

Australian/New Zealand Standard™

**Information technology—Security
techniques—Encryption algorithms**

Part 4: Stream ciphers



AS/NZS ISO/IEC 18033.4:2006

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-012, Information Systems, Security and Identification. It was approved on behalf of the Council of Standards Australia on 31 March 2006 and on behalf of the Council of Standards New Zealand on 16 June 2006. This Standard was published on 10 July 2006.

The following are represented on Committee IT-012:

Attorney General's Department
Australia Post
Australian Association of Permanent Building Societies
Australian Bankers Association
Australian Chamber of Commerce and Industry
Australian Electrical and Electronic Manufacturers Association
Australian Information Industry Association
Certification Forum of Australia
Consumers' Federation of Australia
Department of Defence (Australia)
Department of Social Welfare New Zealand
Government Communications Security Bureau, New Zealand
Internet Industry Association
NSW Police Service
New Zealand Defence Force
Reserve Bank of Australia

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.standards.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

Alternatively, both organizations publish an annual printed Catalogue with full details of all current Standards. For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

This Standard was issued in draft form for comment as DR 06035.

Australian/New Zealand Standard™

Information technology—Security techniques—Encryption algorithms

Part 4: Stream ciphers

First published as AS/NZS ISO/IEC 18033.4:2006.

COPYRIGHT

© Standards Australia/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Jointly published by Standards Australia, GPO Box 476, Sydney, NSW 2001 and Standards New Zealand, Private Bag 2439, Wellington 6020

ISBN 0 7337 7594 2

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-012, Information Systems, Security and Identification.

This Standard is identical with, and has been reproduced from ISO/IEC 18033-4:2005, *Information technology—Security techniques—Encryption algorithms—Part 4: Stream ciphers*.

The objective of this Standard is to provide the Information Security programming and development community with clear guidance to the selection and implementation of appropriate encryption algorithms.

This Standard is Part 4 of AS/NZS ISO/IEC 18033, *Information technology—Security techniques—Encryption Algorithms*, which when complete will consist of the following:

Part 1: General

Part 2: Asymmetric ciphers

Part 3: Block ciphers

Part 4: Stream ciphers (this Standard)

The terms ‘normative’ and ‘informative’ are used to define the application of the annex to which they apply. A normative annex is an integral part of a standard, whereas an informative annex is only for information and guidance.

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this part of ISO/IEC 18033’ should read ‘this Australian/New Zealand Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

<i>Reference to International Standard</i>		<i>Australian/New Zealand Standard</i>	
ISO/IEC		AS/NZS	
18033	Information technology—Security techniques—Encryption algorithms	18033	Information technology—Security techniques—Encryption algorithms
18033-1	Part 1: General	18033.1	Part 1: General
18033-3	Part 3: Block ciphers	18033.3	Part 3: Block ciphers

CONTENTS

Page

1	Scope	1
2	Normative references	1
3	Terms and definitions	1
4	Symbols and abbreviated terms	4
4.1	Left-truncation of bits	5
4.2	Shift operation	6
4.3	Variable $l(k)$	6
5	General models for stream ciphers	6
5.1	Keystream generators	6
5.1.1	Synchronous keystream generators	6
5.1.2	Self-synchronizing keystream generators	6
5.2	Output functions	7
5.2.1	Binary-additive output function	7
5.2.2	MULTI-S01 output function	8
6	Constructing keystream generators from block ciphers	10
6.1	Modes of a block cipher for a synchronous keystream generator	10
6.1.1	OFB mode	11
6.1.2	CTR mode	11
6.2	Mode of a block cipher for a self-synchronizing keystream generator	12
6.2.1	CFB mode	12
7	Dedicated keystream generators	13
7.1	MUGI keystream generator	13
7.1.1	Initialization function <i>Init</i>	14
7.1.2	Next-state function <i>Next</i>	15
7.1.3	Keystream function <i>Strm</i>	15
7.1.4	Function ρ_1	15
7.1.5	Function λ_1	16
7.1.6	Function <i>F</i>	16
7.1.7	Function S_R	17
7.1.8	Function <i>M</i>	18
7.2	SNOW 2.0 keystream generator	18
7.2.1	Initialization function <i>Init</i>	19
7.2.2	Next-state function <i>Next</i>	20
7.2.3	Keystream function <i>Strm</i>	21
7.2.4	Function <i>T</i>	21
7.2.5	Multiplications of α in finite field arithmetic	22
7.2.6	Multiplications of α^{-1} in finite field arithmetic	22
7.2.7	Function <i>FSM</i> (x, y, z)	23
Annex A	(informative) Examples	24
A.1	Operations over the finite field $GF(2^n)$	24
A.2	Example for MUGI	24
A.2.1	Key, initialization vector, and keystream triplets	24
A.2.2	Sample internal states	24
A.3	Example for SNOW 2.0	30
A.3.1	128-bit key	30
A.3.2	256-bit key	34

Annex B (informative) Security information	39
B.1 Security levels of stream ciphers.....	39
B.1.1 Security-efficiency trade-off in MULTI-S01	40
B.2 Implementation examples of dedicated keystream generators	40
Annex C (normative) Object identifiers.....	41
Bibliography	43

AUSTRALIAN/NEW ZEALAND STANDARD

Information technology — Security techniques — Encryption algorithms —

Part 4: Stream ciphers

1 Scope

This part of ISO/IEC 18033 specifies stream cipher algorithms. A stream cipher is an encryption mechanism that uses a keystream to encrypt a plaintext in bitwise or block-wise manner. There are two types of stream cipher: a synchronous stream cipher, in which the keystream is only generated from 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). Typically the encryption operation is the additive bitwise XOR operation between a keystream and the message. This part of ISO/IEC 18033 describes both pseudorandom number generators for producing keystream, and output functions for stream ciphers.

The algorithms specified in this part of ISO/IEC 18033 have been assigned object identifiers in accordance with ISO/IEC 9834. The list of assigned object identifiers is given in Annex C. Any change to the specification of the algorithms resulting in a change of functional behaviour will result in a change of the object identifier assigned to the algorithm.

NOTE 1 – In applications where a combination of algorithms is being used, or when an algorithm is parameterized by the choice of a combination of other algorithms, the combination may be specified as a sequence of object identifiers. Alternatively, the object identifiers of lower layer algorithms may be included in the parameter field of the higher layer algorithm's identifier structure. For example, the object identifier of a block cipher could be included as a parameter in the algorithm identifier structure for a keystream generator. The algorithm identifier structure is defined in ISO/IEC 9594-8.

NOTE 2 – The encoding of object identifiers is application dependent.

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*

ISO/IEC 18033-3, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*

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

a method of storage of multi-byte numbers with the most significant bytes at the lowest memory addresses.
[ISO/IEC 10118-1: 2000]

3.2**block**

string of bits of defined length. [ISO/IEC 18033-1: 2005]

3.3**block cipher**

symmetric encipherment system with the property that the encryption algorithm operates on a block of plaintext, i.e. a string of bits of a defined length, to yield a block of ciphertext. [ISO/IEC 18033-1: 2005]

3.4**cipher**

alternative term for encipherment system. [ISO/IEC 18033-1: 2005]

3.5**ciphertext**

data which has been transformed to hide its information content. [ISO/IEC 10116: 1997]

3.6**confidentiality**

the property that information is not made available or disclosed to unauthorized individuals, entities, or processes. [ISO/IEC 13335-1: 2004]

3.7**data integrity**

the property that data has not been altered or destroyed in an unauthorized manner. [ISO/IEC 9797-1: 1999]

3.8**decipherment**

alternative term for decryption. [ISO/IEC 18033-1: 2005]

3.9**decryption**

reversal of a corresponding encipherment. [ISO/IEC 11770-1: 1996]

3.10**encipherment**

alternative term for encryption. [ISO/IEC 18033-1: 2005]

3.11**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: 1996]

3.12**initialization value**

value used in defining the starting point of an encipherment process. [ISO 8372: 1987]

3.13**key**

sequence of symbols that controls the operation of a cryptographic transformation (e.g. encipherment, decipherment). [ISO/IEC 11770-1: 1996]

3.14**keystream**

pseudorandom sequence of symbols, intended to be secret, used by the encryption and decryption algorithms of a stream cipher. If a portion of the keystream is known by an attacker, then it shall be computationally infeasible for the attacker to deduce any information about the remainder of the keystream. [ISO/IEC 18033-1:2005]

3.15**keystream function**

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

3.16**keystream generator**

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

3.17 **n -bit block cipher**

block cipher with the property that plaintext blocks and ciphertext blocks are n bits in length.

[ISO/IEC 10116:1997]

3.18**next-state function**

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

3.19**output function**

output function that combines the keystream and the plaintext to produce the ciphertext. This function is often bitwise XOR.

3.20**padding**

appending extra bits to a data string. [ISO/IEC 10118-1: 2000]

3.21**plaintext**

unenciphered information. [ISO/IEC 9797-1: 1999]

3.22**secret key**

key used with symmetric cryptographic techniques by a specified set of entities. [ISO/IEC 11770-3: 1999]

3.23**self-synchronizing stream cipher**

stream cipher with the property that the keystream symbols are generated as a function of a secret key and a fixed number of previous ciphertext bits. [ISO/IEC 18033-1: 2005]

3.24**state**

current internal state of a keystream generator.

3.25**stream cipher**

symmetric encryption system with the property that the encryption algorithm involves combining a sequence of plaintext symbols with a sequence of keystream symbols one symbol at a time, using an invertible function. Two types of stream cipher can be identified: synchronous stream ciphers and self-synchronizing stream ciphers, distinguished by the method to obtain the keystream. [ISO/IEC 18033-1: 2005]

3.26**synchronous stream cipher**

stream cipher with the property that the keystream symbols are generated as a function of a secret key, and are independent of the plaintext and ciphertext. [ISO/IEC 18033-1: 2005]

4 Symbols and abbreviated terms

$0^{(n)}$	n -bit variable where 0 is assigned to every bit.
0x	Prefix for hexadecimal values.
a_i	Variables in an internal state of a keystream generator.
b_i	Variables in an internal state of a keystream generator.
C_i	Ciphertext block.
CFB	Cipher FeedBack mode of a block cipher.
CTR	CounTeR mode of a block cipher.
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.
$F[x]$	The polynomial ring over the finite field F .
$GF(2^n)$	Finite field of exactly 2^n elements.
$Init$	Function which generates the initial internal state of a keystream generator.
IV	Initialization vector.
K	Key.
M	Subfunction used for MUGI.
n	Block length.
$Next$	Next-state function of a keystream generator.
OFB	Output FeedBack mode of a block cipher.
OR	Bitwise 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_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 S_i values for $i < 0$.

S_i	Subfunction used for MUGI.
-------	----------------------------

$Strm$	Keystream function of a keystream generator.
SUB	Lookup table used for MUGI and SNOW 2.0.
T	Subfunction used for SNOW 2.0.
Z	Keystream.
Z_i	Keystream block.
α_{MUL}	Lookup table used for SNOW 2.0.
α_{inv_MUL}	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.
$\parallel$	Bit concatenation.
$+_m$	Integer addition modulo 2^m .
$\oplus$	Bitwise XOR (eXclusive OR) operation.
$\otimes$	Operation of multiplication of elements in the finite field $GF(2^n)$. E.g. $C = A \otimes B$: 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 $A = \{a_{n-1}, a_{n-2}, \dots, a_0\}$ and $B = \{b_{n-1}, b_{n-2}, \dots, b_0\}$ (where the a_i and b_i are bits) are represented as the polynomials, $A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_0$ 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 $\{c_{n-1}, c_{n-2}, \dots, c_0\}$.
$\ll_n t$	t -bit left shift in an n -bit register.
$\gg_n t$	t -bit right shift in an n -bit register.
$\lll_n t$	t -bit left circular rotation in an n -bit register.
$\ggg_n t$	t -bit right circular rotation in an n -bit register.

4.1 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 $1 \leq j \leq m$. [ISO/IEC 10116]

4.2 Shift operation

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

$$\text{Shift}_k(X \mid F) = (x_k, x_{k+1}, \dots, x_{n-1}, f_0, f_1, \dots, f_{k-1}) \quad (k < n)$$

$$\text{Shift}_k(X \mid F) = (f_0, f_1, \dots, f_{k-1}) \quad (k = n)$$

The effect is to shift the bits of array X left by k places, discarding $x_0, x_1, \dots, x_{k-1}$ and to place the array F in the rightmost k places of X . When $k = n$ the effect is to totally replace X by F . [ISO/IEC 10116]

4.3 Variable $I(k)$

The variable $I(k)$ is a k -bit variable where 1 is assigned to every bit. [ISO/IEC 10116]

5 General models for stream ciphers

In this clause general models for stream ciphers are specified. A stream cipher consists of the combination of a keystream generator and an output function.

5.1 Keystream generators

5.1.1 Synchronous keystream generators

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

1. 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 .
2. 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} .
3. 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

$$S_0 = \text{Init}(IV, K).$$

On demand the synchronous keystream generator will for $i=0,1,\dots$:

1. Output a keystream block $Z_i = \text{Strm}(S_i, K)$.
2. Update the state of the machine $S_{i+1} = \text{Next}(S_i, K)$.

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

5.1.2 Self-synchronizing keystream generators

Generation of 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.

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 binary additive mode.

1. 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 $C_{-1}, C_{-2}, \dots, C_{-r}$.
2. A keystream function, *Strm*, that takes as input *S* and *r* ciphertext blocks $C_{i-1}, C_{i-2}, \dots, C_{i-r}$, 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*.

5.2 Output functions

In this clause two stream cipher output functions are specified, 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 an invertible function *Out* that combines a plaintext block P_i , a keystream block Z_i and, optionally, some other input *R* to give a ciphertext block C_i ($i \geq 0$). A general model for stream cipher output function is now defined.

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

$$C_i = \text{Out}(P_i, Z_i, R),$$

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

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

The output function shall be such that, for any keystream block Z_i , plaintext block P_i , and other input *R*, we have that

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

5.2.1 Binary-additive output function

A binary-additive stream cipher is a stream cipher in which the keystream, plaintext, and ciphertext blocks are 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 its calculation. Let *n* to be the bit length of P_i . This function is specified by

$$\text{Set } R = 0^{(n)}.$$

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

The operation Out^{-1} is specified by

$$\text{Set } R = 0^{(n)}.$$

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

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 Message Authentication Code (such mechanisms are specified in ISO/IEC 9797).

5.2.2 MULTI-S01 output function

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 MULTI-S01 function only accepts messages whose length is a multiple of n . To encrypt messages whose length is not a multiple of n , it is necessary to pad the plaintext. A padding technique is described in clause 5.2.2.3. In addition to P and Z , the MULTI-S01 takes as another input some redundancy R . One possibility is to make R a fixed public n -bit value. This should be available to both the encryption and decryption processes.

NOTE 1 An encryption mechanism based on MULTI-S01 is secure as long as the underlying keystream generator is secure. The security level with respect to data integrity is generally as high as the security level of the keystream generator. However the security level is always upper-bounded by the length of a forged message. It is believed that a forged message of length $u \cdot n$ bits will be accepted with probability at most $(u-2) \cdot 2^{-n}$, where n is the security parameter of the mode. For practical purposes, the parameter n should be at least 64. For higher security, $n=128$ is recommended.

NOTE 2 – MULTI-S01 was originally proposed in [3].

5.2.2.1 *Out*(P, Z, R)

The *Out* function operates in an iterative fashion, processing n bits of plaintext per iteration.

INPUT: $n \cdot u$ -bit plaintext P , keystream Z , 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 $(P_0, P_1, \dots, P_{u-1}) = P$, where P_i is an n -bit block.
3. Set $P_u = Z_{t+u+3}$.
4. Set $P_{u+1} = R$.
5. Set $W_{-1} = 0^{(n)}$.
6. For $i = 0, 1, \dots, u+1$ do the following calculations:
 - 6.1. Let $W_i = P_i \oplus Z_{t+i+1}$.
 - 6.2. Let $X_i = Z_i \otimes W_i$ (in $\text{GF}(2^n)$).
 - 6.3. Let $C_i = X_i \oplus W_{i-1}$.
7. Set $C = C_0 \parallel C_1 \parallel \dots \parallel C_{u+1}$.
8. Output C .

Figure 1 shows the operation of the *Out* function.

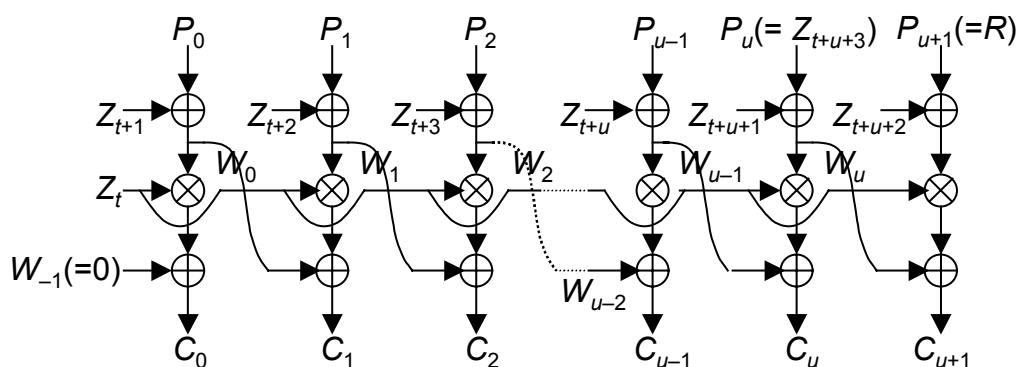


Figure 1 — *Out* function of MULTI-S01 mode

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

5.2.2.2 *Out*⁻¹(C, Z, R)

Although the *Out*⁻¹ function does not output any text before the integrity check, the core computation of the *Out*⁻¹ function is also incremental and keeps an internal variable W . To generate a pre-plaintext value P_i , the function requires W_{i-1} value in addition to C_i , generating W_i for the following process.

INPUT: n - 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 $(C_0, C_1, \dots, C_{v-1}) = C$, where C_i is an n -bit block.
3. For each C_i , do the following calculations (for $i = 0, 1, \dots, v-1$):
 - 3.1. Let $X_i = C_i \oplus W_{i-1}$, where $W_{-1} = 0^{(n)}$.
 - 3.2. Let $W_i = Z_t^{-1} \otimes X_i$ (in $\text{GF}(2^n)$).
 - 3.3. 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 $P = P_0 \parallel P_1 \parallel \dots \parallel P_{v-3}$ as plaintext. Otherwise output the special symbol meaning “*reject*” without any text.

Figure 2 shows the operation of the Out^{-1} function.

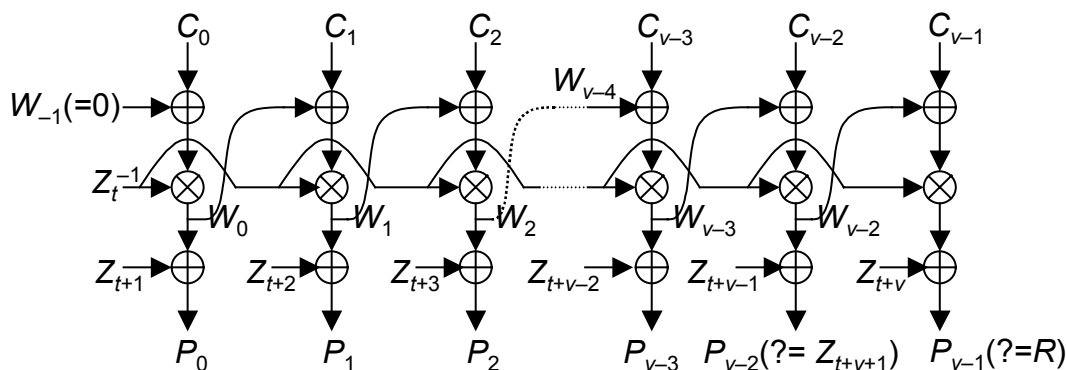


Figure 2 — Out^{-1} function of MULTI-S01 mode

5.2.2.3 Padding mechanism $Pad(M)$

There is a recommended padding method for implementations where lengths of input messages are not guaranteed to be multiples of n .

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

OUTPUT: Padded plaintext P .

1. Append a single '1' bit to the end of the message.
2. Append the $(n - c - 1)$ -bit string $0^{(n - c - 1)}$ to the end of the string generated by step 1.
3. Output the whole data string of length $(nv + n)$ bits.

NOTE 1 – If the length of the message is a multiple of n in an environment where this situation is not guaranteed, then use of this padding mechanism is recommended.

NOTE 2 – In order to unpad the message, remove consecutive 0 bits at the end of the data, and remove another bit "1".

6 Constructing keystream generators from block ciphers

In this clause three modes of operation for block ciphers are specified. These modes are standardised in ISO/IEC 10116, and the meaning of the functions used in the specification is defined in clauses 5.1.1 and 5.1.2 of this document. These three block cipher modes of operation are methods by which a block cipher can be used to construct a keystream generator. The first two of these three block cipher modes of operation, specified in clause 6.1, define synchronous stream ciphers; the third block cipher mode, specified in clause 6.2, defines a self-synchronizing stream cipher.

NOTE – Block ciphers are defined in part 3 of this international standard.

6.1 Modes of a block cipher for a synchronous keystream generator

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