

$$X_2^{(i+1)} = G_2 +_{32} (G_1 <<<_{32} 16) +_{32} (G_0 <<<_{32} 16)$$

$$X_3^{(i+1)} = G_3 +_{32} (G_2 <<<_{32} 8) +_{32} G_1$$

$$X_4^{(i+1)} = G_4 +_{32} (G_3 <<<_{32} 16) +_{32} (G_2 <<<_{32} 16)$$

$$X_5^{(i+1)} = G_5 +_{32} (G_4 <<<_{32} 8) +_{32} G_3$$

$$X_6^{(i+1)} = G_6 +_{32} (G_5 <<<_{32} 16) +_{32} (G_4 <<<_{32} 16)$$

$$X_7^{(i+1)} = G_7 +_{32} (G_6 <<<_{32} 8) +_{32} G_5$$

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

8.3.5 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.

a) Set Z_i as follows:

$$Z_i^{[15..0]} = X_0^{(i)[15..0]} \oplus X_5^{(i)[31..16]}$$

$$Z_i^{[31..16]} = X_0^{(i)[31..16]} \oplus X_3^{(i)[15..0]}$$

$$Z_i^{[47..32]} = X_2^{(i)[15..0]} \oplus X_7^{(i)[31..16]}$$

$$Z_i^{[63..48]} = X_2^{(i)[31..16]} \oplus X_5^{(i)[15..0]}$$

$$Z_i^{[79..64]} = X_4^{(i)[15..0]} \oplus X_1^{(i)[31..16]}$$

$$Z_i^{[95..80]} = X_4^{(i)[31..16]} \oplus X_7^{(i)[15..0]}$$

$$Z_i^{[111..96]} = X_6^{(i)[15..0]} \oplus X_3^{(i)[31..16]}$$

$$Z_i^{[127..112]} = X_6^{(i)[31..16]} \oplus X_1^{(i)[15..0]}$$

b) Output Z_i.

8.3.6 Function *g*

The function *g* is specified as follows:

Input: Two 32-bit parameters u and v.

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

Local variables: 64-bit positive integer *temp*

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

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

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

8.4 Decim^{v2} keystream generator

8.4.1 Introduction to Decim^{v2}

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 *LF*. In keystream generation mode, the output of *LF* 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.

DECIM^{v2} is described in Figure 10, which shows a snapshot, at time i, omitting the time dependence variable (i) from the notation.

NOTE 1 The paper [6] is referred for theoretical background on the design rationale of DECIM^{v2}.

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

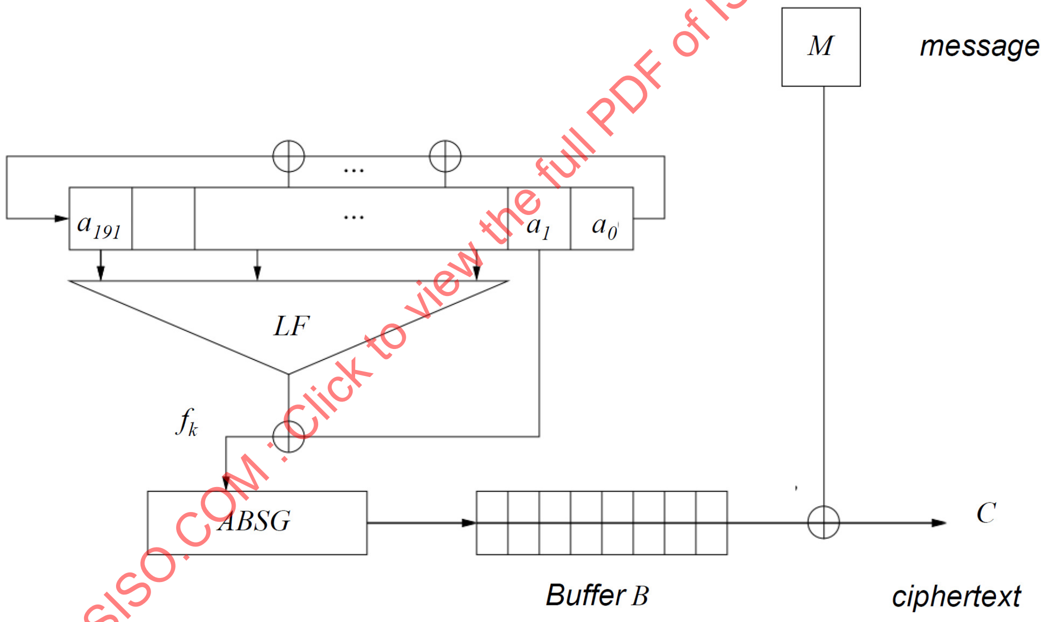

Figure 10 — Schematic drawing of DECIM^{v2}

The *Init* function, defined in detail in 8.4.3, 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 8.4.5, takes as an input the value of the state variable $S_i = (a^{(i)}, T^{(i)}, b^{(i)}, I^{(i)})$ and produces as output 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 runs in three 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 8.4.6, 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 [6].

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 8.4.3.

NOTE 4 DECIM^{v2} is immune to the attacks as described in [18].

8.4.2 Additional variables and notation

For the Decim^{v2} keystream generator, the following notation is added:

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}
I, I'	Inner state variables for Decim^{v2}
LF	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.

8.4.3 Initialization function *Init*

The Initialization function *Init* is defined as follows.

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.

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

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

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

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

b) Initialize the buffer and the compression function:

— Set $T^{(-256)} = 000$.

— Set $b_j^{(-256)} = 0$ for $j = 0,1,\ldots,31$.

— Set $I^{(-256)} = 0$.

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

d) Set $i = -64$.

e) While $I^{(i)} < 32$ and $i<0$: set S_{i+1} = *InitNext*(S_i , BUFF) and $i = i + 1$. The test $I^{(i)} < 32$ can be removed, if a fixed, constant number of steps in the *Init* function are needed for implementation.

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 $1\text{-}2^{-97}$. On average, the buffer is full after 24 iterations.

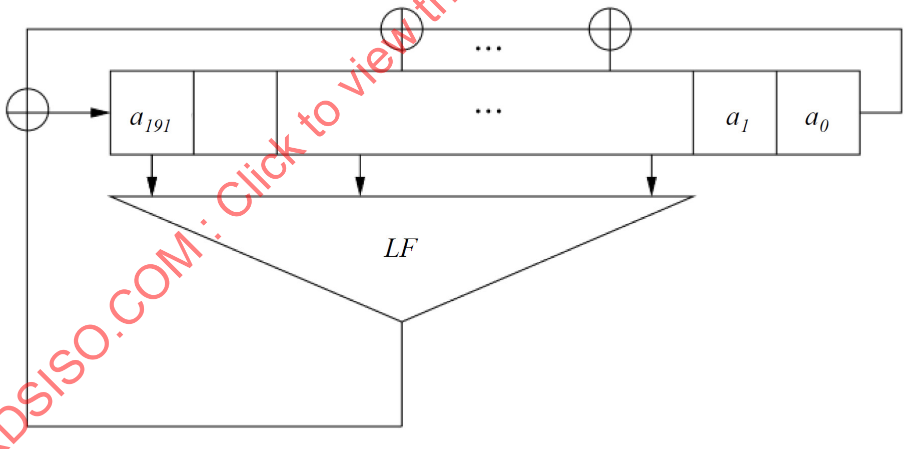

Figure 11 — LFSR mode of Initialization Next-state function *InitNext*

8.4.4 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 $S_i = (a^{(i)}, T^{(i)}, b^{(i)}, I^{(i)})$, mode $\in$ {LFSR, BUFF}.

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

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

LFSR mode (execute if mode = LFSR):

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

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

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

— Set f_k = *LF* ($\alpha^{(k)}$) and $r = L(\alpha^{(k)}) \oplus f_k$.

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

— Set $\alpha_{191}^{(k+1)} = r$.

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

Figure 11 shows the block diagram of the LFSR mode of *InitNext* function.

BUFF mode (execute if mode = BUFF):

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

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

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

— Set $f_k = \alpha_1^{(k)} \oplus$ *LF*($\alpha^{(k)}$) and $r = L(\alpha^{(k)})$.

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

— Set $\alpha_{191}^{(k+1)} = r$.

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

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

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

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

— Set $c = f_k \oplus \tau_2^{(k)}$.

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

— 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 $I^{(i+1)} = \iota^{(4)}$.

8.4.5 Next-state function *Next*

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

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

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

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

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

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

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

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

   — Set $\alpha_{191}^{(k+1)} = r$.

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

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

c) For $j = 0, 1, \ldots, \tau^{(0)}-1$, 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:

   — Set $c = f_k \oplus \tau_2^{(k)}$.

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

   — 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 $I^{(i+1)} = \iota^{(4)}$.

NOTE 1 The condition $\iota^{(0)} = 0$ in step 1) of step d) should never be satisfied; if it is, this means that the buffer has become empty during the keystream generation. This happens with probability less than 2^{-80} at every state update, see [8] for details. Also, this probability is higher if the buffer is not full after the *Init* function, but, as mentioned in 8.4.3 (NOTE), this also happens with negligible probability.

NOTE 2 The *InitNext* function and the *Next* function share many computational steps. Indeed, the LFSR mode of the *InitNext* function mainly consists of the LFSR update of the BUFF mode and of the *Next* function, the only difference being that the Boolean function output is added to the feedback bit. The BUFF mode of the *InitNext* function and the *Next* function differ only in that the buffer B is shifted only in the latter.

8.4.6 Keystream function *Strm*

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

Output: Keystream bit Z_i.

a) Set $Z_i = b_0^{(i)}$.

b) Output Z_i.

8.4.7 Linear feedback function *L*

Input: 192-bit string $w = (w_0, w_1, \ldots, w_{191})$.

Output: Bit $q = L(w)$.

Set $q = w_0 \oplus w_3 \oplus w_4 \oplus w_{23} \oplus w_{36} \oplus w_{37} \oplus w_{60} \oplus w_{61} \oplus w_{98} \oplus w_{115} \oplus w_{146} \oplus w_{175} \oplus w_{176} \oplus w_{187}$.

8.4.8 Filtering function *LF*

Input: 192-bit string $w = (w_0, w_1, \ldots, w_{191})$.

Output: Bit $q = LF(w)$. Set $q = Y((w_{13}, w_{28}, w_{45}, w_{54}, w_{65}, w_{104}, w_{111}, w_{144}, w_{162}, w_{172}, w_{178}, w_{186}, w_{191}))$.

8.4.9 Boolean function *Y*

Input: 13-bit string $w = (w_0, w_1, \ldots, w_{12})$.

Output: Bit $q = Y(w)$. Set $q = (\oplus_{0 \le j \le 12} w_j) \oplus (\oplus_{0 \le j \le 12} w_j w_k)$.

NOTE Equivalently, q is given by $q = 0$ if $X = 0$ or $X = 3$, and $q = 1$ otherwise, with $X = w_0 + w_1 + \ldots + w_{12} \bmod 4$.

8.4.10 Compression function *ABSG*

Input: 3-bit state T, input bit c.

Output: 3-bit state $T' = ABSG(T, c)$.

a) If $T_0 = 1$, set $T'_1 = T_1$, otherwise set $T'_1 = c$.

b) Set $T'_2 = T_0$ AND $(T_1 \oplus c)$.

c) Set $T'_0 = (T_0 \oplus 1)$ OR T'_2.

8.4.11 Buffering function *B*

Input: 32-bit string $b = (b_0, b_1, \ldots, b_{31})$, index I, Boolean *output*, input bit c.

Output: 32-bit string $b' = (b'_0, b'_1, \ldots, b'_{31})$, index I'.

a) Set $I' = I$, $b' = b$.

b) If *output* = TRUE and $I' < 32$, do the following:

— Set $b'I' = c$.

— Set $I' = I' + 1$.

c) Output $B(b, I, output, c) = (b', I')$.

8.5 KCipher-2 (K2) keystream generator

8.5.1 Introduction to KCipher-2 (K2)

KCipher-2 (K2) is a keystream generator which uses as input a 128-bit secret key K and a 128-bit initial vector IV. These are used to initialize state variables $S_i\,(i \geq 0)$ consisting of twenty 32-bit blocks, where S_i represents the internal state of K2 at clock i. Bit/byte order is big-endian, i.e., if the key and initialization vector are given as a sequence of bits/bytes, the first/leftmost bit/byte is the most significant of the corresponding data. For every iteration of the *Strm* function, a 64-bit keystream Z_i is produced as output.

K2's state variable S_i consists of three components. The first component $A^{(i)}$ consists of a sequence of five 32-bit variables:

$$A^{(i)}=(A_4^{(i)}, A_3^{(i)}, A_2^{(i)}, A_1^{(i)}, A_0^{(i)}) \quad (A_m^{(i)} \text{ in } GF(2^{32}), m\geq 0)$$

which form the state for a feedback shift register (*FSR*) *A*. The second component $B^{(i)}$ consists of a sequence of eleven 32-bit variables:

$$B^{(i)}=(B_{10}^{(i)}, B_9^{(i)} ,..., B_0^{(i)}) \quad (B_m^{(i)} \text{ in } GF(2^{32}), m\geq 0)$$

which form the state for an *FSR B*. The third component consists of a set of four 32-bit variables:

$$R1^{(i)},L1^{(i)} ,R2^{(i)},L2^{(i)} \text{ in } GF(2^{32})$$

which maintain the state of a non-linear function. The operation of K2 is summarized in Figures 12 and 13, which show a snapshot of operation, at time i, omitting the time dependent variable (i) from the notation.

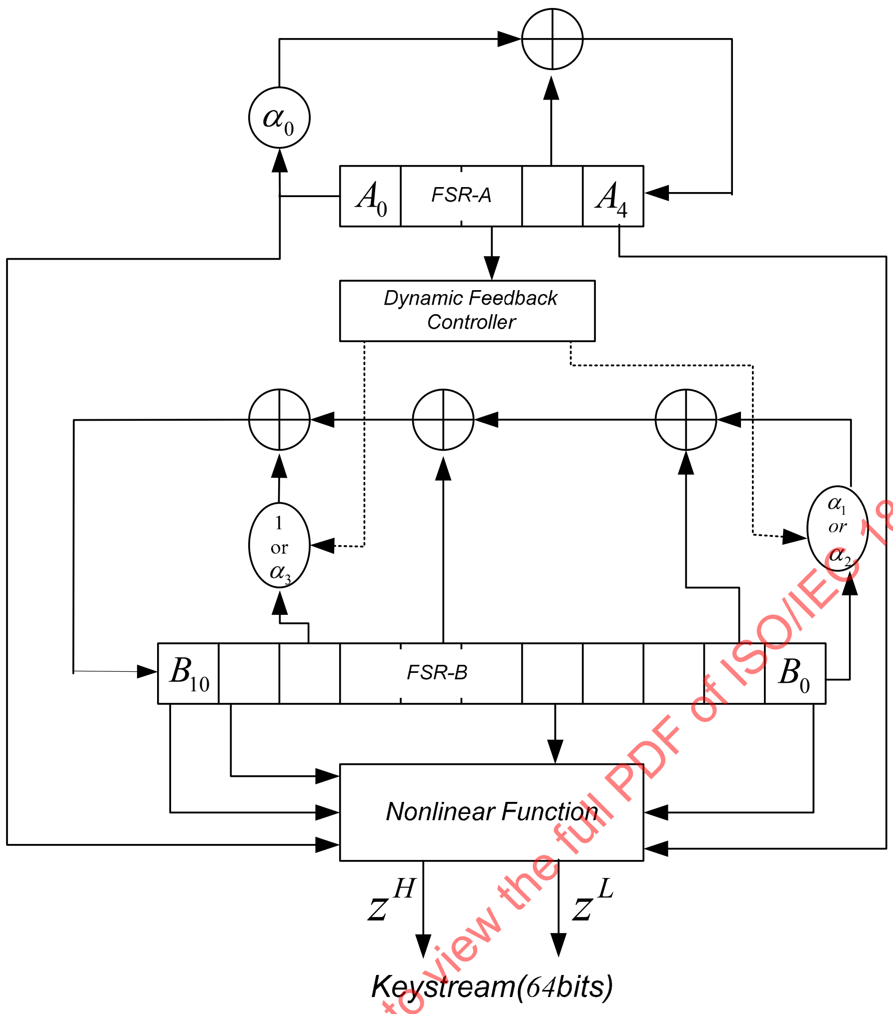

Figure 12 — Schematic drawing of K2

The operation of K2 is defined by the following three functions:

The *Init* function, defined in 8.5.2, takes as input the 128-bit key K and the 128-bit IV to produce the initial state S_0=($A^{(0)}$, $B^{(0)}$, $R1^{(0)}$, $L1^{(0)}$, $R2^{(0)}$, $L2^{(0)}$).

The *Next* function, defined in 8.5.3, takes as input the internal state, S_i=($A^{(i)}$, $B^{(i)}$, $R1^{(i)}$, $L1^{(i)}$, $R2^{(i)}$, $L2^{(i)}$) and produces as output the next value of the state variable S_{i+1}=($A^{(i+1)}$, $B^{(i+1)}$, $R1^{(i+1)}$, $L1^{(i+1)}$, $R2^{(i+1)}$, $L2^{(i+1)}$). The *Next* function runs in two modes, depending on whether the iteration performed is part of the initialization, or, of the normal mode of generating output.

The *Strm* function, defined in 8.5.4, takes as input the internal state, S_i=($A^{(i)}$, $B^{(i)}$, $R1^{(i)}$, $L1^{(i)}$, $R2^{(i)}$, $L2^{(i)}$) and produces as output the 64-bit keystream Z_i=(Z_i^H, Z_i^L).

NOTE 1 The recommended maximum number of keystream bits without either re-keying or re-initializing with new IV is 2^{64} bits.

NOTE 2 For the design rational for K2, refer to [12], [13], [14].

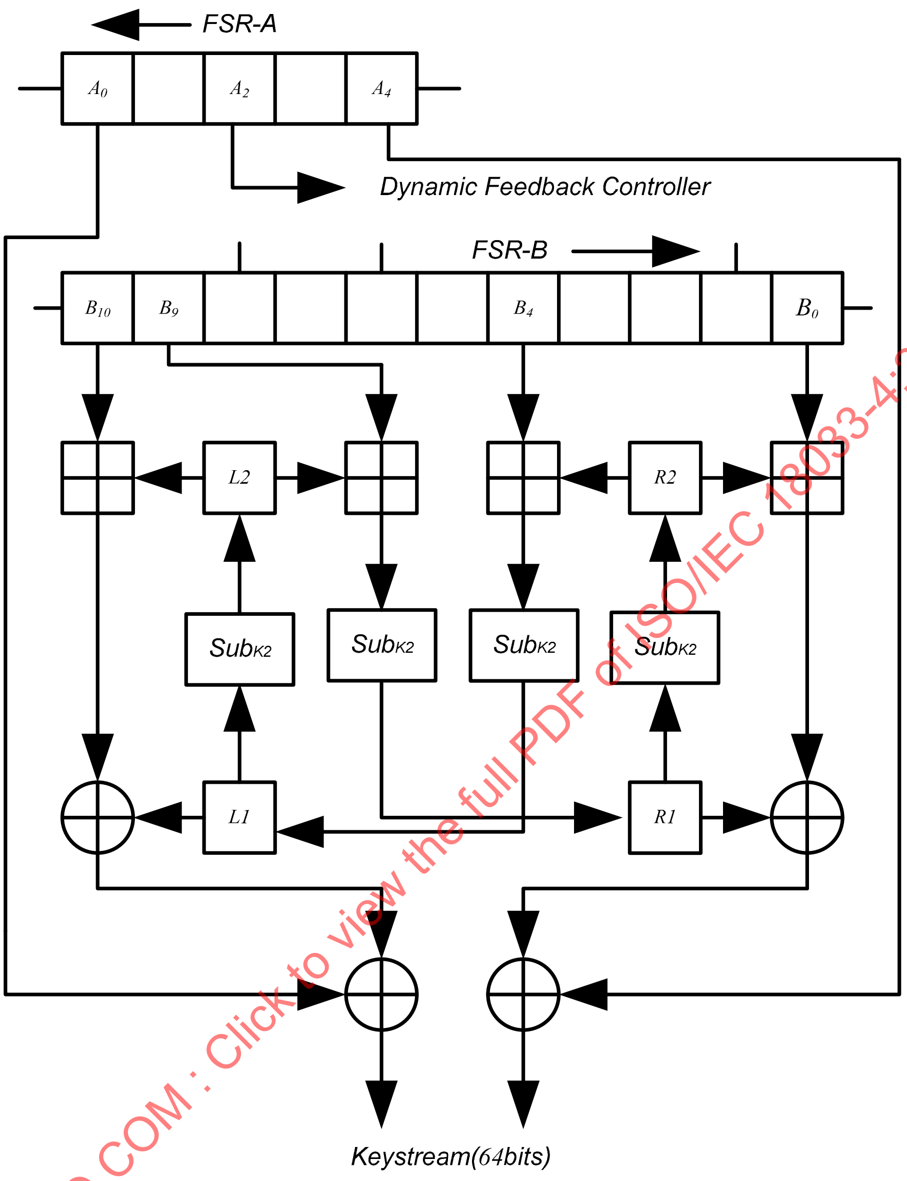

Figure 13 — Nonlinear function of K2

8.5.2 Initialization function *Init*

The initialization function *Init* works as follows.

Input: 128-bit key K and 128-bit initial vector IV.

Output; Initial value of state variable S_0=($A^{(0)}$,$B^{(0)}$,$R1^{(0)}$,$L1^{(0)}$,$R2^{(0)}$,$L2^{(0)}$).

Local variables: counter m.

a) Expand the 128-bit key K=(K_0, K_1, K_2, K_3) into the 384-bit internal key IK=(IK_0, IK_1,...,IK_{11}) as follows:

1) For m=0, 1, 2, 3, set IK_m= K_m.

2) For m=4,5,...,11,

   — If $m \neq 4$ or 8, set $IK_m = IK_{m\text{-}4} \oplus IK_{m\text{-}1}$.

   — If m=4 set *Rcon*[0]=(`0x01, 0x00, 0x00, 0x00`) and $IK_m = IK_{m\text{-}4} \oplus Sub_{K2}((IK_{m\text{-}1} \ll_{32} 8) \oplus (IK_{m\text{-}1} \gg_{32} 24)) \oplus Rcon[m/4\text{-}1]$. The Sub_{K2} function is referred to in 8.5.5.

   — If m=8 set *Rcon*[1]=(`0x02, 0x00, 0x00, 0x00`) and $IK_m = IK_{m\text{-}4} \oplus Sub_{K2}((IK_{m\text{-}1} \ll_{32} 8) \oplus (IK_{m\text{-}1} \gg_{32} 24)) \oplus Rcon[m/4\text{-}1]$. The Sub_{K2} function is referred to in 8.5.5.

b) Initialize the registers with the internal key IK and $IV = (IV_0, IV_1, IV_2, IV_3)$.

— For m=0, 1, 2, 3, 4, set $A_m^{(-24)} = IK_{4\text{-}m}$.

— Set the registers in *FSR-B* as follows.

$$B_0^{(-24)}=IK_{10},\ B_1^{(-24)}=IK_{11},\ B_2^{(-24)}=IV_0,\ B_3^{(-24)}=IV_1,\ B_4^{(-24)}= IK_8,$$
$$B_5^{(-24)}= IK_9,\ B_6^{(-24)}=IV_2,\ B_7^{(-24)}=IV_3,\ B_8^{(-24)} = IK_7,\ B_9^{(-24)} = IK_5,\ B_{10}^{(-24)} = IK_6.$$

— Set the registers in the nonlinear function as follows.

$R1^{(-24)}$=`0x00000000`, $L1^{(-24)}$ =`0x00000000`, $R2^{(-24)}$=`0x00000000`, $L2^{(-24)}$=`0x00000000`.

c) Set S_0= *Next*^{24}(S_{-24}, INIT), where *Next*^{24} denotes 24 iterations of the *Next* function.

d) Output S_0.

We refer to 8.5.5 for description of the Sub_{K2} function.

8.5.3 Next-state function *Next*

K2 has two modes for the *Next* function.

Input: State variable $S_i=(A^{(i)}, B^{(i)}, R1^{(i)}, L1^{(i)}, R2^{(i)}, L2^{(i)})$, mode = {INIT, null}.

Output: Next value of the state variable $S_{i+1}=(A^{(i+1)},B^{(i+1)},R1^{(i+1)},L1^{(i+1)},R2^{(i+1)},L2^{(i+1)})$.

Local variables: counter m.

a) Set the variables in the nonlinear function as follows.

$$R1^{(i+1)}=Sub_{K2}(L2^{(i)} +_{32} B_9^{(i)}),$$
$$L1^{(i+1)}=Sub_{K2}(R2^{(i)} +_{32} B_4^{(i)}),$$
$$R2^{(i+1)}=Sub_{K2}(R1^{(i)}),$$
$$L2^{(i+1)}=Sub_{K2}(L1^{(i)}),$$

b) For m=0, 1, 2, 3, set $A_m^{(i+1)} = A_{m+1}^{(i)}$.

c) For m=0, 1,..., 9, set $B_m^{(i+1)}=B_{m+1}^{(i)}$.

d) For INIT mode, set $A_4^{(i+1)}=(\alpha_0 \otimes A_0^{(i)}) \oplus A_3^{(i)} \oplus NLF(B_0^{(i)}, R2^{(i)}, R1^{(i)}, A_4^{(i)})$,
For null mode, set $A_4^{(i+1)}=(\alpha_0 \otimes A_0^{(i)}) \oplus A_3^{(i)}$.

e) For INIT mode,

set $B_{10}^{(i+1)}= ((\alpha_1^{A_2^{(i)}[30]} + \alpha_2^{1-A_2^{(i)}[30]} - 1) \otimes B_0^{(i)}) \oplus B_1^{(i)} \oplus B_6^{(i)} \oplus (\alpha_3^{A_2^{(i)}[31]} \otimes B_8^{(i)}) \oplus NLF(B_{10}^{(i)}, L2^{(i)}, L1^{(i)}, A_0^{(i)})$, For null mode,

set $B_{10}^{(i+1)} = ((\alpha_1^{A_2^{(i)}[30]} + \alpha_2^{1-A_2^{(i)}[30]} - 1) \otimes B_0^{(i)}) \oplus B_1^{(i)} \oplus B_6^{(i)} \oplus (\alpha_3^{A_2^{(i)}[31]} \otimes B_8^{(i)})$.

f) set $S_{i+1}=(A^{(i+1)}, B^{(i+1)}, R1^{(i+1)}, L1^{(i+1)}, R2^{(i+1)}, L2^{(i+1)})$.

g) Output S_{i+1}.

$A_m^{(i)}[Y]$ in $\{0,1\}$ denotes the Y-th bit of the register $A_m^{(i)}$, where $A_m^{(i)}$ [31] is the most significant bit of $A_m^{(i)}$. The description of the *Sub*$_{K2}$ function and the finite field arithmetic involving the fixed elements, α_0, α_1, α_2 and α_3 refers to 8.5.5, 8.5.6, 8.5.7, 8.5.8, and 8.5.9, respectively. Also, the definition of the *NLF* function refers to 8.5.10.

Figure 14 is a block diagram of the INIT mode of the *Next* function.

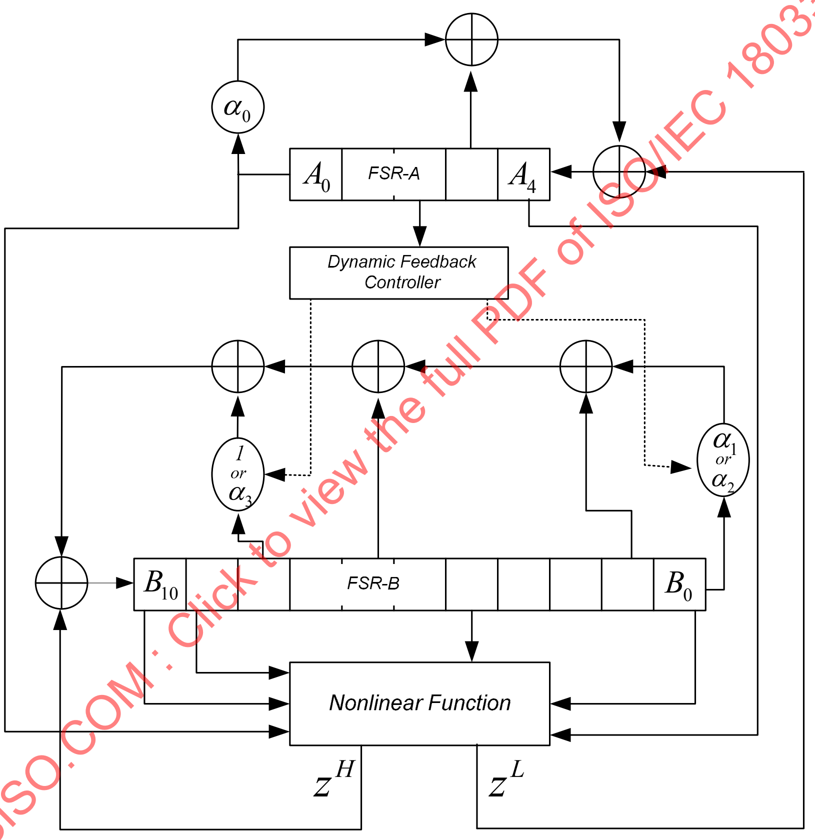

Figure 14 — INIT mode of the *Next* function

8.5.4 Keystream function *Strm*

The keystream function *Strm* works as follows.

Input: State variable $S_i=(A^{(i)}, B^{(i)}, R1^{(i)}, L1^{(i)}, R2^{(i)}, L2^{(i)})$.

Output: 64-bit keystream $Z_i=(Z_i^H, Z_i^L)$.

a) Set Z_i^H=*NLF*$(B_{10}^{(i)}, L2^{(i)}, L1^{(i)}, A_0^{(i)})$.

b) Set Z_i^L= *NLF*$(B_0^{(i)}, R2^{(i)}, R1^{(i)}, A_4^{(i)})$.

c) Set $Z_i=(Z_i^H, Z_i^L)$.

d) Output Z_i.

The function *NLF* is defined in 8.5.10.

8.5.5 Function *Sub*$_{K2}$

The *Sub*$_{K2}$ function is a permutation of $GF(2^{32})$, based on components from the Advanced Encryption Standard (AES) [ISO/IEC 18033-3]. In *Sub*$_{K2}$ function, the 32-bit input value is divided into four 1-byte strings and a nonlinear permutation is applied to each byte using an 8x8 bit substitution function (*SBox)* followed by a 32x32 bit linear permutation. The *SBox* function is the same as *SBox* of AES, and the permutation is the same as AES *Mix Column* operation.

NOTE 1 The AES *SBox* function, *SBox*, can be found in 8.1.8 as the function *SUB*.

NOTE 2 Function *Sub*$_{K2}$ produces the same output as function *T* of 8.2.5.

Input: A 32-bit value w.

Output: A 32-bit string q=*Sub*$_{K2}$(w).

Local variables: counter m.

a) Set $w=(w_3, w_2, w_1, w_0)$, where each w_m is 8-bit.

b) For $m=0,1,2,3$, set t_m=*SBox*(w_m).

c) Set $q=(q_3, q_2, q_1, q_0)$ as follows.

$$\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

Multiplication of the elements t_i is performed in $GF(2^8)$ making use of the irreducible polynomial $f(x)=x^8+x^4+x^3+x+1$.

d) Output q.

8.5.6 Multiplication of α_0 in $GF(2^{32})$

Input: A 32-bit value w, which represents an element of $GF(2^{32})$.

Output: A 32-bit string w', which represents $\alpha_0 \otimes w$ in $GF(2^{32})$.

a) Set $w' = (w <<_{32} 8) \oplus \alpha_{\text{MUL0}}[w >>_{32} 24]$.

b) Output w'.

The function α_{MUL0} is defined in the following:

```
αMUL0[256]={
0x00000000,0xB6086D1A,0xAF10DA34,0x1918B72E,0x9D207768,0x2B281A72,0x3230AD5C,
0x8438C046,0xF940EED0,0x4F4883CA,0x565034E4,0xE05859FE,0x646099B8,0xD268F4A2,
0xCB70438C,0x7D782E96,0x31801F63,0x87887279,0x9E90C557,0x2898A84D,0xACA0680B,
0x1AA80511,0x03B0B23F,0xB5B8DF25,0xC8C0F1B3,0x7EC89CA9,0x67D02B87,0xD1D8469D,
0x55E086DB,0xE3E8EBC1,0xFAF05CEF,0x4CF831F5,0x62C33EC6,0xD4CB53DC,0xCDD3E4F2,
0x7BDB89E8,0xFFE349AE,0x49EB24B4,0x50F3939A,0xE6FBFE80,0x9B83D016,0x2D8BBD0C,
0x34930A22,0x829B6738,0x06A3A77E,0xB0ABCA64,0xA9B37D4A,0x1FBB1050,0x534321A5,
0xE54B4CBF,0xFC53FB91,0x4A5B968B,0xCE6356CD,0x786B3BD7,0x61738CF9,0xD77BE1E3,
0xAA03CF75,0x1C0BA26F,0x05131541,0xB31B785B,0x3723B81D,0x812BD507,0x98336229,
0x2E3B0F33,0xC4457C4F,0x724D1155,0x6B55A67B,0xDD5DCB61,0x59650B27,0xEF6D663D,
0xF675D113,0x407DBC09,0x3D05929F,0x8B0DFF85,0x921548AB,0x241D25B1,0xA025E5F7,
0x162D88ED,0x0F353FC3,0xB93D52D9,0xF5C5632C,0x43CD0E36,0x5AD5B918,0xECDDD402,
0x68E51444,0xDEED795E,0xC7F5CE70,0x71FDA36A,0x0C858DFC,0xBA8DE0E6,0xA39557C8,
0x159D3AD2,0x91A5FA94,0x27AD978E,0x3EB520A0,0x88BD4DBA,0xA6864289,0x108E2F93,
0x099698BD,0xBF9EF5A7,0x3BA635E1,0x8DAE58FB,0x94B6EFD5,0x22BE82CF,0x5FC6AC59,
0xE9CEC143,0xF0D6766D,0x46DE1B77,0xC2E6DB31,0x74EEB62B,0x6DF60105,0xDBFE6C1F,
0x97065DEA,0x210E30F0,0x381687DE,0x8E1EEAC4,0x0A262A82,0xBC2E4798,0xA536F0B6,
0x133E9DAC,0x6E46B33A,0xD84EDE20,0xC156690E,0x775E0414,0xF366C452,0x456EA948,
0x5C761E66,0xEA7E737C,0x4B8AF89E,0xFD829584,0xE49A22AA,0x52924FB0,0xD6AA8FF6,
0x60A2E2EC,0x79BA55C2,0xCFB238D8,0xB2CA164E,0x04C27B54,0x1DDACC7A,0xABD2A160,
0x2FEA6126,0x99E20C3C,0x80FABB12,0x36F2D608,0x7A0AE7FD,0xCC028AE7,0xD51A3DC9,
0x631250D3,0xE72A9095,0x5122FD8F,0x483A4AA1,0xFE3227BB,0x834A092D,0x35426437,
0x2C5AD319,0x9A52BE03,0x1E6A7E45,0xA862135F,0xB17AA471,0x0772C96B,0x2949C658,
0x9F41AB42,0x86591C6C,0x30517176,0xB469B130,0x0261DC2A,0x1B796B04,0xAD71061E,
0xD0092888,0x66014592,0x7F19F2BC,0xC9119FA6,0x4D295FE0,0xFB2132FA,0xE23985D4,
0x5431E8CE,0x18C9D93B,0xAEC1B421,0xB7D9030F,0x01D16E15,0x85E9AE53,0x33E1C349,
0x2AF97467,0x9CF1197D,0xE18937EB,0x57815AF1,0x4E99EDDF,0xF89180C5,0x7CA94083,
0xCAA12D99,0xD3B99AB7,0x65B1F7AD,0x8FCF84D1,0x39C7E9CB,0x20DF5EE5,0x96D733FF,
0x12EFF3B9,0xA4E79EA3,0xBDFF298D,0x0BF74497,0x768F6A01,0xC087071B,0xD99FB035,
0x6F97DD2F,0xEBAF1D69,0x5DA77073,0x44BFC75D,0xF2B7AA47,0xBE4F9BB2,0x0847F6A8,
0x115F4186,0xA7572C9C,0x236FECDA,0x956781C0,0x8C7F36EE,0x3A775BF4,0x470F7562,
0xF1071878,0xE81FAF56,0x5E17C24C,0xDA2F020A,0x6C276F10,0x753FD83E,0xC337B524,
0xED0CBA17,0x5B04D70D,0x421C6023,0xF4140D39,0x702CCD7F,0xC624A065,0xDF3C174B,
0x69347A51,0x144C54C7,0xA24439DD,0xBB5C8EF3,0x0D54E3E9,0x896C23AF,0x3F644EB5,
0x267CF99B,0x90749481,0xDC8CA574,0x6A84C86E,0x739C7F40,0xC594125A,0x41ACD21C,
0xF7A4BF06,0xEEBC0828,0x58B46532,0x25CC4BA4,0x93C426BE,0x8ADC9190,0x3CD4FC8A,
0xB8EC3CCC,0x0EE451D6,0x17FCE6F8,0xA1F48BE2};
```

8.5.7 Multiplication of α_1 in $GF(2^{32})$

Input: A 32-bit value w, which represents an element of $GF(2^{32})$.

Output: A 32-bit string w', which represents $\alpha_1 \otimes w$ in $GF(2^{32})$.

a) Set $w' = (w <<_{32} 8) \oplus \alpha_{\mathrm{MUL1}}[w >>_{32} 24]$.

b) Output w'.

The function α_{MUL1} is defined in the following:

α_{MUL1}[256]={

```
0x00000000,0xA0F5FC2E,0x6DC7D55C,0xCD322972,0xDAA387B8,0x7A567B96,0xB76452E4,
0x1791AECA,0x996B235D,0x399EDF73,0xF4ACF601,0x54590A2F,0x43C8A4E5,0xE33D58CB,
0x2E0F71B9,0x8EFA8D97,0x1FD646BA,0xBF23BA94,0x721193E6,0xD2E46FC8,0xC575C102,
0x65803D2C,0xA8B2145E,0x0847E870,0x86BD65E7,0x264899C9,0xEB7AB0BB,0x4B8F4C95,
0x5C1EE25F,0xFCEB1E71,0x31D93703,0x912CCB2D,0x3E818C59,0x9E747077,0x53465905,
0xF3B3A52B,0xE4220BE1,0x44D7F7CF,0x89E5DEBD,0x29102293,0xA7EAAF04,0x071F532A,
0xCA2D7A58,0x6AD88676,0x7D4928BC,0xDDBCD492,0x108EFDE0,0xB07B01CE,0x2157CAE3,
0x81A236CD,0x4C901FBF,0xEC65E391,0xFBF44D5B,0x5B01B175,0x96339807,0x36C66429,
0xB83CE9BE,0x18C91590,0xD5FB3CE2,0x750EC0CC,0x629F6E06,0xC26A9228,0x0F58BB5A,
0xAFAD4774,0x7C2F35B2,0xDCDAC99C,0x11E8E0EE,0xB11D1CC0,0xA68CB20A,0x06794E24,
0xCB4B6756,0x6BBE9B78,0xE54416EF,0x45B1EAC1,0x8883C3B3,0x28763F9D,0x3FE79157,
0x9F126D79,0x5220440B,0xF2D5B825,0x63F97308,0xC30C8F26,0x0E3EA654,0xAECB5A7A,
0xB95AF4B0,0x19AF089E,0xD49D21EC,0x7468DDC2,0xFA925055,0x5A67AC7B,0x97558509,
0x37A07927,0x2031D7ED,0x80C42BC3,0x4DF602B1,0xED03FE9F,0x42AEB9EB,0xE25B45C5,
0x2F696CB7,0x8F9C9099,0x980D3E53,0x38F8C27D,0xF5CAEB0F,0x553F1721,0xDBC59AB6,
0x7B306698,0xB6024FEA,0x16F7B3C4,0x01661D0E,0xA193E120,0x6CA1C852,0xCC54347C,
0x5D78FF51,0xFD8D037F,0x30BF2A0D,0x904AD623,0x87DB78E9,0x272E84C7,0xEA1CADB5,
0x4AE9519B,0xC413DC0C,0x64E62022,0xA9D40950,0x0921F57E,0x1EB05BB4,0xBE45A79A,
0x73778EE8,0xD38272C6,0xF85E6A49,0x58AB9667,0x9599BF15,0x356C433B,0x22FDEDF1,
0x820811DF,0x4F3A38AD,0xEFCFC483,0x61354914,0xC1C0B53A,0x0CF29C48,0xAC076066,
0xBB96CEAC,0x1B633282,0xD6511BF0,0x76A4E7DE,0xE7882CF3,0x477DD0DD,0x8A4FF9AF,
0x2ABA0581,0x3D2BAB4B,0x9DDE5765,0x50EC7E17,0xF0198239,0x7EE30FAE,0xDE16F380,
0x1324DAF2,0xB3D126DC,0xA4408816,0x04B57438,0xC9875D4A,0x6972A164,0xC6DFE610,
0x662A1A3E,0xAB18334C,0x0BEDCF62,0x1C7C61A8,0xBC899D86,0x71BBB4F4,0xD14E48DA,
0x5FB4C54D,0xFF413963,0x32731011,0x9286EC3F,0x851742F5,0x25E2BEDB,0xE8D097A9,
0x48256B87,0xD909A0AA,0x79FC5C84,0xB4CE75F6,0x143B89D8,0x03AA2712,0xA35FDB3C,
0x6E6DF24E,0xCE980E60,0x406283F7,0xE0977FD9,0x2DA556AB,0x8D50AA85,0x9AC1044F,
0x3A34F861,0xF706D113,0x57F32D3D,0x84715FFB,0x2484A3D5,0xE9B68AA7,0x49437689,
0x5ED2D843,0xFE27246D,0x33150D1F,0x93E0F131,0x1D1A7CA6,0xBDEF8088,0x70DDA9FA,
0xD02855D4,0xC7B9FB1E,0x674C0730,0xAA7E2E42,0x0A8BD26C,0x9BA71941,0x3B52E56F,
0xF660CC1D,0x56953033,0x41049EF9,0xE1F162D7,0x2CC34BA5,0x8C36B78B,0x02CC3A1C,
0xA239C632,0x6F0BEF40,0xCFFE136E,0xD86FBDA4,0x789A418A,0xB5A868F8,0x155D94D6,
0xBAF0D3A2,0x1A052F8C,0xD73706FE,0x77C2FAD0,0x6053541A,0xC0A6A834,0x0D948146,
0xAD617D68,0x239BF0FF,0x836E0CD1,0x4E5C25A3,0xEEA9D98D,0xF9387747,0x59CD8B69,
0x94FFA21B,0x340A5E35,0xA5269518,0x05D36936,0xC8E14044,0x6814BC6A,0x7F8512A0,
0xDF70EE8E,0x1242C7FC,0xB2B73BD2,0x3C4DB645,0x9CB84A6B,0x518A6319,0xF17F9F37,
0xE6EE31FD,0x461BCDD3,0x8B29E4A1,0x2BDC188F};
```

8.5.8 Multiplication of α_2 in $GF(2^{32})$

Input: A 32-bit value w, which represents an element of $GF(2^{32})$.

Output: A 32-bit string w', which represents $\alpha_2 \otimes w$ in $GF(2^{32})$.

a) Set $w' = (w <<_{32} 8) \oplus \alpha_{\mathrm{MUL2}}[w >>_{32} 24]$.

b) Output w'.

The function α_{MUL2} is defined in the following:

```
αMUL2[256]={
0x00000000,0x5BF87F93,0xB6BDFE6B,0xED4581F8,0x2137B1D6,0x7ACFCE45,0x978A4FBD,
0xCC72302E,0x426E2FE1,0x19965072,0xF4D3D18A,0xAF2BAE19,0x63599E37,0x38A1E1A4,
0xD5E4605C,0x8E1C1FCF,0x84DC5E8F,0xDF24211C,0x3261A0E4,0x6999DF77,0xA5EBEF59,
0xFE1390CA,0x13561132,0x48AE6EA1,0xC6B2716E,0x9D4A0EFD,0x700F8F05,0x2BF7F096,
0xE785C0B8,0xBC7DBF2B,0x51383ED3,0x0AC04140,0x45F5BC53,0x1E0DC3C0,0xF3484238,
0xA8B03DAB,0x64C20D85,0x3F3A7216,0xD27FF3EE,0x89878C7D,0x079B93B2,0x5C63EC21,
0xB1266DD9,0xEADE124A,0x26AC2264,0x7D545DF7,0x9011DC0F,0xCBE9A39C,0xC129E2DC,
0x9AD19D4F,0x77941CB7,0x2C6C6324,0xE01E530A,0xBBE62C99,0x56A3AD61,0x0D5BD2F2,
0x8347CD3D,0xD8BFB2AE,0x35FA3356,0x6E024CC5,0xA2707CEB,0xF9880378,0x14CD8280,
0x4F35FD13,0x8AA735A6,0xD15F4A35,0x3C1ACBCD,0x67E2B45E,0xAB908470,0xF068FBE3,
0x1D2D7A1B,0x46D50588,0xC8C91A47,0x933165D4,0x7E74E42C,0x258C9BBF,0xE9FEAB91,
0xB206D402,0x5F4355FA,0x04BB2A69,0x0E7B6B29,0x558314BA,0xB8C69542,0xE33EEAD1,
0x2F4CDAFF,0x74B4A56C,0x99F12494,0xC2095B07,0x4C1544C8,0x17ED3B5B,0xFAA8BAA3,
0xA150C530,0x6D22F51E,0x36DA8A8D,0xDB9F0B75,0x806774E6,0xCF5289F5,0x94AAF666,
0x79EF779E,0x2217080D,0xEE653823,0xB59D47B0,0x58D8C648,0x0320B9DB,0x8D3CA614,
0xD6C4D987,0x3B81587F,0x607927EC,0xAC0B17C2,0xF7F36851,0x1AB6E9A9,0x414E963A,
0x4B8ED77A,0x1076A8E9,0xFD332911,0xA6CB5682,0x6AB966AC,0x3141193F,0xDC0498C7,
0x87FCE754,0x09E0F89B,0x52188708,0xBF5D06F0,0xE4A57963,0x28D7494D,0x732F36DE,
0x9E6AB726,0xC592C8B5,0x59036A01,0x02FB1592,0xEFBE946A,0xB446EBF9,0x7834DBD7,
0x23CCA444,0xCE8925BC,0x95715A2F,0x1B6D45E0,0x40953A73,0xADD0BB8B,0xF628C418,
0x3A5AF436,0x61A28BA5,0x8CE70A5D,0xD71F75CE,0xDDDF348E,0x86274B1D,0x6B62CAE5,
0x309AB576,0xFCE88558,0xA710FACB,0x4A557B33,0x11AD04A0,0x9FB11B6F,0xC44964FC,
0x290CE504,0x72F49A97,0xBE86AAB9,0xE57ED52A,0x083B54D2,0x53C32B41,0x1CF6D652,
0x470EA9C1,0xAA4B2839,0xF1B357AA,0x3DC16784,0x66391817,0x8B7C99EF,0xD084E67C,
0x5E98F9B3,0x05608620,0xE82507D8,0xB3DD784B,0x7FAF4865,0x245737F6,0xC912B60E,
0x92EAC99D,0x982A88DD,0xC3D2F74E,0x2E9776B6,0x756F0925,0xB91D390B,0xE2E54698,
0x0FA0C760,0x5458B8F3,0xDA44A73C,0x81BCD8AF,0x6CF95957,0x370126C4,0xFB7316EA,
0xA08B6979,0x4DCEE881,0x16369712,0xD3A45FA7,0x885C2034,0x6519A1CC,0x3EE1DE5F,
0xF293EE71,0xA96B91E2,0x442E101A,0x1FD66F89,0x91CA7046,0xCA320FD5,0x27778E2D,
0x7C8FF1BE,0xB0FDC190,0xEB05BE03,0x06403FFB,0x5DB84068,0x57780128,0x0C807EBB,
0xE1C5FF43,0xBA3D80D0,0x764FB0FE,0x2DB7CF6D,0xC0F24E95,0x9B0A3106,0x15162EC9,
0x4EEE515A,0xA3ABD0A2,0xF853AF31,0x34219F1F,0x6FD9E08C,0x829C6174,0xD9641EE7,
0x9651E3F4,0xCDA99C67,0x20EC1D9F,0x7B14620C,0xB7665222,0xEC9E2DB1,0x01DBAC49,
0x5A23D3DA,0xD43FCC15,0x8FC7B386,0x6282327E,0x397A4DED,0xF5087DC3,0xAEF00250,
0x43B583A8,0x184DFC3B,0x128DBD7B,0x4975C2E8,0xA4304310,0xFFC83C83,0x33BA0CAD,
0x6842733E,0x8507F2C6,0xDEFF8D55,0x50E3929A,0x0B1BED09,0xE65E6CF1,0xBDA61362,
0x71D4234C,0x2A2C5CDF,0xC769DD27,0x9C91A2B4};
```

8.5.9 Multiplication of α_3 in $GF(2^{32})$

Input: A 32-bit value w, which represents an element of $GF(2^{32})$.

Output: A 32-bit string w', which represents $\alpha_3 \otimes w$ in $GF(2^{32})$.

a) Set $w' = (w <<_{32} 8) \oplus \alpha_{\mathrm{MUL3}}[w >>_{32} 24]$.

b) Output w'.

The function α_{MUL3} is defined in the following:

```
αMUL3[256]={
0x00000000,0x4559568B,0x8AB2AC73,0xCFEBFAF8,0x71013DE6,0x34586B6D,0xFBB39195,
0xBEEAC71E,0xE2027AA9,0xA75B2C22,0x68B0D6DA,0x2DE98051,0x9303474F,0xD65A11C4,
0x19B1EB3C,0x5CE8BDB7,0xA104F437,0xE45DA2BC,0x2BB65844,0x6EEF0ECF,0xD005C9D1,
0x955C9F5A,0x5AB765A2,0x1FEE3329,0x43068E9E,0x065FD815,0xC9B422ED,0x8CED7466,
0x3207B378,0x775EE5F3,0xB8B51F0B,0xFDEC4980,0x27088D6E,0x6251DBE5,0xADBA211D,
0xE8E37796,0x5609B088,0x1350E603,0xDCBB1CFB,0x99E24A70,0xC50AF7C7,0x8053A14C,
0x4FB85BB4,0x0AE10D3F,0xB40BCA21,0xF1529CAA,0x3EB96652,0x7BE030D9,0x860C7959,
0xC3552FD2,0x0CBED52A,0x49E783A1,0xF70D44BF,0xB2541234,0x7DBFE8CC,0x38E6BE47,
0x640E03F0,0x2157557B,0xEEBCAF83,0xABE5F908,0x150F3E16,0x5056689D,0x9FBD9265,
0xDAE4C4EE,0x4E107FDC,0x0B492957,0xC4A2D3AF,0x81FB8524,0x3F11423A,0x7A4814B1,
0xB5A3EE49,0xF0FAB8C2,0xAC120575,0xE94B53FE,0x26A0A906,0x63F9FF8D,0xDD133893,
0x984A6E18,0x57A194E0,0x12F8C26B,0xEF148BEB,0xAA4DDD60,0x65A62798,0x20FF7113,
0x9E15B60D,0xDB4CE086,0x14A71A7E,0x51FE4CF5,0x0D16F142,0x484FA7C9,0x87A45D31,
0xC2FD0BBA,0x7C17CCA4,0x394E9A2F,0xF6A560D7,0xB3FC365C,0x6918F2B2,0x2C41A439,
0xE3AA5EC1,0xA6F3084A,0x1819CF54,0x5D4099DF,0x92AB6327,0xD7F235AC,0x8B1A881B,
0xCE43DE90,0x01A82468,0x44F172E3,0xFA1BB5FD,0xBF42E376,0x70A9198E,0x35F04F05,
0xC81C0685,0x8D45500E,0x42AEAAF6,0x07F7FC7D,0xB91D3B63,0xFC446DE8,0x33AF9710,
0x76F6C19B,0x2A1E7C2C,0x6F472AA7,0xA0ACD05F,0xE5F586D4,0x5B1F41CA,0x1E461741,
0xD1ADEDB9,0x94F4BB32,0x9C20FEDD,0xD979A856,0x169252AE,0x53CB0425,0xED21C33B,
0xA87895B0,0x67936F48,0x22CA39C3,0x7E228474,0x3B7BD2FF,0xF4902807,0xB1C97E8C,
0x0F23B992,0x4A7AEF19,0x859115E1,0xC0C8436A,0x3D240AEA,0x787D5C61,0xB796A699,
0xF2CFF012,0x4C25370C,0x097C6187,0xC6979B7F,0x83CECDF4,0xDF267043,0x9A7F26C8,
0x5594DC30,0x10CD8ABB,0xAE274DA5,0xEB7E1B2E,0x2495E1D6,0x61CCB75D,0xBB2873B3,
0xFE712538,0x319ADFC0,0x74C3894B,0xCA294E55,0x8F7018DE,0x409BE226,0x05C2B4AD,
0x592A091A,0x1C735F91,0xD398A569,0x96C1F3E2,0x282B34FC,0x6D726277,0xA299988F,
0xE7C0CE04,0x1A2C8784,0x5F75D10F,0x909E2BF7,0xD5C77D7C,0x6B2DBA62,0x2E74ECE9,
0xE19F1611,0xA4C6409A,0xF82EFD2D,0xBD77ABA6,0x729C515E,0x37C507D5,0x892FC0CB,
0xCC769640,0x039D6CB8,0x46C43A33,0xD2308101,0x9769D78A,0x58822D72,0x1DDB7BF9,
0xA331BCE7,0xE668EA6C,0x29831094,0x6CDA461F,0x3032FBA8,0x756BAD23,0xBA8057DB,
0xFFD90150,0x4133C64E,0x046A90C5,0xCB816A3D,0x8ED83CB6,0x73347536,0x366D23BD,
0xF986D945,0xBCDF8FCE,0x023548D0,0x476C1E5B,0x8887E4A3,0xCDDEB228,0x91360F9F,
0xD46F5914,0x1B84A3EC,0x5EDDF567,0xE0373279,0xA56E64F2,0x6A859E0A,0x2FDCC881,
0xF5380C6F,0xB0615AE4,0x7F8AA01C,0x3AD3F697,0x84393189,0xC1606702,0x0E8B9DFA,
0x4BD2CB71,0x173A76C6,0x5263204D,0x9D88DAB5,0xD8D18C3E,0x663B4B20,0x23621DAB,
0xEC89E753,0xA9D0B1D8,0x543CF858,0x1165AED3,0xDE8E542B,0x9BD702A0,0x253DC5BE,
0x60649335,0xAF8F69CD,0xEAD63F46,0xB63E82F1,0xF367D47A,0x3C8C2E82,0x79D57809,
0xC73FBF17,0x8266E99C,0x4D8D1364,0x08D445EF};
```

8.5.10 Function *NLF*(a,b,c,d)

Input: Four 32-bit values, a, b, c and d.

Output: A 32-bit string q.

a) Set q=($a +_{32} b$)$\oplus c \oplus d$.

b) Output q.

Annex A
(normative)

Object Identifiers

This annex lists the object identifiers assigned to algorithms specified in this part of ISO/IEC 18033 and defines algorithm parameter structures. Please refer to ISO/IEC 18033-3 for Object IDs for modes of a block cipher.

```
EncryptionAlgorithms-4 {
   iso(1) standard(0) encryption-algorithms(18033) part(4)
      asn1-module(0) algorithm-object-identifiers(0) }
   DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- EXPORTS All; --

-- IMPORTS None; --

OID ::= OBJECT IDENTIFIER -- Alias

-- Synonyms --

is18033-4 OID ::= { iso(1) standard(0) is18033(18033) part4(4) }

id-kg OID ::= { is18033-4 keystream-generator(1) }
id-scmode OID ::= { is18033-4 stream-cipher-mode(2) }

-- Assignments --

id-kg-mugi OID ::= { id-kg mugi(1) }
id-kg-snow OID ::= { id-kg snow(2) }
id-kg-rabbit OID ::= { id-kg rabbit(3) }
id-kg-decim2 OID ::= { id-kg decim2(4) }
id-kg-k2 OID ::= { id-kg k2(5) }

id-scmode-additive OID ::= { id-scmode additive(1) }
id-scmode-multis01 OID ::= { id-scmode multis01(2) }

-- Algorithms and parameters --

StreamCipher ::= AlgorithmIdentifier {{ StreamCipherAlgorithms }}

StreamCipherAlgorithms ALGORITHM ::= {
  additiveStreamCipher |
  multiS01StreamCipher,

  ... -- Expect additional algorithms --
}

additiveStreamCipher ALGORITHM ::= {
   OID id-scmode-additive PARMS AdditiveStreamCipherParameters
}

AdditiveStreamCipherParameters ::= KeyGenerator

multiS01StreamCipher ALGORITHM ::= {
   OID id-scmode-multis01 PARMS MultiS01StreamCipherParameters
}
```

```
MultiS01StreamCipherParameters ::= SEQUENCE {
   keyGenerator KeyGenerator,
   securityParameter INTEGER DEFAULT 64,
   irreduciblePolynoial BIT STRING,
   redandancy BIT STRING,
   publicParameterR BIT STRING
      -- length determined by securityParameter
      -- for full interoperability multis01 parameters should
      -- include the padding method but they do not have object
      -- identifiers. for the time being they will have to be
      -- negotiated in an application-dependent way
}

KeyGenerator ALGORITHM ::= {
   mugiKeyGenerator |
   snowKeyGenerator |
   rabbitKeyGenerator |
   decim2KeyGenerator |
   k2KeyGenerator,

   ... -- Expect additional algorithms --
}

mugiKeyGenerator ALGORITHM ::= {
   OID id-kg-mugi PARMS NullParameters
}

snowKeyGenerator ALGORITHM ::= {
   OID id-kg-snow PARMS NullParameters
}

rabbitKeyGenerator ALGORITHM ::={
   OID id-kg-rabbit PARMS NullParameters
}

decim2KeyGenerator ALGORITHM ::={
   OID id-kg-decim2 PARMS NullParameters
}

k2KeyGenerator ALGORITHM ::={
   OID id-kg-k2 PARMS NullParameters
}

NullParameters ::= NULL

-- Cryptographic algorithm identification --

ALGORITHM ::= CLASS {
   &id OBJECT IDENTIFIER UNIQUE,
   &Type OPTIONAL
}
   WITH SYNTAX { OID &id [PARMS &Type] }

AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
   algorithm ALGORITHM.&id( {IOSet} ),
   parameters ALGORITHM.&Type( {IOSet}{@algorithm} ) OPTIONAL
}

END -- EncryptionAlgorithms-4 --
```

Annex B
(informative)

Operations over the finite field $GF(2^n)$

For any positive integer n there exists a finite field containing exactly 2^n elements. This field is unique up to isomorphism, and in this part of ISO/IEC 18033 it is referred to as the finite field $GF(2^n)$.

In the polynomial representation, each element of $GF(2^n)$ is represented by a binary polynomial of degree less than n. More explicitly the bit string $a = a_{n-1}\ldots a_2a_1a_0$ is taken to represent the binary polynomial $a(x) = a_{n-1}x^{n-1} +\ldots+ a_2x^2 + a_1x + a_0$. The polynomial basis is the set $B = (x^{n-1},\ldots, x^2, x, 1)$. For two bit strings $a = a_{n-1}\ldots a_2a_1a_0$ and $b = b_{n-1}\ldots b_2b_1b_0$, the sum is $c = a \oplus b = c_{n-1}\ldots c_2c_1c_0$, where $c_i = a_i \oplus b_i$.

Multiplication in the finite field, written $a \otimes b$, corresponds to the multiplication of two polynomials $a(x)b(x)$ modulo a binary irreducible polynomial $p(x)$ of degree n. A polynomial is irreducible if it has no non-trivial divisors.

$GF(2^n)\backslash\{0\}$ denoted as $GF(2^n)^*$ is an abelian group with respect to multiplication and the identity is 1. For any non-zero binary polynomial $b(x)$ of degree less than n, the multiplicative inverse of $b(x)$, denoted $b^{-1}(x)$, can be computed as follows: the extended Euclidean algorithm is used to compute polynomials $a(x)$ and $c(x)$ such that $b(x)\bullet a(x) + p(x)\bullet c(x) = 1$. Hence, $a(x)\bullet b(x) \bmod p(x) = 1$, which means $b^{-1}(x) = a(x) \bmod p(x)$. The extended Euclidean algorithm is described in [15].

Annex C
(informative)

Examples

C.1 Example for MUGI

C.1.1 Key, initialization vector, and keystream triplets

```
K = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Z = c7 6e 14 e7 08 36 e6 b6 cb 0e 9c 5a 0b f0 3e 1e 0a cf 9a f4 9e be 6d 67 d5 72 6e 37 4b 13 97 ac.

K = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV= 34 61 69 88 51 81 21 39 01 55 00 a5 3b 7e 59 87
Z = 2a a1 c5 c7 20 73 b1 b3 a9 d1 0d c6 85 50 66 10 28 30 56 0d 9a 24 65 c9 9c 29 1c 13 81 4e 08 8d.

K = 51 34 00 b1 04 a0 59 91 30 ad 00 fc 48 d7 59 e0
IV= 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Z = bd df ad 5f 04 b8 86 25 c3 ad ac e1 56 d1 c1 99 36 ff a4 e9 a7 fd f7 5a aa b8 29 13 42 85 aa 4b.

K = 69 e7 06 ee 52 95 37 2c 75 13 01 47 30 23 79 93
IV= 2a 00 45 c8 49 27 49 d5 3a 9b 16 4a 25 e4 49 15
Z = e3 cc 67 a0 25 5b 0f 28 2d 9a 5b 1b bd f7 f2 df 84 eb 46 f6 07 d6 e6 dd 32 86 13 43 94 dd 95 fb.
```

C.1.2 Sample internal states

```
K = 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
IV= f0 e0 d0 c0 b0 a0 90 80 70 60 50 40 30 20 10 00
Z = bc 62 43 06 14 b7 9b 71 71 a6 66 81 c3 55 42 de 7a ba 5b 4f b8 0e 82 d7 0b 96 98 28 90 b6 e1 43

Intermediate values of the internal state

rho function 0
a: 0001020304050607 08090a0b0c0d0e0f 7498f5f1e727d094
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000

rho function 1
a: 08090a0b0c0d0e0f 9724d9144c5d8926 64b47311d52100a5
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 08090a0b0c0d0e0f

rho function 2
a: 9724d9144c5d8926 09671cfbcfaa95fb e2d338166cd8c441
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 9724d9144c5d8926 08090a0b0c0d0e0f
```

```
rho function 3
a: 09671cfbcfaa95fb 9c0c2097edb20067 6ef29c62b7691210
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 4
a: 9c0c2097edb20067 c08ee4dcb2d08591 201239b2b04d5d6a
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 5
a: c08ee4dcb2d08591 738177859f3210f6 48963357b89312eb
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 6
a: 738177859f3210f6 b36b4d944f5d04cb bc7ac7e83f40cca1
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 7
a: b36b4d944f5d04cb 2d13c00221057d8d 65e12d98fb29feca
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 8
a: 2d13c00221057d8d 20ead0479e63cdc3 7169edbc504968d2
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 0000000000000000
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 9
a: 20ead0479e63cdc3 591a6857e3112cee 8269181ee80366a1
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 0000000000000000 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 10
a: 591a6857e3112cee dfbbb88c02c9c80a fa312d220ef73c78
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 0000000000000000 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 11
a: dfbbb88c02c9c80a 5cc4835080bc5321 78e69bd217041ca7
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   0000000000000000 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 12
a: 5cc4835080bc5321 fd5755df9cc0ceb9 dd032b76f3534504
b: 0000000000000000 0000000000000000 0000000000000000 0000000000000000
   5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 13
a: fd5755df9cc0ceb9 c905d08f50fa71db cfcb255e594b38ee
b: 0000000000000000 0000000000000000 0000000000000000 fd5755df9cc0ceb9
   5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 14
a: c905d08f50fa71db bfe2485ac2696cc7 0a77652c7dbcc580
b: 0000000000000000 0000000000000000 c905d08f50fa71db fd5755df9cc0ceb9
   5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 15
a: bfe2485ac2696cc7 7dea261cb61d4fea 3991ce48e105a4a1
b: 0000000000000000 bfe2485ac2696cc7 c905d08f50fa71db fd5755df9cc0ceb9
   5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

buffer init
a: 7dea261cb61d4fea eafb528479bb687d eb8189612089ff0b
b: 7dea261cb61d4fea bfe2485ac2696cc7 c905d08f50fa71db fd5755df9cc0ceb9
   5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

rho function 0
a: 8d0af6dc06bddf6a 9a9b02c4499b787d f100cffe031d365b

rho function 1
a: 9a9b02c4499b787d 435407f3bbc2c760 b8576326c43c7141

rho function 2
a: 435407f3bbc2c760 b5117172dcf5e507 10d44d672b0cb32b

rho function 3
a: b5117172dcf5e507 9157292760b2892f 45de3e448a22a274

rho function 4
a: 9157292760b2892f aee0542493e7889e d92646e5bf6e90fd

rho function 5
a: aee0542493e7889e a9f2f7fac6cff1ff 668ac5cf634db73d

rho function 6
a: a9f2f7fac6cff1ff 9cb8969f9fc84dc6 d3db5a83153c2d75
```

```
rho function 7
a: 9cb8969f9fc84dc6 b1260b2ec980a340 4c06fba0602d20da

rho function 8
a: b1260b2ec980a340 192a6fd877969848 4e9d5f10f22daa44

rho function 9
a: 192a6fd877969848 bbac287d38601209 c31e21b47993441d

rho function 10
a: bbac287d38601209 d58486545129be34 88b995cf25723d71

rho function 11
a: d58486545129be34 c8af8f1422e98119 7cb36f5145a5f171

rho function 12
a: c8af8f1422e98119 00bba312081aa445 2e8517e066c8b410

rho function 13
a: 00bba312081aa445 2f3864a9c279a14c 4e1ba1aafc06cb55

rho function 14
a: 2f3864a9c279a14c 6551f5e9cbc1e0d7 acf8aaa64583d0d7

rho function 15
a: 6551f5e9cbc1e0d7 4e466dffcb92db48 4a8ffe073636f5c3

state init
a: 4e466dffcb92db48 f5eb67b928359d8b 5d3c31a0af9cd78f
b: 7dea261cb61d4fea bfe2485ac2696cc7 c905d08f50fa71db fd5755df9cc0ceb9
   5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee 20ead0479e63cdc3
   2d13c00221057d8d b36b4d944f5d04cb 738177859f3210f6 c08ee4dcb2d08591
   9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926 08090a0b0c0d0e0f

update 1
a: f5eb67b928359d8b ace6a90bde0af786 529108c358fa4ada
b: 464f67f4c79fd547 7dea261cb61d4fea bfe2485ac2696cc7 c905d08f50fa71db
   ddbd859802a3037a 5cc4835080bc5321 dfbbb88c02c9c80a 591a6857e3112cee
   20ead0479e63cdc3 2d13c00221057d8d 7cc1d86f463a1830 738177859f3210f6
   c08ee4dcb2d08591 9c0c2097edb20067 09671cfbcfaa95fb 9724d9144c5d8926

update 2
a: ace6a90bde0af786 9fa7a15367ae1667 5f241cf311a0bfa7
b: 62cfbead646814ad 464f67f4c79fd547 7dea261cb61d4fea bfe2485ac2696cc7
   901fb8d8b3eb5d35 ddbd859802a3037a 5cc4835080bc5321 dfbbb88c02c9c80a
   591a6857e3112cee 20ead0479e63cdc3 c0a1c065bd095d1a 7cc1d86f463a1830
   738177859f3210f6 c08ee4dcb2d08591 9c0c2097edb20067 09671cfbcfaa95fb

update 3
a: 9fa7a15367ae1667 75195c2e249e4399 8bd43dd671ad8b05
b: a581b5f011a0627d 62cfbead646814ad 464f67f4c79fd547 7dea261cb61d4fea
   6059f0d6c0a0a4cd 901fb8d8b3eb5d35 ddbd859802a3037a 5cc4835080bc5321
   dfbbb88c02c9c80a 591a6857e3112cee 923a55d65eed291f c0a1c065bd095d1a
   7cc1d86f463a1830 738177859f3210f6 c08ee4dcb2d08591 9c0c2097edb20067

update 4
a: 75195c2e249e4399 9b2239a28cc5a4e3 5554ab4e803a0a19
b: 03ab81c48a1c1600 a581b5f011a0627d 62cfbead646814ad 464f67f4c79fd547
   212ea54c36a11ccb 6059f0d6c0a0a4cd 901fb8d8b3eb5d35 ddbd859802a3037a
   5cc4835080bc5321 dfbbb88c02c9c80a c62878a190905b6b 923a55d65eed291f
   c0a1c065bd095d1a 7cc1d86f463a1830 738177859f3210f6 c08ee4dcb2d08591

update 5
a: 9b2239a28cc5a4e3 80c1d0bc18b29e62 4b4f363e4a322d0e
b: b597b8f2964ec608 03ab81c48a1c1600 a581b5f011a0627d 62cfbead646814ad
   9bf2e26cc53cd63d 212ea54c36a11ccb 6059f0d6c0a0a4cd 901fb8d8b3eb5d35
   ddbd859802a3037a 5cc4835080bc5321 9981a0bc7e081065 c62878a190905b6b
   923a55d65eed291f c0a1c065bd095d1a 7cc1d86f463a1830 738177859f3210f6

update 6
a: 80c1d0bc18b29e62 dfe2225186dfbca3 1277ae469887f627
b: e8a34e2713f7b415 b597b8f2964ec608 03ab81c48a1c1600 a581b5f011a0627d
   f2d00675d7834998 9bf2e26cc53cd63d 212ea54c36a11ccb 6059f0d6c0a0a4cd
   901fb8d8b3eb5d35 ddbd859802a3037a e1cdde4a401d9344 9981a0bc7e081065
   c62878a190905b6b 923a55d65eed291f c0a1c065bd095d1a 7cc1d86f463a1830

update 7
a: dfe2225186dfbca3 ca86cdc9d2bf2007 7a5c92de7be1811f
b: fc0008d35e888652 e8a34e2713f7b415 b597b8f2964ec608 03ab81c48a1c1600
   c5d84526d100c6b0 f2d00675d7834998 9bf2e26cc53cd63d 212ea54c36a11ccb
   6059f0d6c0a0a4cd 901fb8d8b3eb5d35 8350ac87909956ac e1cdde4a401d9344
   9981a0bc7e081065 c62878a190905b6b 923a55d65eed291f c0a1c065bd095d1a

update 8
a: ca86cdc9d2bf2007 0870fbf8e065b266 067f3c2be88481d4
b: 1f43e2343bd6e1b9 fc0008d35e888652 e8a34e2713f7b415 b597b8f2964ec608
   22852488bcbd0acb c5d84526d100c6b0 f2d00675d7834998 9bf2e26cc53cd63d
   212ea54c36a11ccb 6059f0d6c0a0a4cd 008fe3b375c32594 8350ac87909956ac
   e1cdde4a401d9344 9981a0bc7e081065 c62878a190905b6b 923a55d65eed291f

update 9
a: 0870fbf8e065b266 73b145404394710d d756724ed3994273
b: 58bc981f8c520918 1f43e2343bd6e1b9 fc0008d35e888652 e8a34e2713f7b415
   2e655a9e53721035 22852488bcbd0acb c5d84526d100c6b0 f2d00675d7834998
   9bf2e26cc53cd63d 212ea54c36a11ccb 1e51e0b359210471 008fe3b375c32594
   8350ac87909956ac e1cdde4a401d9344 9981a0bc7e081065 c62878a190905b6b

update 10
a: 73b145404394710d 82d164adcac96d62 0607785b7d152b8b
b: ce58835970f5e90d 58bc981f8c520918 1f43e2343bd6e1b9 fc0008d35e888652
   1a734852c474fd8d 2e655a9e53721035 22852488bcbd0acb c5d84526d100c6b0
   f2d00675d7834998 9bf2e26cc53cd63d 61333608d76cc281 1e51e0b359210471
   008fe3b375c32594 8350ac87909956ac e1cdde4a401d9344 9981a0bc7e081065

update 11
a: 82d164adcac96d62 c14072735c68e7e9 f61c61bbde49ed28
b: ea30e5fc3d9c6168 ce58835970f5e90d 58bc981f8c520918 1f43e2343bd6e1b9
   39d84df58f8840e2 1a734852c474fd8d 2e655a9e53721035 22852488bcbd0acb
   c5d84526d100c6b0 f2d00675d7834998 0b6bb4c0466c7aba 61333608d76cc281
   1e51e0b359210471 008fe3b375c32594 8350ac87909956ac e1cdde4a401d9344

update 12
a: c14072735c68e7e9 ce0bee4623950852 af052447a7444e65
b: 631cbae78ad4fe26 ea30e5fc3d9c6168 ce58835970f5e90d 58bc981f8c520918
   3dc6c6bc876beb72 39d84df58f8840e2 1a734852c474fd8d 2e655a9e53721035
   22852488bcbd0acb c5d84526d100c6b0 871323e1d70caa2b 0b6bb4c0466c7aba
   61333608d76cc281 1e51e0b359210471 008fe3b375c32594 8350ac87909956ac

update 13
```

```
a: ce0bee4623950852 f6c22506fc93fb5a 9eb296971244bcb3
b: 4210def4ccf1b145 631cbae78ad4fe26 ea30e5fc3d9c6168 ce58835970f5e90d
   76d9c281df20192d 3dc6c6bc876beb72 39d84df58f8840e2 1a734852c474fd8d
   2e655a9e53721035 22852488bcbd0acb 9cf94157cf512603 871323e1d70caa2b
   0b6bb4c0466c7aba 61333608d76cc281 1e51e0b359210471 008fe3b375c32594

update 14
a: f6c22506fc93fb5a b36f504b7eb67fe6 a66ba7dd058722d3
b: ce840df556562dc6 4210def4ccf1b145 631cbae78ad4fe26 ea30e5fc3d9c6168
   d42bcb0bb4811480 76d9c281df20192d 3dc6c6bc876beb72 39d84df58f8840e2
   1a734852c474fd8d 2e655a9e53721035 f5e9e609dd8e3cc3 9cf94157cf512603
   871323e1d70caa2b 0b6bb4c0466c7aba 61333608d76cc281 1e51e0b359210471

update 15
a: b36f504b7eb67fe6 0ce5a4d1a0cbc0f7 bd0c30563f8ee4f7
b: e893c5b5a5b2ff2b ce840df556562dc6 4210def4ccf1b145 631cbae78ad4fe26
   d3e8a809b214218a d42bcb0bb4811480 76d9c281df20192d 3dc6c6bc876beb72
   39d84df58f8840e2 1a734852c474fd8d 680920245819a4f5 f5e9e609dd8e3cc3
   9cf94157cf512603 871323e1d70caa2b 0b6bb4c0466c7aba 61333608d76cc281

update 16
a: 0ce5a4d1a0cbc0f7 316993816117e50f bc62430614b79b71
b: d25c6643a9dabd67 e893c5b5a5b2ff2b ce840df556562dc6 4210def4ccf1b145
   5eda7c5b0dbf1554 d3e8a809b214218a d42bcb0bb4811480 76d9c281df20192d
   3dc6c6bc876beb72 39d84df58f8840e2 cd7fe2794367de6c 680920245819a4f5
   f5e9e609dd8e3cc3 9cf94157cf512603 871323e1d70caa2b 0b6bb4c0466c7aba

update 1
a: 316993816117e50f 4f7c747ce422e686 71a66681c35542de
b: 078e1011e6a7ba4d d25c6643a9dabd67 e893c5b5a5b2ff2b ce840df556562dc6
   34c91c7513d1a868 5eda7c5b0dbf1554 d3e8a809b214218a d42bcb0bb4811480
   76d9c281df20192d 3dc6c6bc876beb72 f6896bf6137101b5 cd7fe2794367de6c
   680920245819a4f5 f5e9e609dd8e3cc3 9cf94157cf512603 871323e1d70caa2b

update 2
a: 4f7c747ce422e686 0aeab5f525c1a62f 7aba5b4fb80e82d7
b: b67ab060b61b4f24 078e1011e6a7ba4d d25c6643a9dabd67 e893c5b5a5b2ff2b
   1aafc6fee2d73946 34c91c7513d1a868 5eda7c5b0dbf1554 d3e8a809b214218a
   d42bcb0bb4811480 76d9c281df20192d e048fa7f72820d7b f6896bf6137101b5
   cd7fe2794367de6c 680920245819a4f5 f5e9e609dd8e3cc3 9cf94157cf512603

update 3
a: 0aeab5f525c1a62f bd1a2938a57319c8 0b96982890b6e143
b: d385352b2b73c085 b67ab060b61b4f24 078e1011e6a7ba4d d25c6643a9dabd67
   3b7b6dbc17a6dea1 1aafc6fee2d73946 34c91c7513d1a868 5eda7c5b0dbf1554
   d3e8a809b214218a d42bcb0bb4811480 2ec06674b7293909 e048fa7f72820d7b
   f6896bf6137101b5 cd7fe2794367de6c 680920245819a4f5 f5e9e609dd8e3cc3

update 4
a: bd1a2938a57319c8 e4684a2bf28ff50d 4930b5d033157f46
b: ff0353fcf84f9aec d385352b2b73c085 b67ab060b61b4f24 078e1011e6a7ba4d
   8c861a18a465a833 3b7b6dbc17a6dea1 1aafc6fee2d73946 34c91c7513d1a868
   5eda7c5b0dbf1554 d3e8a809b214218a 974c156779fef6f9 2ec06674b7293909
   e048fa7f72820d7b f6896bf6137101b5 cd7fe2794367de6c 680920245819a4f5
```

C.2 128-bit key example for SNOW 2.0

C.2.1 Key, initialization vector, and keystream triplets

(IV_3, IV_2, IV_1, IV_0) = (0, 0, 0, 0), key = 80000000000000000000000000000000
Keystream output: 8D590AE9A74A7D056DC9CA74B72D1A4599B0A083FB45D13FCF9411BD9A503783.

(IV_3, IV_2, IV_1, IV_0) = (0, 0, 0, 0), key = AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Keystream output: E00982F525F02054214992D8706F2B20DA585E5B85E2746D09F22681B2749407.

(IV_3, IV_2, IV_1, IV_0) = (4,3,2,1), key = 80000000000000000000000000000000
Keystream output: D6403358E0354A6957F43FCE44B4B13FF78E24C246618A0767AC83C10BFC45F0.

(IV_3, IV_2, IV_1, IV_0) = (4,3,2,1), key = AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Keystream output: C355385DB31D6CBDF774AF5366C2E8774DEADAC7DC7229DFED171D7B.

C.2.2 Sample internal states

```
K = 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV= 00 00 00 04 00 00 00 03 00 00 00 02 00 00 00 01
Z = d6 40 33 58 e0 35 4a 69 57 f4 3f ce 44 b4 b1 3f f7 8e 24 c2 46 61 8a 07 67 ac 83 c1 0b fc 45 f0

Snow 2.0 Internal state at time -34
15:80000000 14:00000000 13:00000000 12:00000000 11:7fffffff 10:ffffffff 09:ffffffff 08:ffffffff
07:80000000 06:00000000 05:00000000 04:00000000 03:7fffffff 02:ffffffff 01:ffffffff 00:ffffffff

Snow 2.0 Internal state at time -33
15:80000001 14:00000000 13:00000000 12:00000002 11:7fffffff 10:fffffffc 09:fffffffb 08:ffffffff
07:80000000 06:00000000 05:00000000 04:00000000 03:7fffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:00000000 R2:00000000

Snow 2.0 Internal state at time -32
15:09dfef08 14:80000001 13:00000000 12:00000000 11:00000002 10:7fffffff 09:fffffffc 08:fffffffb
07:ffffffff 06:80000000 05:00000000 04:00000000 03:00000000 02:7fffffff 01:ffffffff 00:ffffffff
R1:00000000 R2:63636363

Snow 2.0 Internal state at time -31
15:e5f1b94c 14:09dfef08 13:80000001 12:00000000 11:00000000 10:00000002 09:7fffffff 08:fffffffc
07:fffffffb 06:ffffffff 05:80000000 04:00000000 03:00000000 02:00000000 01:7fffffff 00:ffffffff
R1:63636363 R2:63636363

Snow 2.0 Internal state at time -30
15:ea9a3527 14:e5f1b94c 13:09dfef08 12:80000001 11:00000000 10:00000000 09:00000002 08:7fffffff
07:fffffffc 06:fffffffb 05:ffffffff 04:80000000 03:00000000 02:00000000 01:00000000 00:7fffffff
R1:e3636363 R2:fbfbfbfb

Snow 2.0 Internal state at time -29
15:a69fa10d 14:ea9a3527 13:e5f1b94c 12:09dfef08 11:80000001 10:00000000 09:00000000 08:00000002
07:7fffffff 06:fffffffc 05:fffffffb 04:ffffffff 03:80000000 02:00000000 01:00000000 00:00000000
R1:fbfbfbfa R2:34de1111

Snow 2.0 Internal state at time -28
15:8ecaccdb 14:a69fa10d 13:ea9a3527 12:e5f1b94c 11:09dfef08 10:80000001 09:00000000 08:00000000
07:00000002 06:7fffffff 05:fffffffc 04:fffffffb 03:ffffffff 02:80000000 01:00000000 00:00000000
R1:34de110c R2:692d2d4b

Snow 2.0 Internal state at time -27
15:eaf4d48f 14:8ecaccdb 13:a69fa10d 12:ea9a3527 11:e5f1b94c 10:09dfef08 09:80000001 08:00000000
07:00000000 06:00000002 05:7fffffff 04:fffffffc 03:fffffffb 02:ffffffff 01:80000000 00:00000000
```

```
R1:692d2d47 R2:b66ede7f

Snow 2.0 Internal state at time -26
15:19805681 14:eaf4d48f 13:8ecaccdb 12:a69fa10d 11:ea9a3527 10:e5f1b94c 09:09dfef08 08:80000001
07:00000000 06:00000000 05:00000002 04:7fffffff 03:fffffffc 02:fffffffb 01:ffffffff 00:80000000
R1:366ede7e R2:12c38109

Snow 2.0 Internal state at time -25
15:e8688f55 14:19805681 13:eaf4d48f 12:8ecaccdb 11:a69fa10d 10:ea9a3527 09:e5f1b94c 08:09dfef08
07:80000001 06:00000000 05:00000000 04:00000002 03:7fffffff 02:fffffffc 01:fffffffb 00:ffffffff
R1:12c3810b R2:86c47640

Snow 2.0 Internal state at time -24
15:fa565ef1 14:e8688f55 13:19805681 12:eaf4d48f 11:8ecaccdb 10:a69fa10d 09:ea9a3527 08:e5f1b94c
07:09dfef08 06:80000001 05:00000000 04:00000000 03:00000002 02:7fffffff 01:fffffffc 00:fffffffb
R1:86c47640 R2:d63b88a5

Snow 2.0 Internal state at time -23
15:6c7a94aa 14:fa565ef1 13:e8688f55 12:19805681 11:eaf4d48f 10:8ecaccdb 09:a69fa10d 08:ea9a3527
07:e5f1b94c 06:09dfef08 05:80000001 04:00000000 03:00000000 02:00000002 01:7fffffff 00:fffffffc
R1:d63b88a5 R2:b7c51902

Snow 2.0 Internal state at time -22
15:5ced2805 14:6c7a94aa 13:fa565ef1 12:e8688f55 11:19805681 10:eaf4d48f 09:8ecaccdb 08:a69fa10d
07:ea9a3527 06:e5f1b94c 05:09dfef08 04:80000001 03:00000000 02:00000000 01:00000002 00:7fffffff
R1:37c51903 R2:db1c5e4f

Snow 2.0 Internal state at time -21
15:26ac1e81 14:5ced2805 13:6c7a94aa 12:fa565ef1 11:e8688f55 10:19805681 09:eaf4d48f 08:8ecaccdb
07:a69fa10d 06:ea9a3527 05:e5f1b94c 04:09dfef08 03:80000001 02:00000000 01:00000000 00:00000002
R1:e4fc4d57 R2:d04da3ad

Snow 2.0 Internal state at time -20
15:2e5cc1a4 14:26ac1e81 13:5ced2805 12:6c7a94aa 11:fa565ef1 10:e8688f55 09:19805681 08:eaf4d48f
07:8ecaccdb 06:a69fa10d 05:ea9a3527 04:e5f1b94c 03:09dfef08 02:80000001 01:00000000 00:00000000
R1:b63f5cf9 R2:6c782451

Snow 2.0 Internal state at time -19
15:2eb71be8 14:2e5cc1a4 13:26ac1e81 12:5ced2805 11:6c7a94aa 10:fa565ef1 09:e8688f55 08:19805681
07:eaf4d48f 06:8ecaccdb 05:a69fa10d 04:ea9a3527 03:e5f1b94c 02:09dfef08 01:80000001 00:00000000
R1:57125978 R2:13ebdccc

Snow 2.0 Internal state at time -18
15:dc33fa8c 14:2eb71be8 13:2e5cc1a4 12:26ac1e81 11:5ced2805 10:6c7a94aa 09:fa565ef1 08:e8688f55
07:19805681 06:eaf4d48f 05:8ecaccdb 04:a69fa10d 03:ea9a3527 02:e5f1b94c 01:09dfef08 00:80000001
R1:ba8b7dd9 R2:6b132ab7

Snow 2.0 Internal state at time -17
15:3007668a 14:dc33fa8c 13:2eb71be8 12:2e5cc1a4 11:26ac1e81 10:5ced2805 09:6c7a94aa 08:fa565ef1
07:e8688f55 06:19805681 05:eaf4d48f 04:8ecaccdb 03:a69fa10d 02:ea9a3527 01:e5f1b94c 00:09dfef08
R1:f9ddf792 R2:6eb763b9

Snow 2.0 Internal state at time -16
15:6fbbfcfb 14:3007668a 13:dc33fa8c 12:2eb71be8 11:2e5cc1a4 10:26ac1e81 09:5ced2805 08:6c7a94aa
07:fa565ef1 06:e8688f55 05:19805681 04:eaf4d48f 03:8ecaccdb 02:a69fa10d 01:ea9a3527 00:e5f1b94c
R1:59ac3848 R2:510e5e7e

Snow 2.0 Internal state at time -15
15:47128118 14:6fbbfcfb 13:3007668a 12:dc33fa8c 11:2eb71be8 10:2e5cc1a4 09:26ac1e81 08:5ced2805
```

```
07:6c7a94aa 06:fa565ef1 05:e8688f55 04:19805681 03:eaf4d48f 02:8ecaccdb 01:a69fa10d 00:ea9a3527
R1:6a8eb4ff R2:ed2a3ff7

Snow 2.0 Internal state at time -14
15:9dd8c346 14:47128118 13:6fbbfcfb 12:3007668a 11:dc33fa8c 10:2eb71be8 09:2e5cc1a4 08:26ac1e81
07:5ced2805 06:6c7a94aa 05:fa565ef1 04:e8688f55 03:19805681 02:eaf4d48f 01:8ecaccdb 00:a69fa10d
R1:d592cf4c R2:aaaf3ebb

Snow 2.0 Internal state at time -13
15:14c6e4b1 14:9dd8c346 13:47128118 12:6fbbfcfb 11:3007668a 10:dc33fa8c 09:2eb71be8 08:2e5cc1a4
07:26ac1e81 06:5ced2805 05:6c7a94aa 04:fa565ef1 03:e8688f55 02:19805681 01:eaf4d48f 00:8ecaccdb
R1:a5059dac R2:b838f49b

Snow 2.0 Internal state at time -12
15:2be1899a 14:14c6e4b1 13:9dd8c346 12:47128118 11:6fbbfcfb 10:3007668a 09:dc33fa8c 08:2eb71be8
07:2e5cc1a4 06:26ac1e81 05:5ced2805 04:6c7a94aa 03:fa565ef1 02:e8688f55 01:19805681 00:eaf4d48f
R1:24b38945 R2:911396b6

Snow 2.0 Internal state at time -11
15:09363669 14:2be1899a 13:14c6e4b1 12:9dd8c346 11:47128118 10:6fbbfcfb 09:3007668a 08:dc33fa8c
07:2eb71be8 06:2e5cc1a4 05:26ac1e81 04:5ced2805 03:6c7a94aa 02:fa565ef1 01:e8688f55 00:19805681
R1:ee00bebb R2:1449ba75

Snow 2.0 Internal state at time -10
15:9e6f24ce 14:09363669 13:2be1899a 12:14c6e4b1 11:9dd8c346 10:47128118 09:6fbbfcfb 08:3007668a
07:dc33fa8c 06:2eb71be8 05:2e5cc1a4 04:26ac1e81 03:5ced2805 02:6c7a94aa 01:fa565ef1 00:e8688f55
R1:3af5d8f6 R2:b8fa206d

Snow 2.0 Internal state at time -9
15:f87d0a5a 14:9e6f24ce 13:09363669 12:2be1899a 11:14c6e4b1 10:9dd8c346 09:47128118 08:6fbbfcfb
07:3007668a 06:dc33fa8c 05:2eb71be8 04:2e5cc1a4 03:26ac1e81 02:5ced2805 01:6c7a94aa 00:fa565ef1
R1:e756e211 R2:5a6f3141

Snow 2.0 Internal state at time -8
15:056b74c0 14:f87d0a5a 13:9e6f24ce 12:09363669 11:2be1899a 10:14c6e4b1 09:9dd8c346 08:47128118
07:6fbbfcfb 06:3007668a 05:dc33fa8c 04:2eb71be8 03:2e5cc1a4 02:26ac1e81 01:5ced2805 00:6c7a94aa
R1:89264d29 R2:87c4f589

Snow 2.0 Internal state at time -7
15:82d49257 14:056b74c0 13:f87d0a5a 12:9e6f24ce 11:09363669 10:2be1899a 09:14c6e4b1 08:9dd8c346
07:47128118 06:6fbbfcfb 05:3007668a 04:dc33fa8c 03:2eb71be8 02:2e5cc1a4 01:26ac1e81 00:5ced2805
R1:63f8f015 R2:b541dd3f

Snow 2.0 Internal state at time -6
15:4f549ab4 14:82d49257 13:056b74c0 12:f87d0a5a 11:9e6f24ce 10:09363669 09:2be1899a 08:14c6e4b1
07:9dd8c346 06:47128118 05:6fbbfcfb 04:3007668a 03:dc33fa8c 02:2eb71be8 01:2e5cc1a4 00:26ac1e81
R1:e54943c9 R2:cb416287

Snow 2.0 Internal state at time -5
15:01d936bb 14:4f549ab4 13:82d49257 12:056b74c0 11:f87d0a5a 10:9e6f24ce 09:09363669 08:2be1899a
07:14c6e4b1 06:9dd8c346 05:47128118 04:6fbbfcfb 03:3007668a 02:dc33fa8c 01:2eb71be8 00:2e5cc1a4
R1:3afd5f82 R2:f4c17d6d

Snow 2.0 Internal state at time -4
15:99c99eb5 14:01d936bb 13:4f549ab4 12:82d49257 11:056b74c0 10:f87d0a5a 09:9e6f24ce 08:09363669
07:2be1899a 06:14c6e4b1 05:9dd8c346 04:47128118 03:6fbbfcfb 02:3007668a 01:dc33fa8c 00:2eb71be8
R1:3bd3fe85 R2:b5efeab8
```

```
Snow 2.0 Internal state at time -3
15:0ea4e3f0 14:99c99eb5 13:01d936bb 12:4f549ab4 11:82d49257 10:056b74c0 09:f87d0a5a 08:9e6f24ce
07:09363669 06:2be1899a 05:14c6e4b1 04:9dd8c346 03:47128118 02:6fbbfcfb 01:3007668a 00:dc33fa8c
R1:53c8adfe R2:a0ddb267

Snow 2.0 Internal state at time -2
15:fa000bc8 14:0ea4e3f0 13:99c99eb5 12:01d936bb 11:4f549ab4 10:82d49257 09:056b74c0 08:f87d0a5a
07:9e6f24ce 06:09363669 05:2be1899a 04:14c6e4b1 03:9dd8c346 02:47128118 01:6fbbfcfb 00:3007668a
R1:b5a49718 R2:6ac944cc

Snow 2.0 Internal state at time -1
15:61dec1b8 14:fa000bc8 13:0ea4e3f0 12:99c99eb5 11:01d936bb 10:4f549ab4 09:82d49257 08:056b74c0
07:f87d0a5a 06:9e6f24ce 05:09363669 04:2be1899a 03:14c6e4b1 02:9dd8c346 01:47128118 00:6fbbfcfb
R1:96aace66 R2:9cd3a85e

Snow 2.0 Internal state at time 0
15:31f914d5 14:61dec1b8 13:fa000bc8 12:0ea4e3f0 11:99c99eb5 10:01d936bb 09:4f549ab4 08:82d49257
07:056b74c0 06:f87d0a5a 05:9e6f24ce 04:09363669 03:2be1899a 02:14c6e4b1 01:9dd8c346 00:47128118
R1:a609dec7 R2:495041dc

Snow 2.0 Internal state at time 1
15:9098ec10 14:31f914d5 13:61dec1b8 12:fa000bc8 11:0ea4e3f0 10:99c99eb5 09:01d936bb 08:4f549ab4
07:82d49257 06:056b74c0 05:f87d0a5a 04:9e6f24ce 03:09363669 02:2be1899a 01:14c6e4b1 00:9dd8c346
R1:e7bf66aa R2:05b5db95

Snow 2.0 Internal state at time 2
15:a5e7b806 14:9098ec10 13:31f914d5 12:61dec1b8 11:fa000bc8 10:0ea4e3f0 09:99c99eb5 08:01d936bb
07:4f549ab4 06:82d49257 05:056b74c0 04:f87d0a5a 03:9e6f24ce 02:09363669 01:2be1899a 00:14c6e4b1
R1:fe32e5ef R2:e728468a

Snow 2.0 Internal state at time 3
15:962fd59e 14:a5e7b806 13:9098ec10 12:31f914d5 11:61dec1b8 10:fa000bc8 09:0ea4e3f0 08:99c99eb5
07:01d936bb 06:4f549ab4 05:82d49257 04:056b74c0 03:f87d0a5a 02:9e6f24ce 01:09363669 00:2be1899a
R1:ec93bb4a R2:ed96a84d

Snow 2.0 Internal state at time 4
15:be037f87 14:962fd59e 13:a5e7b806 12:9098ec10 11:31f914d5 10:61dec1b8 09:fa000bc8 08:0ea4e3f0
07:99c99eb5 06:01d936bb 05:4f549ab4 04:82d49257 03:056b74c0 02:f87d0a5a 01:9e6f24ce 00:09363669
R1:706b3aa4 R2:d0d6a880

Snow 2.0 Internal state at time 5
15:3e9ee9ba 14:be037f87 13:962fd59e 12:a5e7b806 11:9098ec10 10:31f914d5 09:61dec1b8 08:fa000bc8
07:0ea4e3f0 06:99c99eb5 05:01d936bb 04:4f549ab4 03:82d49257 02:056b74c0 01:f87d0a5a 00:9e6f24ce
R1:202b4334 R2:86c48227

Snow 2.0 Internal state at time 6
15:a14a61e1 14:3e9ee9ba 13:be037f87 12:962fd59e 11:a5e7b806 10:9098ec10 09:31f914d5 08:61dec1b8
07:fa000bc8 06:0ea4e3f0 05:99c99eb5 04:01d936bb 03:4f549ab4 02:82d49257 01:056b74c0 00:f87d0a5a
R1:889db8e2 R2:b6399358

Snow 2.0 Internal state at time 7
15:cc852528 14:a14a61e1 13:3e9ee9ba 12:be037f87 11:962fd59e 10:a5e7b806 09:9098ec10 08:31f914d5
07:61dec1b8 06:fa000bc8 05:0ea4e3f0 04:99c99eb5 03:01d936bb 02:4f549ab4 01:82d49257 00:056b74c0
R1:5003320d R2:121f6605

Snow 2.0 Internal state at time 8
15:4b895ab7 14:cc852528 13:a14a61e1 12:3e9ee9ba 11:be037f87 10:962fd59e 09:a5e7b806 08:9098ec10
07:31f914d5 06:61dec1b8 05:fa000bc8 04:0ea4e3f0 03:99c99eb5 02:01d936bb 01:4f549ab4 00:82d49257
R1:20c449f5 R2:9cf74ff8
```

C.3 256-bit key example for SNOW 2.0

C.3.1 Key, initialization vector, and keystream triplets

$(IV_3, IV_2, IV_1, IV_0) = (0,0,0,0)$,
```
key = 8000000000000000000000000000000000000000000000000000000000000000
Keystream output: 0B5BCCE20323E28E0FC203809C66AB73CA35A680F2A5DD197E0C5C02287BE822.
```

$(IV_3, IV_2, IV_1, IV_0) = (0,0,0,0)$,
```
key = AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Keystream output: D9CC22FD861492D0AE6F43FB0F072012078C5AEEE479DE8CF0E555F458EED858.
```

$(IV_3, IV_2, IV_1, IV_0) = (4,3,2,1)$,
```
key = 8000000000000000000000000000000000000000000000000000000000000000
Keystream output: 7861080D5755E90B736F10916ED519B12C1A3A4255297FC2246AB7FA6C089526.
```

$(IV_3, IV_2, IV_1, IV_0) = (4,3,2,1)$,
```
key = AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Keystream output: 29261FCE5ED038201D6AFAF8B87E74FED49ECB10197EAC025D024EB45E0C7655.
```

C.3.2 Sample internal state

```
K =  80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV= 00 00 00 04 00 00 00 03 00 00 00 02 00 00 00 01
Z0= 78 61 08 0d 57 55 e9 0b 73 6f 10 91 6e d5 19 b1 2c 1a 3a 42 55 29 7f c2 24 6a b7 fa 6c 08 95 26

Snow 2.0 Internal state at time -34
15:80000000 14:00000000 13:00000000 12:00000000 11:00000000 10:00000000 09:00000000 08:00000000
07:7fffffff 06:ffffffff 05:ffffffff 04:ffffffff 03:ffffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:0804a9cc R2:00000001

Snow 2.0 Internal state at time -33
15:80000001 14:00000000 13:00000000 12:00000002 11:00000000 10:00000003 09:00000004 08:00000000
07:7fffffff 06:ffffffff 05:ffffffff 04:ffffffff 03:ffffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:00000000 R2:00000000

Snow 2.0 Internal state at time -32
15:bf53b515 14:80000001 13:00000000 12:00000000 11:00000002 10:00000000 09:00000003 08:00000004
07:00000000 06:7fffffff 05:ffffffff 04:ffffffff 03:ffffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:ffffffff R2:63636363

Snow 2.0 Internal state at time -31
15:d37de350 14:bf53b515 13:80000001 12:00000000 11:00000000 10:00000002 09:00000000 08:00000003
07:00000004 06:00000000 05:7fffffff 04:ffffffff 03:ffffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:63636362 R2:16161616

Snow 2.0 Internal state at time -30
15:1fa4e5b0 14:d37de350 13:bf53b515 12:80000001 11:00000000 10:00000000 09:00000002 08:00000000
07:00000003 06:00000004 05:00000000 04:7fffffff 03:ffffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:96161615 R2:08aaaa59

Snow 2.0 Internal state at time -29
15:8243e488 14:1fa4e5b0 13:d37de350 12:bf53b515 11:80000001 10:00000000 09:00000000 08:00000002
07:00000000 06:00000003 05:00000004 04:00000000 03:7fffffff 02:ffffffff 01:ffffffff 00:ffffffff
R1:08aaaa59 R2:d03b8eac

Snow 2.0 Internal state at time -28
15:7d09f594 14:8243e488 13:1fa4e5b0 12:d37de350 11:bf53b515 10:80000001 09:00000000 08:00000000
```

```
07:00000002 06:00000000 05:00000003 04:00000004 03:00000000 02:7fffffff 01:ffffffff 00:ffffffff
R1:d03b8eb0 R2:267457fe

Snow 2.0 Internal state at time -27
15:851e8381 14:7d09f594 13:8243e488 12:1fa4e5b0 11:d37de350 10:bf53b515 09:80000001 08:00000000
07:00000000 06:00000002 05:00000000 04:00000003 03:00000004 02:00000000 01:7fffffff 00:ffffffff
R1:26745801 R2:29b1986c

Snow 2.0 Internal state at time -26
15:cf3efe13 14:851e8381 13:7d09f594 12:8243e488 11:1fa4e5b0 10:d37de350 09:bf53b515 08:80000001
07:00000000 06:00000000 05:00000002 04:00000000 03:00000003 02:00000004 01:00000000 00:7fffffff
R1:29b1986c R2:892bf223

Snow 2.0 Internal state at time -25
15:7b69e0b3 14:cf3efe13 13:851e8381 12:7d09f594 11:8243e488 10:1fa4e5b0 09:d37de350 08:bf53b515
07:80000001 06:00000000 05:00000000 04:00000002 03:00000000 02:00000003 01:00000004 00:00000000
R1:892bf225 R2:2f693a07

Snow 2.0 Internal state at time -24
15:0a8b53d5 14:7b69e0b3 13:cf3efe13 12:851e8381 11:7d09f594 10:8243e488 09:1fa4e5b0 08:d37de350
07:bf53b515 06:80000001 05:00000000 04:00000000 03:00000002 02:00000000 01:00000003 00:00000004
R1:2f693a07 R2:6cbd99a8

Snow 2.0 Internal state at time -23
15:fd75ecf7 14:0a8b53d5 13:7b69e0b3 12:cf3efe13 11:851e8381 10:7d09f594 09:8243e488 08:1fa4e5b0
07:d37de350 06:bf53b515 05:80000001 04:00000000 03:00000000 02:00000002 01:00000000 00:00000003
R1:6cbd99a8 R2:0793dbe6

Snow 2.0 Internal state at time -22
15:94a70314 14:fd75ecf7 13:0a8b53d5 12:7b69e0b3 11:cf3efe13 10:851e8381 09:7d09f594 08:8243e488
07:1fa4e5b0 06:d37de350 05:bf53b515 04:80000001 03:00000000 02:00000000 01:00000002 00:00000000
R1:8793dbe7 R2:6928db9c

Snow 2.0 Internal state at time -21
15:743ca456 14:94a70314 13:fd75ecf7 12:0a8b53d5 11:7b69e0b3 10:cf3efe13 09:851e8381 08:7d09f594
07:8243e488 06:1fa4e5b0 05:d37de350 04:bf53b515 03:80000001 02:00000000 01:00000000 00:00000002
R1:287c90b1 R2:ecb79528

Snow 2.0 Internal state at time -20
15:c250e943 14:743ca456 13:94a70314 12:fd75ecf7 11:0a8b53d5 10:7b69e0b3 09:cf3efe13 08:851e8381
07:7d09f594 06:8243e488 05:1fa4e5b0 04:d37de350 03:bf53b515 02:80000001 01:00000000 00:00000000
R1:c0357878 R2:5bd40c0f

Snow 2.0 Internal state at time -19
15:4d848699 14:c250e943 13:743ca456 12:94a70314 11:fd75ecf7 10:0a8b53d5 09:7b69e0b3 08:cf3efe13
07:851e8381 06:7d09f594 05:8243e488 04:1fa4e5b0 03:d37de350 02:bf53b515 01:80000001 00:00000000
R1:7b78f1bf R2:9ae2c490

Snow 2.0 Internal state at time -18
15:9b3a221f 14:4d848699 13:c250e943 12:743ca456 11:94a70314 10:fd75ecf7 09:0a8b53d5 08:7b69e0b3
07:cf3efe13 06:851e8381 05:7d09f594 04:8243e488 03:1fa4e5b0 02:d37de350 01:bf53b515 00:80000001
R1:1d26a918 R2:47a9af75

Snow 2.0 Internal state at time -17
15:35d95fd1 14:9b3a221f 13:4d848699 12:c250e943 11:743ca456 10:94a70314 09:fd75ecf7 08:0a8b53d5
07:7b69e0b3 06:cf3efe13 05:851e8381 04:7d09f594 03:8243e488 02:1fa4e5b0 01:d37de350 00:bf53b515
R1:c4b3a509 R2:9b7cb67c

Snow 2.0 Internal state at time -16
```

```
15:e7492c66 14:35d95fd1 13:9b3a221f 12:4d848699 11:c250e943 10:743ca456 09:94a70314 08:fd75ecf7
07:0a8b53d5 06:7b69e0b3 05:cf3efe13 04:851e8381 03:7d09f594 02:8243e488 01:1fa4e5b0 00:d37de350
R1:209b39fd R2:50f9a679

Snow 2.0 Internal state at time -15
15:dce814e5 14:e7492c66 13:35d95fd1 12:9b3a221f 11:4d848699 10:c250e943 09:743ca456 08:94a70314
07:fd75ecf7 06:0a8b53d5 05:7b69e0b3 04:cf3efe13 03:851e8381 02:7d09f594 01:8243e488 00:1fa4e5b0
R1:2038a48c R2:8facfb3d

Snow 2.0 Internal state at time -14
15:e8e83f37 14:dce814e5 13:e7492c66 12:35d95fd1 11:9b3a221f 10:4d848699 09:c250e943 08:743ca456
07:94a70314 06:fd75ecf7 05:0a8b53d5 04:7b69e0b3 03:cf3efe13 02:851e8381 01:7d09f594 00:8243e488
R1:0b16dbf0 R2:97e148a3

Snow 2.0 Internal state at time -13
15:387810f3 14:e8e83f37 13:dce814e5 12:e7492c66 11:35d95fd1 10:9b3a221f 09:4d848699 08:c250e943
07:743ca456 06:94a70314 05:fd75ecf7 04:0a8b53d5 03:7b69e0b3 02:cf3efe13 01:851e8381 00:7d09f594
R1:a26c9c78 R2:27c607bf

Snow 2.0 Internal state at time -12
15:4bcddadb 14:387810f3 13:e8e83f37 12:dce814e5 11:e7492c66 10:35d95fd1 09:9b3a221f 08:4d848699
07:c250e943 06:743ca456 05:94a70314 04:fd75ecf7 03:0a8b53d5 02:7b69e0b3 01:cf3efe13 00:851e8381
R1:253bf4b6 R2:258cd170

Snow 2.0 Internal state at time -11
15:f05c5d45 14:4bcddadb 13:387810f3 12:e8e83f37 11:dce814e5 10:e7492c66 09:35d95fd1 08:9b3a221f
07:4d848699 06:c250e943 05:743ca456 04:94a70314 03:fd75ecf7 02:0a8b53d5 01:7b69e0b3 00:cf3efe13
R1:ba33d484 R2:f16f299b

Snow 2.0 Internal state at time -10
15:21e0b756 14:f05c5d45 13:4bcddadb 12:387810f3 11:e8e83f37 10:dce814e5 09:e7492c66 08:35d95fd1
07:9b3a221f 06:4d848699 05:c250e943 04:743ca456 03:94a70314 02:fd75ecf7 01:0a8b53d5 00:7b69e0b3
R1:65abcdf1 R2:998d6551

Snow 2.0 Internal state at time -9
15:00098c25 14:21e0b756 13:f05c5d45 12:4bcddadb 11:387810f3 10:e8e83f37 09:dce814e5 08:e7492c66
07:35d95fd1 06:9b3a221f 05:4d848699 04:c250e943 03:743ca456 02:94a70314 01:fd75ecf7 00:0a8b53d5
R1:5bde4e94 R2:bd0f2baa

Snow 2.0 Internal state at time -8
15:81a343e1 14:00098c25 13:21e0b756 12:f05c5d45 11:4bcddadb 10:387810f3 09:e8e83f37 08:dce814e5
07:e7492c66 06:35d95fd1 05:9b3a221f 04:4d848699 03:c250e943 02:743ca456 01:94a70314 00:fd75ecf7
R1:0a93b243 R2:267c6211

Snow 2.0 Internal state at time -7
15:7b933a92 14:81a343e1 13:00098c25 12:21e0b756 11:f05c5d45 10:4bcddadb 09:387810f3 08:e8e83f37
07:dce814e5 06:e7492c66 05:35d95fd1 04:9b3a221f 03:4d848699 02:c250e943 01:743ca456 00:94a70314
R1:c1b68430 R2:0b276cd6

Snow 2.0 Internal state at time -6
15:0339b827 14:7b933a92 13:81a343e1 12:00098c25 11:21e0b756 10:f05c5d45 09:4bcddadb 08:387810f3
07:e8e83f37 06:dce814e5 05:e7492c66 04:35d95fd1 03:9b3a221f 02:4d848699 01:c250e943 00:743ca456
R1:4100cca7 R2:ed4f10df

Snow 2.0 Internal state at time -5
15:e5fd1e4e 14:0339b827 13:7b933a92 12:81a343e1 11:00098c25 10:21e0b756 09:f05c5d45 08:4bcddadb
07:387810f3 06:e8e83f37 05:dce814e5 04:e7492c66 03:35d95fd1 02:9b3a221f 01:4d848699 00:c250e943
R1:d4983d45 R2:d14fec85
```

```
Snow 2.0 Internal state at time -4
15:991f7362 14:e5fd1e4e 13:0339b827 12:7b933a92 11:81a343e1 10:00098c25 09:21e0b756 08:f05c5d45
07:4bcddadb 06:387810f3 05:e8e83f37 04:dce814e5 03:e7492c66 02:35d95fd1 01:9b3a221f 00:4d848699
R1:ae38016a R2:431da2bb

Snow 2.0 Internal state at time -3
15:a0bca2f7 14:991f7362 13:e5fd1e4e 12:0339b827 11:7b933a92 10:81a343e1 09:00098c25 08:21e0b756
07:f05c5d45 06:4bcddadb 05:387810f3 04:e8e83f37 03:dce814e5 02:e7492c66 01:35d95fd1 00:9b3a221f
R1:2c05e1f2 R2:ae471763

Snow 2.0 Internal state at time -2
15:928b5256 14:a0bca2f7 13:991f7362 12:e5fd1e4e 11:0339b827 10:7b933a92 09:81a343e1 08:00098c25
07:21e0b756 06:f05c5d45 05:4bcddadb 04:387810f3 03:e8e83f37 02:dce814e5 01:e7492c66 00:35d95fd1
R1:e6bf2856 R2:f134ae00

Snow 2.0 Internal state at time -1
15:be366d56 14:928b5256 13:a0bca2f7 12:991f7362 11:e5fd1e4e 10:0339b827 09:7b933a92 08:81a343e1
07:00098c25 06:21e0b756 05:f05c5d45 04:4bcddadb 03:387810f3 02:e8e83f37 01:dce814e5 00:e7492c66
R1:3d0288db R2:f31c4fa3

Snow 2.0 Internal state at time 0
15:a5fadb9e 14:be366d56 13:928b5256 12:a0bca2f7 11:991f7362 10:e5fd1e4e 09:0339b827 08:7b933a92
07:81a343e1 06:00098c25 05:21e0b756 04:f05c5d45 03:4bcddadb 02:387810f3 01:e8e83f37 00:dce814e5
R1:e378ace8 R2:2dfa946e

Snow 2.0 Internal state at time 1
15:b70c6e50 14:a5fadb9e 13:be366d56 12:928b5256 11:a0bca2f7 10:991f7362 09:e5fd1e4e 08:0339b827
07:7b933a92 06:81a343e1 05:00098c25 04:21e0b756 03:f05c5d45 02:4bcddadb 01:387810f3 00:e8e83f37
R1:4fdb4bc4 R2:b95a6c28

Snow 2.0 Internal state at time 2
15:bce23d5c 14:b70c6e50 13:a5fadb9e 12:be366d56 11:928b5256 10:a0bca2f7 09:991f7362 08:e5fd1e4e
07:0339b827 06:7b933a92 05:81a343e1 04:00098c25 03:21e0b756 02:f05c5d45 01:4bcddadb 00:387810f3
R1:b963f84d R2:3d5135cb

Snow 2.0 Internal state at time 3
15:4eb9692d 14:bce23d5c 13:b70c6e50 12:a5fadb9e 11:be366d56 10:928b5256 09:a0bca2f7 08:991f7362
07:e5fd1e4e 06:0339b827 05:7b933a92 04:81a343e1 03:00098c25 02:21e0b756 01:f05c5d45 00:4bcddadb
R1:bef479ac R2:28b521b3

Snow 2.0 Internal state at time 4
15:96a599a9 14:4eb9692d 13:bce23d5c 12:b70c6e50 11:a5fadb9e 10:be366d56 09:928b5256 08:a0bca2f7
07:991f7362 06:e5fd1e4e 05:0339b827 04:7b933a92 03:81a343e1 02:00098c25 01:21e0b756 00:f05c5d45
R1:a4485c45 R2:e6ab92e9

Snow 2.0 Internal state at time 5
15:62ad427d 14:96a599a9 13:4eb9692d 12:bce23d5c 11:b70c6e50 10:a5fadb9e 09:be366d56 08:928b5256
07:a0bca2f7 06:991f7362 05:e5fd1e4e 04:0339b827 03:7b933a92 02:81a343e1 01:00098c25 00:21e0b756
R1:e9e54b10 R2:385b4519

Snow 2.0 Internal state at time 6
15:19397ef2 14:62ad427d 13:96a599a9 12:4eb9692d 11:bce23d5c 10:b70c6e50 09:a5fadb9e 08:be366d56
07:928b5256 06:a0bca2f7 05:991f7362 04:e5fd1e4e 03:0339b827 02:7b933a92 01:81a343e1 00:00098c25
R1:1e586367 R2:13f2d986

Snow 2.0 Internal state at time 7
15:5f8525f0 14:19397ef2 13:62ad427d 12:96a599a9 11:4eb9692d 10:bce23d5c 09:b70c6e50 08:a5fadb9e
07:be366d56 06:928b5256 05:a0bca2f7 04:991f7362 03:e5fd1e4e 02:0339b827 01:7b933a92 00:81a343e1
R1:ad124ce8 R2:e13ca41f
```

```
Snow 2.0 Internal state at time 8
15:fb9c0bcf 14:5f8525f0 13:19397ef2 12:62ad427d 11:96a599a9 10:4eb9692d 09:bce23d5c 08:b70c6e50
07:a5fadb9e 06:be366d56 05:928b5256 04:a0bca2f7 03:991f7362 02:e5fd1e4e 01:0339b827 00:7b933a92
R1:81f94716 R2:679f1c0a
```

C.4 Example for Rabbit

C.4.1 Introduction

All test vectors for Rabbit are given in the little-endian notation, i.e., for multi-byte numbers, the most significant bytes are stored at the highest memory addresses.

C.4.2 Key, initialization vector, and keystream triplets

```
K = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV= 00 00 00 00 00 00 00 00
Z = ED B7 05 67 37 5D CD 7C D8 95 54 F8 5E 27 A7 C6 8D 4A DC 70 32 29 8F 7B D4 EF F5 04 AC A6 29 5F
    66 8F BF 47 8A DB 2B E5 1E 6C DE 29 2B 82 DE 2A B4 8D 2A C6 56 59 79 22 0E C9 09 A7 E7 57 60 98

K = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
IV= 00 01 02 03 04 05 06 07
Z = 98 71 C7 BA 4E A3 08 07 CD AA 49 64 66 39 2D 2F 4A FF 43 55 EF 90 69 56 10 9B 96 65 97 8D AC ED
    9B 7C 6F 7F C8 2C 67 D2 73 22 CB DE 9D B0 16 45 8C 38 2C 9C 7D 30 44 E6 52 0B B9 2A 13 53 C0 FF

K = 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
IV= 00 00 00 00 00 00 00 00
Z = A8 F7 E6 9B 69 40 A7 8D 13 6A 5C 15 4A 15 79 52 A6 E4 23 58 59 E3 02 20 EA 68 64 36 BB 38 EF 53
    9C 29 40 55 6B 09 EC D7 FE A2 B0 AC 83 07 F1 69 62 65 A3 D6 44 28 1C 39 C9 CD 5E 1E 2F 9B E4 D0

K = 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
IV= 00 01 02 03 04 05 06 07
Z = F2 89 19 DD A1 28 F8 F9 0A 30 34 6E 97 94 D2 B7 4C 69 A2 D9 91 37 27 BC 5A 30 18 E6 33 2A F7 F3
    BE 3A C3 EF B3 68 F4 3A 4C B8 58 67 B8 1C 91 F9 24 29 0C 81 6B 8B 57 88 98 C5 7F B4 C0 BA 05 BD
```

C.4.3 Sample internal states

```
K = 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
IV= 00 01 02 03 04 05 06 07
Z = F2 89 19 DD A1 28 F8 F9 0A 30 34 6E 97 94 D2 B7 4C 69 A2 D9 91 37 27 BC 5A 30 18 E6 33 2A F7 F3
    BE 3A C3 EF B3 68 F4 3A 4C B8 58 67 B8 1C 91 F9 24 29 0C 81 6B 8B 57 88 98 C5 7F B4 C0 BA 05 BD

After key expansion (Internal state S(-9))
x0:03020100 x1:0D0C0B0A x2:07060504 x3:01000F0E x4:0B0A0908 x5:05040302 x6:0F0E0D0C x7:09080706
c0:09080B0A c1:03020504 c2:0D0C0F0E c3:07060908 c4:01000302 c5:0B0A0D0C c6:05040706 c7:0F0E0100
carry:0

After key setup iteration 1 (Internal state S(-8))
x0:05783933 x1:162113C0 x2:B38F168E x3:F08A919E x4:7F2CDA94 x5:ACBEB878 x6:0D5257A9 x7:4FF46B46
c0:563CDE57 c1:D64F39D7 c2:41DF5C42 c3:543ADC55 c4:D44D37D5 c5:3FDD5A40 c6:5238DA53 c7:E25B35D3
carry:0

After key setup iteration 2 (Internal state S(-7))
x0:798C2CEC x1:CC05FFD4 x2:50D68324 x3:2C306745 x4:AD519559 x5:81595E7A x6:29A589E2 x7:15212B97
c0:A371B1A4 c1:A99C6EAA c2:76B2A977 c3:A16FAFA2 c4:A79A6CA8 c5:74B0A775 c6:9F6DADA0 c7:B5A86AA6
carry:1
```

```
After key setup iteration 3 (Internal state S(-6))
x0:CD328957 x1:66D5AB1F x2:0D115824 x3:FCCEB784 x4:12E900D7 x5:36A46997 x6:9F40C5BC x7:AB1C8A08
c0:F0A684F2 c1:7CE9A37D c2:AB85F6AC c3:EEA482EF c4:7AE7A17B c5:A983F4AA c6:ECA280ED c7:88F59F79
carry:1

After key setup iteration 4 (Internal state S(-5))
x0:A31515F8 x1:5DFD3AC6 x2:33CD6AD2 x3:4BD778E5 x4:89708269 x5:D93095C1 x6:5E495F60 x7:C197863A
c0:3DDB5840 c1:5036D851 c2:E05943E1 c3:3BD9563C c4:4E34D64F c5:DE5741DF c6:39D7543A c7:5C42D44D
carry:1

After counter modification / IV setup (Internal state S(-4))
x0:A31515F8 x1:5DFD3AC6 x2:33CD6AD2 x3:4BD778E5 x4:89708269 x5:D93095C1 x6:5E495F60 x7:C197863A
c0:B7A9DB29 c1:8E004E92 c2:B9161985 c3:FF4AD106 c4:EE23C2B7 c5:84AC781B c6:0D1C3BEC c7:1291ADA8
carry:1

After IV setup iteration 1 (Internal state S(-3))
x0:054A3F2F x1:BE444CDE x2:573425A4 x3:9347FAD1 x4:29036A2F x5:DD3C6B50 x6:12CC3803 x7:6F7847C0
c0:04DEAE77 c1:614D8366 c2:EDE966BA c3:4C7FA453 c4:C170F78B c5:B97FC550 c6:5A510F39 c7:E5DEE27B
carry:0

After IV setup iteration 2 (Internal state S(-2))
x0:0FDB9A3A x1:334807E8 x2:E66BCC98 x3:0FDA371C x4:9C3E3036 x5:7774E657 x6:C6FCBB4C x7:A8D1AC4F
c0:521381C4 c1:349AB839 c2:22BCB3EF c3:99B477A1 c4:94BE2C5E c5:EE531285 c6:A785E286 c7:B92C174E
carry:1

After IV setup iteration 3 (Internal state S(-1))
x0:1A2EF77E x1:FDEEE287 x2:A918F5A1 x3:D6414F76 x4:4848D473 x5:BCE9BD30 x6:3E524094 x7:16242C51
c0:9F485512 c1:07E7ED0C c2:57900124 c3:E6E94AEE c4:680B6131 c5:23265FBA c6:F4BAB5D4 c7:8C794C21
carry:1

After IV setup iteration 4 (Internal state S(0))
x0:987651C2 x1:FF5F0007 x2:5C48C79E x3:661B3E75 x4:49247B9A x5:3C7AA744 x6:4AEF3F40 x7:D117584E
c0:EC7D2860 c1:DB3521DF c2:8C634E58 c3:341E1E3B c4:3B589605 c5:57F9ACEF c6:41EF8921 c7:5FC680F5
carry:1

After keystream iteration 1 (Internal state S(1))
x0:2A158BE4 x1:D93EC5A4 x2:298B7C1B x3:01F4F70C x4:E241E934 x5:0216D073 x6:72769563 x7:54BA8C75
c0:39B1FBAE c1:AE8256B3 c2:C1369B8D c3:8152F188 c4:0EA5CAD8 c5:8CCCFA24 c6:8F245C6E c7:3313B5C8
carry:1
output F2 89 19 DD A1 28 F8 F9 0A 30 34 6E 97 94 D2 B7

After keystream iteration 2 (Internal state S(2))
x0:46EC0492 x1:A4B5D46E x2:7B374C9E x3:93249F4E x4:E93894EF x5:6DDEC710 x6:2799B917 x7:7B0F0F20
c0:86E6CEFC c1:81CF8B86 c2:F609E8C2 c3:CE87C4D5 c4:E1F2FFAB c5:C1A04758 c6:DC592FBB c7:0660EA9B
carry:1
output 4C 69 A2 D9 91 37 27 BC 5A 30 18 E6 33 2A F7 F3

After keystream iteration 3 (Internal state S(3))
x0:98C27422 x1:0D5B5EC2 x2:FEEC9F8D x3:423F7701 x4:E22AB517 x5:4E9CC418 x6:A7535E87 x7:F73E8572
c0:D41BA24A c1:551CC059 c2:2ADD35F7 c3:1BBC9823 c4:B540347F c5:F673948D c6:298E0308 c7:D9AE1F6F
carry:0
output BE 3A C3 EF B3 68 F4 3A 4C B8 58 67 B8 1C 91 F9

After keystream iteration 4 (Internal state S(4))
x0:3B844C36 x1:AF5CD78B x2:2619A0AC x3:774FBA88 x4:D16C6AC4 x5:6512AE4E x6:6A8ECD8F x7:2BC76513
c0:21507597 c1:2869F52D c2:5FB0832C c3:68F16B70 c4:888D6952 c5:2B46E1C2 c6:76C2D656 c7:ACFB5442
carry:1
output 24 29 0C 81 6B 8B 57 88 98 C5 7F B4 C0 BA 05 BD
```

C.5 Example for Decim^{v2}

C.5.1 Introduction to the Decim^{v2} example

The byte-values and binary decomposition of bytes follow the big-endian notation, i.e., for multi-byte numbers, the most significant bytes are stored at the lowest memory addresses. In particular, this holds for the key, *IV*, keystream, register and buffer byte- and binary values given below.

$K = K_{79} \dots K_0$

$IV = IV_{63} \dots IV_0$

$Z = Z_n \dots Z_0$

$a = a_{191} \dots a_0$

$b = b_{31} \dots b_0$

$T = T_2T_1T_0$

and, for instance, given the key

```
K = de aa 00 40 00 30 00 0f 08 80,
```

each registers are defined to:

$[K_{79} \dots K_{72}]=de,\ [K_{71} \dots K_{64}]=aa\ \dots\ [K_7 \dots K_0]=80,$

with bit-decomposition as follows:

$K_{79} \dots K_0$ = 11011110 10101010 00000000 01000000 00000000
          00110000 00000000 00001111 00001000 10000000

C.5.2 Key, initialization vector, and keystream triplets

```
K = 00 00 00 00 00 00 00 00 00 80
IV= 00 00 00 00 00 00 00 00
Z = 76 e3 89 be 1b fb ad d5 3c ce a0 fe 43 b8 c8 fb d3 92 b8 0b 52 94 60 f8

K = 00 00 00 00 00 00 00 00 00 00
IV= 00 00 00 00 00 00 00 80
Z = 4c ec bd b3 0e cd c9 c0 8b 41 8f 7f 28 ff 83 48 75 40 ff c5 cb 0a 33 da

K = 09 09 09 09 09 09 09 09 09 09
IV= 00 00 00 00 00 00 00 00
Z = 43 9b ba f8 a7 84 dc f9 e6 d2 90 1d 12 4d 43 09 22 33 f2 47 60 19 70 53

K = 09 08 07 06 05 04 03 02 01 00
IV= 00 00 00 00 00 00 00 00
Z = 52 b1 73 10 01 2a cd 3a d2 20 4f e2 b2 2a 5d 21 64 41 f6 3d d3 b4 43 6a

K = eb 98 45 f2 9f 4c f9 a6 53 00
IV= de 77 10 a9 42 db 74 0d
Z = 62 ff c9 cc 21 0e 07 ea 6e 50 f0 fb 1b 60 36 7f 88 a6 a5 27 9b 18 cb b8

K = fa a7 54 01 ae 5b 08 b5 62 0f
IV= f9 92 2b c4 5d f6 8f 28
Z = f0 af 66 52 2a 23 8b 29 63 37 8b 18 ec 1f 4c a8 27 91 3d 2c f0 ad 94 d9
```

C.5.3 Sample internal states

The binary equivalents of the internal states for key stages, namely at time -256, time -64, time 0 and time 193 are provided.

```
K = 00 00 00 00 00 00 00 00 00 80
IV= 00 00 00 00 00 00 00 00
Z = 76 e3 89 be 1b fb ad d5 3c ce a0 fe 43 b8 c8 fb d3 92 b8 0b 52 94 60 f8
```

For time -256 until -64 (executions of *InitNext* (S,LFSR)), internal state variables T, b and I have the following values:

```
T: 000
b: 00 00 00 00
I: 0
```

```
Decim v2 Internal State at time -256
a: ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 00 00 80

Decim v2 Binary Internal State at time -256 (Binary notation)
a: 11111111 11111111 11111111 11111111 00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000000 00000000 10000000 00000000 00000000
   00000000 00000000 00000000 00000000 00000000 00000000 00000000 10000000

Executions of InitNext(S,LFSR)

Decim v2 Internal State at time -255
a: 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 00 00 08

Decim v2 Internal State at time -254
a: 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -253
a: 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -252
a: 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -251
a: 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -250
a: 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00 00

Decim v2 Internal State at time -249
a: e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00

Decim v2 Internal State at time -248
a: 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00 00

Decim v2 Internal State at time -247
a: 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00

Decim v2 Internal State at time -246
a: 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00 00

Decim v2 Internal State at time -245
a: d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00 00

Decim v2 Internal State at time -244
a: 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00 00

Decim v2 Internal State at time -243
a: 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00 00

Decim v2 Internal State at time -242
a: d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00 00

Decim v2 Internal State at time -241
a: fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00 00
```

```
Decim v2 Internal State at time -240
a: df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00 00

Decim v2 Internal State at time -239
a: ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00 00

Decim v2 Internal State at time -238
a: fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80 00

Decim v2 Internal State at time -237
a: ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08 00

Decim v2 Internal State at time -236
a: ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00 80

Decim v2 Internal State at time -235
a: ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00 08

Decim v2 Internal State at time -234
a: 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -233
a: 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -232
a: 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00 00

Decim v2 Internal State at time -231
a: b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00 00

Decim v2 Internal State at time -230
a: 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00 00

Decim v2 Internal State at time -229
a: c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00 00

Decim v2 Internal State at time -228
a: bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00 00

Decim v2 Internal State at time -227
a: 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00 00

Decim v2 Internal State at time -226
a: d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00 00

Decim v2 Internal State at time -225
a: bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00 00

Decim v2 Internal State at time -224
a: 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00 00

Decim v2 Internal State at time -223
a: 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00 00

Decim v2 Internal State at time -222
a: b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00 00

Decim v2 Internal State at time -221
a: fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00 00

Decim v2 Internal State at time -220
a: 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00 00

Decim v2 Internal State at time -219
a: 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0 00

Decim v2 Internal State at time -218
a: 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff 00

Decim v2 Internal State at time -217
a: 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff f0

Decim v2 Internal State at time -216
a: f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff ff
```

```
Decim v2 Internal State at time -215
a: 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff ff

Decim v2 Internal State at time -214
a: e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff ff

Decim v2 Internal State at time -213
a: 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff ff

Decim v2 Internal State at time -212
a: f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff ff

Decim v2 Internal State at time -211
a: 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f ff

Decim v2 Internal State at time -210
a: 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42 ff

Decim v2 Internal State at time -209
a: 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84 2f

Decim v2 Internal State at time -208
a: 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98 42

Decim v2 Internal State at time -207
a: d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99 84

Decim v2 Internal State at time -206
a: 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89 98

Decim v2 Internal State at time -205
a: 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8 99

Decim v2 Internal State at time -204
a: 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e 89

Decim v2 Internal State at time -203
a: 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04 e8

Decim v2 Internal State at time -202
a: 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00 4e

Decim v2 Internal State at time -201
a: c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0 04

Decim v2 Internal State at time -200
a: 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d 00

Decim v2 Internal State at time -199
a: 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16 d0

Decim v2 Internal State at time -198
a: 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1 6d

Decim v2 Internal State at time -197
a: 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd 16

Decim v2 Internal State at time -196
a: 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df d1

Decim v2 Internal State at time -195
a: 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed fd

Decim v2 Internal State at time -194
a: c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe df

Decim v2 Internal State at time -193
a: bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef ed

Decim v2 Internal State at time -192
a: 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee fe

Decim v2 Internal State at time -191
a: c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce ef
```

```
Decim v2 Internal State at time -190
a: 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c ee

Decim v2 Internal State at time -189
a: 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60 ce

Decim v2 Internal State at time -188
a: 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06 0c

Decim v2 Internal State at time -187
a: a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0 60

Decim v2 Internal State at time -186
a: ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b 06

Decim v2 Internal State at time -185
a: fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5 b0

Decim v2 Internal State at time -184
a: 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc 5b

Decim v2 Internal State at time -183
a: 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b c5

Decim v2 Internal State at time -182
a: f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6 bc

Decim v2 Internal State at time -181
a: af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd 6b

Decim v2 Internal State at time -180
a: 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b d6

Decim v2 Internal State at time -179
a: 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38 bd

Decim v2 Internal State at time -178
a: b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3 8b

Decim v2 Internal State at time -177
a: 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb 38

Decim v2 Internal State at time -176
a: 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f b3

Decim v2 Internal State at time -175
a: e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84 fb

Decim v2 Internal State at time -174
a: 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18 4f

Decim v2 Internal State at time -173
a: d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21 84

Decim v2 Internal State at time -172
a: 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2 18

Decim v2 Internal State at time -171
a: f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f 21

Decim v2 Internal State at time -170
a: 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9 f2

Decim v2 Internal State at time -169
a: d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e 9f

Decim v2 Internal State at time -168
a: cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6 e9

Decim v2 Internal State at time -167
a: dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f 6e

Decim v2 Internal State at time -166
a: ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45 f6
```

```
Decim v2 Internal State at time -165
a: fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04 5f

Decim v2 Internal State at time -164
a: 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10 45

Decim v2 Internal State at time -163
a: 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1 04

Decim v2 Internal State at time -162
a: 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d 10

Decim v2 Internal State at time -161
a: d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31 d1

Decim v2 Internal State at time -160
a: 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13 1d

Decim v2 Internal State at time -159
a: 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81 31

Decim v2 Internal State at time -158
a: 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88 13

Decim v2 Internal State at time -157
a: d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8 81

Decim v2 Internal State at time -156
a: ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c 88

Decim v2 Internal State at time -155
a: 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77 c8

Decim v2 Internal State at time -154
a: 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17 7c

Decim v2 Internal State at time -153
a: b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91 77

Decim v2 Internal State at time -152
a: 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49 17

Decim v2 Internal State at time -151
a: 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44 91

Decim v2 Internal State at time -150
a: e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4 49

Decim v2 Internal State at time -149
a: fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc 44

Decim v2 Internal State at time -148
a: cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b c4

Decim v2 Internal State at time -147
a: 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5 bc

Decim v2 Internal State at time -146
a: 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c 5b

Decim v2 Internal State at time -145
a: 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25 c5

Decim v2 Internal State at time -144
a: 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92 5c

Decim v2 Internal State at time -143
a: d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9 25

Decim v2 Internal State at time -142
a: 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca 92

Decim v2 Internal State at time -141
a: b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc a9
```

```
Decim v2 Internal State at time -140
a: 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f ca

Decim v2 Internal State at time -139
a: f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89 fc

Decim v2 Internal State at time -138
a: 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8 9f

Decim v2 Internal State at time -137
a: 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af 89

Decim v2 Internal State at time -136
a: 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a f8

Decim v2 Internal State at time -135
a: 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46 af

Decim v2 Internal State at time -134
a: e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4 6a

Decim v2 Internal State at time -133
a: 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b 46

Decim v2 Internal State at time -132
a: f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76 b4

Decim v2 Internal State at time -131
a: 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7 6b

Decim v2 Internal State at time -130
a: 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e 76

Decim v2 Internal State at time -129
a: c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2 e7

Decim v2 Internal State at time -128
a: 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d 2e

Decim v2 Internal State at time -127
a: 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8 d2

Decim v2 Internal State at time -126
a: 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f 8d

Decim v2 Internal State at time -125
a: 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6 f8

Decim v2 Internal State at time -124
a: 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd 6f

Decim v2 Internal State at time -123
a: 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc d6

Decim v2 Internal State at time -122
a: 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed cd

Decim v2 Internal State at time -121
a: 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe dc

Decim v2 Internal State at time -120
a: f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f ed

Decim v2 Internal State at time -119
a: 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21 fe

Decim v2 Internal State at time -118
a: d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22 1f

Decim v2 Internal State at time -117
a: 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2 21

Decim v2 Internal State at time -116
a: 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d 22
```

```
Decim v2 Internal State at time -115
a: 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53 d2

Decim v2 Internal State at time -114
a: 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45 3d

Decim v2 Internal State at time -113
a: 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4 53

Decim v2 Internal State at time -112
a: 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed 45

Decim v2 Internal State at time -111
a: 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e d4

Decim v2 Internal State at time -110
a: b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23 ed

Decim v2 Internal State at time -109
a: 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2 3e

Decim v2 Internal State at time -108
a: 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b 23

Decim v2 Internal State at time -107
a: 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91 b2

Decim v2 Internal State at time -106
a: a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9 1b

Decim v2 Internal State at time -105
a: ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe 91

Decim v2 Internal State at time -104
a: bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf e9

Decim v2 Internal State at time -103
a: 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c fe

Decim v2 Internal State at time -102
a: 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64 cf

Decim v2 Internal State at time -101
a: 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46 4c

Decim v2 Internal State at time -100
a: a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94 64

Decim v2 Internal State at time -99
a: 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9 46

Decim v2 Internal State at time -98
a: 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d 94

Decim v2 Internal State at time -97
a: 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6 d9

Decim v2 Internal State at time -96
a: d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b 6d

Decim v2 Internal State at time -95
a: 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9 b6

Decim v2 Internal State at time -94
a: 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f 9b

Decim v2 Internal State at time -93
a: 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51 f9

Decim v2 Internal State at time -92
a: 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85 1f

Decim v2 Internal State at time -91
a: 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18 51
```

```
Decim v2 Internal State at time -90
a: a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1 85

Decim v2 Internal State at time -89
a: 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e 18

Decim v2 Internal State at time -88
a: a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6 e1

Decim v2 Internal State at time -87
a: da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f 6e

Decim v2 Internal State at time -86
a: 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39 f6

Decim v2 Internal State at time -85
a: d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3 9f

Decim v2 Internal State at time -84
a: ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c 39

Decim v2 Internal State at time -83
a: 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38 c3

Decim v2 Internal State at time -82
a: 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53 8c

Decim v2 Internal State at time -81
a: 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55 38

Decim v2 Internal State at time -80
a: a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75 53

Decim v2 Internal State at time -79
a: 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77 55

Decim v2 Internal State at time -78
a: 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07 75

Decim v2 Internal State at time -77
a: c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10 77

Decim v2 Internal State at time -76
a: 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1 07

Decim v2 Internal State at time -75
a: 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f 10

Decim v2 Internal State at time -74
a: 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8 f1

Decim v2 Internal State at time -73
a: 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d 8f

Decim v2 Internal State at time -72
a: 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69 d8

Decim v2 Internal State at time -71
a: 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86 9d

Decim v2 Internal State at time -70
a: 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48 69

Decim v2 Internal State at time -69
a: 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64 86

Decim v2 Internal State at time -68
a: f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06 48

Decim v2 Internal State at time -67
a: 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60 64

Decim v2 Internal State at time -66
a: 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6 06
```

```
Decim v2 Internal State at time -65
a: 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b 60

Decim v2 Internal State at time -64
a: 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71 b6

Decim v2 Internal State at time -64 (Binary notation)
a: 00111000 00100011 11111001 01010100 10010100 01100011 01011000 11010001
   01101001 01010111 10101101 10001101 10100101 10100001 01001001 10100111
   10000111 01001100 10100011 01010101 10111011 10100100 01110001 10110110
T: 000         b: 00000000 00000000 00000000 00000000         I: 0
```

Executions of *InitNext*(S,BUFF)

```
Decim v2 Internal State at time -63
a: 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47 1b
T: 111         b: 00 00 00 00          I: 1

Decim v2 Internal State at time -62
a: a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4 71
T: 111         b: 00 00 00 00          I: 2

Decim v2 Internal State at time -61
a: 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba 47
T: 101         b: 00 00 00 00          I: 3

Decim v2 Internal State at time -60
a: 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb a4
T: 101         b: 00 00 00 08          I: 4

Decim v2 Internal State at time -59
a: b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b ba
T: 101         b: 00 00 00 18          I: 5

Decim v2 Internal State at time -58
a: 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55 bb
T: 001         b: 00 00 00 38          I: 6

Decim v2 Internal State at time -57
a: 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35 5b
T: 101         b: 00 00 00 78          I: 7

Decim v2 Internal State at time -56
a: 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3 55
T: 001         b: 00 00 00 f8          I: 9

Decim v2 Internal State at time -55
a: 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a 35
T: 001         b: 00 00 00 f8          I: 11

Decim v2 Internal State at time -54
a: 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58 a3
T: 010         b: 00 00 00 f8          I: 13

Decim v2 Internal State at time -53
a: a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15 8a
T: 010         b: 00 00 60 f8          I: 15

Decim v2 Internal State at time -52
a: 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1 58
T: 011         b: 00 00 60 f8          I: 16

Decim v2 Internal State at time -51
a: 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d 15
T: 011         b: 00 00 60 f8          I: 17

Decim v2 Internal State at time -50
a: d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69 d1
T: 010         b: 00 00 60 f8          I: 18

Decim v2 Internal State at time -49
a: 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96 9d
T: 001         b: 00 04 60 f8          I: 19

Decim v2 Internal State at time -48
a: 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49 69
```

```
T: 001        b: 00 14 60 f8        I: 21

Decim v2 Internal State at time -47
a: 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14 96
T: 001        b: 00 14 60 f8        I: 23

Decim v2 Internal State at time -46
a: 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1 49
T: 000        b: 00 94 60 f8        I: 25

Decim v2 Internal State at time -45
a: 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a 14
T: 000        b: 02 94 60 f8        I: 27

Decim v2 Internal State at time -44
a: e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5 a1
T: 010        b: 12 94 60 f8        I: 29

Decim v2 Internal State at time -43
a: 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da 5a
T: 010        b: 12 94 60 f8        I: 30

Decim v2 Internal State at time -42
a: 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5
T: 000        b: 52 94 60 f8        I: 32

Decim v2 Internal State at time 0
a: 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d a5
T: 000        b: 52 94 60 f8        I: 32

Decim v2 Internal State at time 0 (Binary notation)
a: 00000111 11100100 01010110 01000010 11011001 00111010 01001100 01100011
   10010100 00000111 10100010 00111000 00100011 11111001 01010100 00111011
   01010011 00100000 10000111 10100111 01010111 10101101 10001101 10100101
T: 000        b: 01010010 10010100 01100000 11111000        I: 32
```

Executions of *Next*(S)

```
Decim v2 Internal State at time 1
a: a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8 da
T: 101   b: a9 4a 30 7c   I: 32

Decim v2 Internal State at time 2
a: 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad 8d
T: 111   b: d4 a5 18 3e   I: 32

Decim v2 Internal State at time 3
a: b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a d8
T: 010   b: 6a 52 8c 1f   I: 32

Decim v2 Internal State at time 4
a: bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57 ad
T: 111   b: b5 29 46 0f   I: 32

Decim v2 Internal State at time 5
a: eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75 7a
T: 111   b: 5a 94 a3 07   I: 32

Decim v2 Internal State at time 6
a: ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7 57
T: 111   b: 2d 4a 51 83   I: 31

Decim v2 Internal State at time 7
a: fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a 75
T: 110   b: 16 a5 28 c1   I: 32

Decim v2 Internal State at time 8
a: 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87 a7
T: 111   b: 0b 52 94 60   I: 32

Decim v2 Internal State at time 9
a: f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8 7a
T: 010   b: 05 a9 4a 30   I: 32

Decim v2 Internal State at time 10
a: 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c 87
```

```
T: 010   b: 02 d4 a5 18   I: 32

Decim v2 Internal State at time 11
a: f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34 c8
T: 101   b: 01 6a 52 8c   I: 32

Decim v2 Internal State at time 12
a: 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63 4c
T: 100   b: 80 b5 29 46   I: 32

Decim v2 Internal State at time 13
a: 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46 34
T: 101   b: c0 5a 94 a3   I: 31

Decim v2 Internal State at time 14
a: 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94 63
T: 110   b: e0 2d 4a 51   I: 31

Decim v2 Internal State at time 15
a: 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49 46
T: 110   b: 70 16 a5 28   I: 32

Decim v2 Internal State at time 16
a: f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54 94
T: 000   b: b8 0b 52 94   I: 32

Decim v2 Internal State at time 17
a: 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95 49
T: 111   b: dc 05 a9 4a   I: 31

Decim v2 Internal State at time 18
a: 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9 54
T: 100   b: ae 02 d4 a5   I: 32

Decim v2 Internal State at time 19
a: 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f 95
T: 010   b: 57 01 6a 52   I: 32

Decim v2 Internal State at time 20
a: b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23 f9
T: 101   b: 2b 80 b5 29   I: 32

Decim v2 Internal State at time 21
a: eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82 3f
T: 101   b: 15 c0 5a 94   I: 31

Decim v2 Internal State at time 22
a: be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38 23
T: 110   b: 4a e0 2d 4a   I: 31

Decim v2 Internal State at time 23
a: 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23 82
T: 000   b: 25 70 16 a5   I: 32

Decim v2 Internal State at time 24
a: 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2 38
T: 100   b: 92 b8 0b 52   I: 32

Decim v2 Internal State at time 25
a: 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a 23
T: 101   b: c9 5c 05 a9   I: 32

Decim v2 Internal State at time 26
a: 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07 a2
T: 010   b: e4 ae 02 d4   I: 32

Decim v2 Internal State at time 27
a: 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0 7a
T: 111   b: 72 57 01 6a   I: 32

Decim v2 Internal State at time 28
a: b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b 07
T: 100   b: 39 2b 80 b5   I: 32

Decim v2 Internal State at time 29
a: 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33 b0
```

```
T: 101   b: 1c 95 c0 5a   I: 31

Decim v2 Internal State at time 30
a: d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53 3b
T: 000   b: 4e 4a e0 2d   I: 32

Decim v2 Internal State at time 31
a: bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05 33
T: 010   b: a7 25 70 16   I: 32

Decim v2 Internal State at time 32
a: 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20 53
T: 110   b: d3 92 b8 0b   I: 32

Decim v2 Internal State at time 33
a: c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2 05
T: 110   b: e9 c9 5c 05   I: 32

Decim v2 Internal State at time 34
a: 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a 20
T: 111   b: f4 e4 ae 02   I: 32

Decim v2 Internal State at time 35
a: c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93 a2
T: 101   b: 7a 72 57 01   I: 32

Decim v2 Internal State at time 36
a: dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9 3a
T: 110   b: bd 39 2b 80   I: 32

Decim v2 Internal State at time 37
a: fd c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d 93
T: 110   b: de 9c 95 c0   I: 32

Decim v2 Internal State at time 38
a: af dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42 d9
T: 110   b: ef 4e 4a e0   I: 32

Decim v2 Internal State at time 39
a: 2a fd c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64 2d
T: 010   b: f7 a7 25 70   I: 32

Decim v2 Internal State at time 40
a: e2 af dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56 42
T: 000   b: fb d3 92 b8   I: 32

Decim v2 Internal State at time 41
a: fe 2a fd c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45 64
T: 111   b: 7d e9 c9 5c   I: 32

Decim v2 Internal State at time 42
a: 9f e2 af dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4 56
T: 000   b: 3e f4 e4 ae   I: 32

Decim v2 Internal State at time 43
a: 59 fe 2a fd c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e 45
T: 010   b: 1f 7a 72 57   I: 32

Decim v2 Internal State at time 44
a: 35 9f e2 af dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07 e4
T: 000   b: 8f bd 39 2b   I: 32

Decim v2 Internal State at time 45
a: d3 59 fe 2a fd c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0 7e
T: 111   b: 47 de 9c 95   I: 32

Decim v2 Internal State at time 46
a: 1d 35 9f e2 af dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a 07
T: 000   b: 23 ef 4e 4a   I: 32

Decim v2 Internal State at time 47
a: 81 d3 59 fe 2a fd c3 c4 bd 6b 52 43 9b eb 74 8f 53 76 f3 f3 fc eb b0 a0
T: 101   b: 11 f7 a7 25   I: 31

Decim v2 Internal State at time 48
a: f8 1d 35 9f e2 af dc 3c 4b d6 b5 24 39 be b7 48 f5 37 6f 3f 3f ce bb 0a
```