INTERNATIONAL STANDARD

ISO/IEC 10116

Fourth edition
2017-07

Information technology — Security techniques — Modes of operation for an *n*-bit block cipher

Technologies de l'information — Techniques de sécurité — Modes opératoires pour un chiffrement par blocs de n *bits*

Reference number
ISO/IEC 10116:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

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

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

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, SC 27, *IT Security techniques*.

This fourth edition cancels and replaces the third edition (ISO/IEC 10116:2006) and ISO/IEC 10116:2006/Cor1:2008, which have been technically revised.

The main technical changes between the third edition and this fourth edition are as follows:

a) the inclusion of padding within the normative scope of ISO/IEC 10116;

b) the inclusion of methods for avoiding ciphertext expansion for CBC, CFB, OFB and CTR modes.

Introduction

This document specifies modes of operation for an *n*-bit block cipher. These modes provide methods for encrypting and decrypting data using a block cipher.

This fourth edition of ISO/IEC 10116 specifies five modes of operation:

a) Electronic Codebook (ECB);

b) Cipher Block Chaining (CBC);

c) Cipher Feedback (CFB);

d) Output Feedback (OFB);

e) Counter (CTR).

NOTE Annex C presents figures describing the modes of operation. Annex D provides numerical examples of the modes of operation.

Information technology — Security techniques — Modes of operation for an *n*-bit block cipher

1 Scope

This document establishes five modes of operation for applications of an *n*-bit block cipher (e.g. protection of data during transmission or in storage). The defined modes only provide protection of data confidentiality. Protection of data integrity is not within the scope of this document. Also, most modes do not protect the confidentiality of message length information.

NOTE 1 Methods for protecting the integrity of data using a block cipher are provided in ISO/IEC 9797-1.

NOTE 2 Methods for simultaneously protecting the confidentiality and integrity of data are provided in ISO/IEC 19772.

This document specifies the modes of operation and gives recommendations for choosing values of parameters (as appropriate).

NOTE 3 The modes of operation specified in this document have been assigned object identifiers in accordance with ISO/IEC 9834. The list of assigned object identifiers is given in Annex A. In applications in which object identifiers are used, the object identifiers specified in Annex A are to be used in preference to any other object identifiers that can exist for the mode concerned.

NOTE 4 Annex B contains comments on the properties of each mode and important security guidance.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 18033-3, *Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers*.

ISO/IEC 29192-2, *Information technology — Security techniques — Lightweight cryptography — Part 2: Block ciphers*.

3 Terms and definitions

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

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

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

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

3.1
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

[SOURCE: ISO/IEC 18033-1:2015, 2.9]

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

[SOURCE: ISO/IEC 18033-1:2015, 2.11]

3.3
counter
bit array of length *n* bits (where *n* is the size of the underlying block cipher) which is used in the Counter mode

Note 1 to entry: The value when considered as the binary representation of an integer increases by one (modulo 2^n) after each block of plaintext is processed.

3.4
cryptographic synchronization
co-ordination of the encryption and decryption processes

3.5
decryption
reversal of a corresponding encryption

[SOURCE: ISO/IEC 18033-1:2015, 2.16, modified]

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

[SOURCE: ISO/IEC 18033-1:2015, 2.21]

3.7
feedback buffer
FB
variable used to store input data for the encryption process

Note 1 to entry: At the starting point, *FB* has the value of *SV.*

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

[SOURCE: ISO/IEC 18033-1:2015, 2.27, modified]

3.9
***n*-bit block cipher**
block cipher with the property that plaintext blocks and ciphertext blocks are *n* bits in length

[SOURCE: ISO/IEC 18033-1:2015, 2.29]

3.10
padding
appending extra bits to a data string

3.11
plaintext
unencrypted information

[SOURCE: ISO/IEC 18033-1:2015, 2.30]

3.12
starting variable
SV
variable possibly derived from some initialization value and used in defining the starting point of the modes of operation

Note 1 to entry: The “starting variable” (*SV*) used in this document is similar to (possibly identical to) the “initialization value” or “initialization vector” (IV) used in some other International Standards. If the starting variable referred to in this document is derived from some initialization value, then it needs to be described in any application of the modes of operation. A method for deriving a starting variable from an initializing value is not defined in this document.

4 Symbols, abbreviated terms and notation

4.1 Symbols and abbreviated terms

C	Ciphertext (string of bits)
CTR	Counter value
dK	Decryption function of the block cipher keyed by key K. The decryption relation defined by the block cipher is written $P = dK(C)$ where — P is the plaintext block; — C is the ciphertext block; — K is the key.
E	Intermediate variable
eK	Encryption function of the block cipher keyed by key K. The encryption relation defined by the block cipher is written $C = eK(P)$ where — P is the plaintext block; — C is the ciphertext block; — K is the key.
F	Intermediate variable
FB	Feedback buffer
i	Iteration
j	Size of plaintext/ciphertext variable
K	Key
m	Number of stored ciphertext blocks
n	Plaintext/ciphertext block length for a block cipher
P	Plaintext (string of bits)
q	Number of plaintext/ciphertext variables
r	Size of feedback buffer
SV	Starting variable (see Clause 5)
X	Block cipher input block
Y	Block cipher output block

$\|$	Concatenation of bit strings
$(a_1, a_2, \ldots, a_m)$	A one-dimensional array of bits. For example, $A = (a_1, a_2, \ldots, a_m)$ and $B = (b_1, b_2, \ldots, b_m)$ are two arrays of m bits, numbered from 1 to m. All arrays of bits are written with the bit with the index 1 in the leftmost position. When interpreting a bit array as an integer, the leftmost bit shall be the most significant bit.
$\oplus$	The operation of bitwise addition, modulo 2, also known as the “exclusive or” function, is shown by the symbol $\oplus$. The operation when applied to arrays A and B of the same length is defined as $A \oplus B = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_m \oplus b_m)$.

4.2 Notation

$a \bmod n$	For integers a and n, $a \bmod n$ denotes the (non-negative) remainder obtained when a is divided by n. Equivalently, if $b = a \bmod n$, then b is the unique integer satisfying: — $0 \le b < n$, and — $(b - a)$ is an integer multiple of n.
~	The operation of selecting the j leftmost bits of an array $A = (a_1, a_2, \ldots, a_m)$ to generate a j-bit array is written $(j \sim A) = (a_1, a_2, \ldots, a_j)$ The operation of selecting the j rightmost bits of an array $A = (a_1, a_2, \ldots, a_m)$ to generate a j-bit array is written $(A \sim j) = (a_{\text{m-j+1}}, a_{\text{m-j+2}}, \ldots, a_m)$ The operation is defined only when $1 \le j \le m$.
$1(t)$	t-bit string where the value 1 is assigned to every bit.
S_t	Given an m-bit variable X and a t-bit variable F where $1 \le t \le m$, the effect of the shift function $S_t(X \| F)$ is to produce the m-bit variable $S_t(X \| F) = (x_{t+1}, x_{t+2}, \ldots, x_m, f_1, f_2, \ldots, f_t)$ $(t < m)$ $S_t(X \| F) = (f_1, f_2, \ldots, f_t)$ $(t = m)$ The effect is to shift the bits of array X left by t places, discarding $x_1, x_2, \ldots, x_t$, and to place the array F in the rightmost t places of X. When $t = m$ the effect is to totally replace X by F.

5 Requirements

For all of the described modes, a block cipher shall be selected from ISO/IEC 18033-3 and/or ISO/IEC 29192-2.

For the Electronic Codebook (ECB) mode, there are no additional parameter values that need to be selected. For the Cipher Block Chaining (CBC) mode of operation (see Clause 7), one parameter m shall be selected. For the Cipher Feedback (CFB) mode of operation (see Clause 8), three parameters r, j and k need to be selected. For the Output Feedback (OFB) mode of operation (see Clause 9) and the Counter (CTR) mode of operation (see Clause 10), one parameter j needs to be selected. When one of these modes of operation is used, the same parameter value(s) need to be chosen and used by all communicating parties. These parameters need not be kept secret.

All modes of operation specified in this document require the parties encrypting and decrypting a data string to share a secret key K for the block cipher in use. All modes of operation apart from the electronic Codebook (ECB) mode also require the parties to share a starting variable SV, where the length of SV will depend on the mode in use. The value of the starting variable should normally be different for every data string encrypted using a particular key. How keys and starting variables are managed and distributed is outside the scope of this document; however, important security guidance related to starting variables is provided in Annex B.

The encrypter and all potential decrypters shall agree on a padding method, unless messages to be encrypted are always a multiple of m bits ($m = n$ for ECB and CBC modes, $m = j$ for CFB, OFB and CTR modes) in length or unless the mode does not require padding. Such a padding method shall take a (non-empty) bit string P of arbitrary length as input and give a sequence of m-bit plaintext blocks as output. Unless the length of the plaintext message is fixed or otherwise defined by the application, then the padding method shall have the property that no two distinct input bit strings can give the same sequence of m-bit blocks as output, i.e. padded data strings can always be uniquely unpadded.

This document recommends the following padding method: the (non-empty) bit string P is right-padded with a single ‘1’ bit. The resulting string is then right-padded with as few (possibly none) ‘0’ bits as necessary to obtain a string whose length (in bits) is a positive integer multiple of m.

NOTE This padding method is identical to padding method 2 in both ISO/IEC 9797-1 and ISO/IEC 10118-1 except that it only considers non-empty bit strings.

6 Electronic Codebook (ECB) mode

6.1 Preliminaries

ECB mode shall be used with an agreed padding method if the bit length of the plaintext might not be a multiple of the block cipher block length n.

The variables employed by the ECB mode of encryption are

a) the input variables

   1) A plaintext string of bits P.

   2) A key K.

b) the output variables, i.e. a ciphertext string of bits C.

6.2 Encryption

If a padding method is to be used, then first pad the data string P to obtain a sequence of q plaintext blocks P_1, P_2, ..., P_q, each of n bits. Otherwise, simply divide the data string P into a sequence of q plaintext blocks P_1, P_2, ..., P_q, each of n bits.

The ECB mode of encryption continues as follows:

$$C_i = eK(P_i) \text{ for } i = 1, 2, \ldots, q.$$

6.3 Decryption

The ECB mode of decryption operates as follows:

$$P_i = dK(C_i) \text{ for } i = 1, 2, \ldots, q.$$

If a padding method is not in use, then P is equal to the concatenation of P_1, P_2, ..., P_q.

If a padding method is in use, apply the inverse of the agreed padding procedure to the sequence of q plaintext blocks P_1, P_2, ..., P_q to obtain the decrypted plaintext string P.

NOTE Unique removal of padding is guaranteed because of the constraint on the choice of padding method given in Clause 5.

7 Cipher Block Chaining (CBC) mode

7.1 Preliminaries

CBC mode encryption and decryption as defined in 7.2 and 7.3 shall use an agreed padding method if the bit length of the plaintext might not be a multiple of the block cipher block length n. If no padding method is agreed, then the plaintext shall be a whole number of complete blocks unless it is agreed to use the approach described in 7.4.

The CBC mode of operation is defined by an interleave parameter $m > 0$, the number of independent encryption processes that could operate in parallel.

NOTE 1 The value of m can be small (typically $m = 1$) and can rarely exceed 1 024.

The variables employed by the CBC mode when being used for encryption are

a) the input variables

1) A plaintext string of bits P.

2) A key K.

3) A sequence of m starting variables $SV_1, SV_2, \ldots, SV_m$ each of n bits. Refer to Annex B for security guidance related to the value of the starting variables.

NOTE 2 If $m = 1$ and no padding is applied, then this mode is compatible with the CBC mode described in the second edition of this document.

b) the output variables, i.e. a ciphertext string of bits C.

7.2 Encryption

If a padding method is to be used, then first pad the data string P to obtain a sequence of q plaintext blocks P_1, P_2, ..., P_q, each of n bits. Otherwise, simply divide the data string P into a sequence of q plaintext blocks P_1, P_2, ..., P_q, each of n bits.

The CBC mode of encryption operates as follows:

$$C_i = eK(P_i \oplus SV_i),\ 1 \leq i \leq \min(m, q)$$

If q > m, all subsequent plaintext blocks are encrypted as:

$$C_i = eK(P_i \oplus C_{i-m}),\ m + 1 \leq \mathrm{i} \leq q$$

At any time during the computation, the values of the m most recent ciphertext blocks need to be stored, e.g. in a cyclically used “feedback buffer” FB (see Figure C.2).

This procedure is shown on the left side of Figure C.2.

7.3 Decryption

The CBC mode of decryption operates as follows:

$$P_i = dK(C_i) \oplus SV_i,\ 1 \leq i \leq \min(m, q)$$

If $q > m$, all subsequent ciphertext blocks are decrypted as:

$$P_i = dK(C_i) \oplus C_{i-m},\ m + 1 \leq i \leq q$$

At any time during the computation, the values of the m most recent ciphertext blocks need to be stored, e.g. in a cyclically used “feedback buffer” FB (see Figure C.2).

This procedure is shown on the right side of Figure C.2.

If a padding method is not in use, then P is equal to the concatenation of P_1, P_2, ..., P_q.

If a padding method is in use, apply the inverse of the agreed padding procedure to the sequence of q plaintext blocks P_1, P_2, ..., P_q to obtain the decrypted plaintext string P.

NOTE Unique removal of padding is guaranteed because of the constraint on the choice of padding method given in Clause 5.

7.4 Avoiding ciphertext expansion

7.4.1 General

In order to avoid ciphertext expansion caused by padding, it is possible to implement ‘ciphertext stealing’. With this method, if the final plaintext block is partial, then the fewest ‘0’ bits are appended so as to complete it (i.e., padding method 1 in ISO/IEC 9797-1:2011) and the resulting plaintext is encrypted as above. Any expansion that would be caused by padding bits is avoided, because bits of the penultimate ciphertext block are discarded as they can be recovered from the decryption of the final ciphertext block.

Specifically, if m=1 (no interleaving) and the plaintext has been padded so that the rightmost p bits of the final plaintext block P_q are the ‘0’ padding bits, then the rightmost p bits of the penultimate ciphertext block $C_{q\text{-}1}$ are not transmitted but can be recovered as the rightmost p bits of $dK(C_q)$.

NOTE Ciphertext stealing cannot be applied if the number of bits in the plaintext is less than n.

7.4.2 Three ciphertext stealing variants of CBC

7.4.2.1 Preliminaries

This document defines three ciphertext stealing variants (CBC_CS) of CBC mode. All three are variants of the basic CBC encryption/decryption defined above using padding method 1 in ISO/IEC 9797-1:2011 [1]. These CBC_CS variants differ from the basic CBC mode only in how the two ciphertext blocks $C_{q\text{-m}}$ and C_q are processed after encryption and before decryption.

Although the CBC_CS variants can be used when the interleave parameter m>1, as noted in 7.1, typically, m=1 in which case the two ciphertext blocks $C_{q\text{-}m}$ and C_q blocks are the final two ciphertext blocks $C_{q\text{-}1}$ and C_q. The description below applies only to the case m=1 but can be extrapolated to the case m>1.

7.4.2.2 Processing after CBC encryption (ciphertext contraction)

After CBC encryption the ciphertext $C_1 \mid C_2 \mid ... \mid C_{q\text{-}1} \mid C_q$ is contracted (reduced in length) by the number p ($0 \leq p < n$) of padding bits. If no padding occurred (p=0), then no contraction occurs but, as specified below, the third CBC_CS variant still modifies the ciphertext by swapping the final two ciphertext blocks $C_{q\text{-}1}$ and C_q :

For all CBC_CS variants define, $C_{q\text{-}1}^* = (n\text{-}p) \sim C_{q\text{-}1}$. The three CBC_CS variants are defined as follows:

- For the first CBC_CS variant, the final two ciphertext blocks $C_{q\text{-}1} \mid C_q$ are replaced by $C_{q\text{-}1}^* \mid C_q$.
- For the second CBC_CS variant, the final two ciphertext blocks $C_{q\text{-}1} \mid C_q$ are replaced by $C_q \mid C_{q\text{-}1}^*$ only if the plaintext was padded (otherwise no change is made).
- For the third CBC_CS variant, the final two ciphertext blocks $C_{q\text{-}1} \mid C_q$ are always replaced by $C_q \mid C_{q\text{-}1}^*$.

7.4.2.3 CBC decryption pre-processing (ciphertext extension)

Before CBC decryption, the received contracted ciphertext is extended to a whole number of n-bit blocks.

Let the number of bits in the received ciphertext be $(q\text{-}1)n+d$ where $0 \le d < n$ and if $d>0$ then let $p=n\text{-}d$ (note in this case that $p \ne 0$).

If $q > 1$ then define $C_{q\text{-}1}{}^* = (n\text{-}p) \sim C_{q\text{-}1}$.

- For the first CBC_CS variant
  - if d=0, then the received ciphertext is a whole number of n-bit blocks and no pre-processing is required.
  - if d>0, then the received ciphertext is not a whole number of n-bit blocks and pre-processing is required. In this case, the rightmost $n+d$ bits of the received ciphertext are parsed into $C_{q\text{-}1}{}^* \mid C_q$ where $C_{q\text{-}1}{}^*$ is a d-bit block and C_q is an n-bit block and the n-bit block $C_{q\text{-}1}$ is formed as $C_{q\text{-}1}{}^* \mid (dK(C_q) \sim p)$.
- For the second CBC_CS variant
  - if d=0, then the received ciphertext is a whole number of n-bit blocks and no pre-processing is required.
  - If d>0, then the received ciphertext is not a whole number of n-bit blocks and pre-processing is required. In this case, the rightmost $n+d$ bits of the received ciphertext are parsed into $C_q \mid C_{q\text{-}1}{}^*$, where $C_{q\text{-}1}{}^*$ is a d-bit block and C_q is an n-bit block and then the n-bit block $C_{q\text{-}1}$ is formed as $C_{q\text{-}1}{}^* \mid (dK(C_q) \sim p)$.
- For the third CBC_CS variant, pre-processing is always required
  - if d=0, then the received ciphertext is a whole number of n-bit blocks the rightmost 2n bits of the received ciphertext are parsed as $C_q \mid C_{q\text{-}1}$. Thus, pre-processing requires that the rightmost two blocks are swapped.
  - If d>0, then the received ciphertext is not a whole number of n-bit blocks. In this case, the rightmost $n+d$ bits of the received ciphertext are parsed into $C_q \mid C_{q\text{-}1}{}^*$, where $C_{q\text{-}1}{}^*$ is a d-bit block and C_q is an n-bit block, and the n-bit block $C_{q\text{-}1}$ is formed as $C_{q\text{-}1}{}^* \mid (dK(C_q) \sim p)$.

CBC decryption then decrypts the ciphertext $C_1 \mid C_2 \mid \ldots \mid C_{q\text{-}1} \mid C_q$ and, if d>0, removes the rightmost p padding bits from the resulting plaintext blocks.

NOTE NIST/SP 800-38A defines three similar variants of ciphertext stealing (the variants differ only in the ordering of the ciphertext bits).

8 Cipher Feedback (CFB) mode

8.1 Preliminaries

CFB mode encryption and decryption as defined in 8.2 and 8.3 shall use an agreed padding method if the bit length of the plaintext might not be a multiple of the parameter j. If no padding method is agreed, then the plaintext shall be a whole number of j-bit blocks unless the approach described in 8.4 is used.

Three parameters define a CFB mode of operation:

- the size of feedback buffer, r, where $n \le r \le 1\,024n$
- the size of feedback variable, k, where $1 \le k \le n$
- the size of plaintext variable, j, where $1 \le j \le k$

NOTE a) $r - k$ is not constrained by n in any way, i.e. $r - k$ can be less than, equal to or greater than n. Figure C.3 shows the special case where $r - k > n$.

b) If $r = n$, then this mode is compatible with the version of CFB mode described in the first edition of this document.

c) the upper bound on r, i.e. $r \leq 1\,024n$ is chosen because it provides a realistic upper bound on the number of hardware processors.

It is recommended that CFB should be used with equal values of j and k (see B.4.1).

CFB mode shall be used with an agreed padding method if the bit length of the plaintext might not be a multiple of the parameter j.

The number of j-bit plaintext blocks (after padding if a padding method is in use) is q.

The variables employed by the CFB mode of operation when being used for encryption are

a) the input variables

   1) A plaintext string of bits P.

   2) A key K.

   3) A starting variable SV of r bits. Refer to Annex B for security guidance related to the value of SV.

b) the intermediate results

   1) A sequence of q block cipher input blocks $X_1, X_2, \ldots, X_q$, each of n bits.

   2) A sequence of q block cipher output blocks $Y_1, Y_2, \ldots, Y_q$, each of n bits.

   3) A sequence of q variables $E_1, E_2, \ldots, E_q$, each of j bits.

   4) A sequence of $q - 1$ feedback variables $F_1, F_2, \ldots, F_{q-1}$, each of k bits.

   5) A sequence of q feedback buffer contents $FB_1, FB_2, \ldots, FB_q$ each of r bits.

c) The output variables, i.e. a ciphertext string of bits C.

8.2 Encryption

If a padding method is to be used, then first pad the data string P to obtain a sequence of q plaintext blocks $P_1, P_2, \ldots, P_q$, each of j bits. Otherwise, simply divide the data string P into a sequence of q plaintext blocks $P_1, P_2, \ldots, P_q$, each of j bits.

The feedback buffer, FB, is set to its starting variable

$$FB_1 = SV$$

The operation of encrypting each plaintext variable employs the following six steps.

a) $X_i = n \sim FB_i$ (selection of leftmost n bits of FB).

b) $Y_i = eK(X_i)$ (use of block cipher).

c) $E_i = j \sim Y_i$ (selection of leftmost j bits of Y_i).

d) $C_i = P_i \oplus E_i$ (generation of ciphertext variable).

e) $F_i = I(k - j) \mid C_i$ (generation of feedback variable).

f) $FB_{i+1} = S_k(FB_i \mid F_i)$ (shift function on FB).

These steps are repeated for $i = 1, 2, \ldots, q$, ending with step (d) on the last cycle. The procedure is shown on the left side of Figure C.3. The leftmost j bits of the output block Y of the block cipher are used to encrypt the j-bit plaintext variable by modulo 2 addition. The remaining bits of Y are discarded. The plaintext and ciphertext variables have bits numbered from 1 to j.

The ciphertext variable is augmented by placing $k - j$ one bits in its leftmost bit positions to become the k-bit feedback variable F. Then the bits of the feedback buffer FB are shifted left by k places and F is inserted in the rightmost k places to produce the new value of the feedback buffer FB. In this shift operation, the leftmost k bits of FB are discarded. The new n leftmost bits of FB are used as the next input X of the encryption process.

8.3 Decryption

The variables employed for decryption are the same as those employed for encryption.

The feedback buffer, FB, is set to its starting variable

$$FB_1 = SV$$

The operation of decrypting each ciphertext variable employs the following six steps.

a) $X_i = n \sim FB_i$ (selection of leftmost n bits of FB).

b) $Y_i = eK(X_i)$ (use of block cipher).

c) $E_i = j \sim Y_i$ (selection of leftmost j bits of Y_i).

d) $P_i = C_i \oplus E_i$ (generation of plaintext variable).

e) $F_i = I(k - j) \mid C_i$ (generation of feedback variable).

f) $FB_{i+1} = S_k(FB_i \mid F_i)$ (shift function on FB).

These steps are repeated for $i = 1, 2, ..., q$, ending with step (d) on the last cycle. The procedure is shown on the right side of Figure C.3. The leftmost j bits of the output block Y of the block cipher are used to decrypt the j-bit ciphertext variable by modulo 2 addition. The remaining bits of Y are discarded. The plaintext and ciphertext variables have bits numbered from 1 to j.

The ciphertext variable is augmented by placing $k - j$ one bits in its leftmost bit positions to become the k-bit feedback variable F. Then the bits of the feedback buffer FB are shifted left by k places and F is inserted in the rightmost k places, to produce the new value of the feedback buffer FB. In this shift operation, the leftmost k bits of FB are discarded. The new n leftmost bits of FB are used as the next input X of the decryption process.

If a padding method is not in use, then P is equal to the concatenation of $P_1, P_2, ..., P_q$.

If a padding method is in use, apply the inverse of the agreed padding procedure to the sequence of q plaintext blocks $P_1, P_2, ..., P_q$ to obtain the decrypted plaintext string P.

NOTE Unique removal of padding is guaranteed because of the constraint on the choice of padding method given in Clause 5.

8.4 Avoiding ciphertext expansion

Using CFB mode, it is possible to avoid ciphertext expansion in the case where the plaintext is not a whole number of j-bit blocks. With this approach, if the plaintext is not a whole number of j-bit blocks, then the plaintext is padded with p $(0 \leq p < j)$ padding bits so that the plaintext becomes a whole number of j-bit blocks but the padding is not included in the ciphertext. The actual value of the padding bits is irrelevant so they could be binary zeroes. The padded plaintext is encrypted as above but the final p bits of the final ciphertext block are discarded because they are not needed.

The decrypter then applies the same padding to produce a ciphertext that is a whole number of *j*-bit blocks, decrypts the ciphertext as above and then discards the final *p* bits of the plaintext.

NOTE 1 One can equally formulate this approach without reference to padding by simply requiring that the final encryption variable E_q be truncated to be the length of the final plaintext/ciphertext bits. Indeed, Annex B of the third edition of this document noted that "*j* can be modified for the last portion of the plaintext. No padding is required when a plaintext block with *z* bits, $z < j$ is encrypted by bitwise addition with only the first *z* bits of the corresponding variable *E*".

NOTE 2 If a padding method has been agreed as in Clause 5, then this approach is not used. If a padding method has not been agreed, then this approach is used.

9 Output Feedback (OFB) mode

9.1 Preliminaries

The OFB mode of operation is defined by one parameter, i.e. the size of the plaintext variable *j*, where $1 \leq j \leq n$.

OFB mode encryption and decryption as defined in 9.2 and 9.3 shall use an agreed padding method if the bit length of the plaintext might not be a multiple of the parameter *j*. If no padding method is agreed, then the plaintext shall be a whole number of *j*-bit blocks unless the approach described in 9.4 is used.

The number of *j*-bit plaintext blocks (after padding if a padding method is in use) is *q*.

The variables employed by the OFB mode of operation when being used for encryption are

a) the input variables

   1) A plaintext string of bits *P*.

   2) A key *K*.

   3) A starting variable *SV* of *n* bits. Refer to Annex B for security guidance related to the value of *SV*.

b) the intermediate results

   1) A sequence of *q* block cipher input blocks $X_1, X_2, \ldots, X_q$, each of *n* bits.

   2) A sequence of *q* block cipher output blocks $Y_1, Y_2, \ldots, Y_q$, each of *n* bits.

   3) A sequence of *q* variables $E_1, E_2, \ldots, E_q$, each of *j* bits.

c) The output variables, i.e. a ciphertext string of bits *C*.

9.2 Encryption

If a padding method is to be used, then first pad the data string *P* to obtain a sequence of *q* plaintext blocks $P_1, P_2, ..., P_q$, each of *j* bits. Otherwise, simply divide the data string *P* into a sequence of *q* plaintext blocks $P_1, P_2, ..., P_q$, each of *j* bits.

The input block *X* is set to the starting variable

$$X_1 = SV$$

The operation of encrypting each plaintext variable employs the following four steps.

a) $Y_i = eK(X_i)$ (use of block cipher).

b) $E_i = j \sim Y_i$ (selection of leftmost *j* bits of Y_i).

c) $C_i = P_i \oplus E_i$ (generation of ciphertext variable).

d) $X_{i+1} = Y_i$ (feedback operation).

These steps are repeated for $i = 1, 2, ..., q$, ending with step (c) on the last cycle. The procedure is shown on the left side of Figure C.4. The plaintext and ciphertext variables have bits numbered from 1 to j.

The result of each use of the block cipher is Y_i and this is fed back to become the next value of X, namely X_{i+1}. The leftmost j bits of Y_i are used to encrypt the input variable.

9.3 Decryption

The variables employed for decryption are the same as those employed for encryption.

The input block X is set to the starting variable

$$X_1 = SV$$

The operation of decrypting each ciphertext variable employs the following four steps.

a) $Y_i = eK(X_i)$ (use of block cipher).

b) $E_i = j \sim Y_i$ (selection of leftmost j bits of Y_i).

c) $P_i = C_i \oplus E_i$ (generation of plaintext variable).

d) $X_{i+1} = Y_i$ (feedback operation).

These steps are repeated for $i = 1, 2, ..., q$, ending with step (c) on the last cycle. The procedure is shown on the right side of Figure C.4. The plaintext and ciphertext variables have bits numbered from 1 to j.

The result of each use of the block cipher is Y_i and this is fed back to become the next value of X, namely X_{i+1}. The leftmost j bits of Y_i are used to decrypt the input variable.

If a padding method is not in use, then P is equal to the concatenation of $P_1, P_2, ..., P_q$.

If a padding method is in use, apply the inverse of the agreed padding procedure to the sequence of q plaintext blocks $P_1, P_2, ..., P_q$ to obtain the decrypted plaintext string P.

NOTE Unique removal of padding is guaranteed because of the constraint on the choice of padding method given in Clause 5.

9.4 Avoiding ciphertext expansion

Using OFB mode, it is possible to avoid ciphertext expansion in the case where the plaintext is not a whole number of j-bit blocks. With this approach, if the plaintext is not a whole number of j-bit blocks, then the plaintext is padded with p ($0 \leq p < j$) padding bits so that the plaintext becomes a whole number of j-bit blocks but the padding is not included in the ciphertext. The actual value of the padding bits is irrelevant so they could be binary zeroes. The padded plaintext is encrypted as above but the final p bits of the final ciphertext block are discarded because they are not needed.

The decrypter then applies the same padding to produce a ciphertext that is a whole number of j-bit blocks, decrypts the ciphertext as above and then discards the final p bits of the plaintext.

NOTE 1 One can equally formulate this approach without reference to padding by simply requiring that the final encryption variable E_q be truncated to be the length of the final plaintext/ciphertext bits. Indeed, Annex B of the third edition of this document noted that "j can be modified for the last portion of the plaintext. No padding is required when a plaintext block with z bits, $z < j$ is encrypted by bitwise addition with only the first z bits of the corresponding variable E".

NOTE 2 If a padding method has been agreed as in Clause 5, then this approach is not used. If a padding method has not been agreed, then this approach is used.

10 Counter (CTR) mode

10.1 Preliminaries

The Counter mode of operation is defined by one parameter, i.e. the size of the plaintext variable j, where $1 \leq j \leq n$.

CTR mode encryption and decryption as defined in 10.2 and 10.3 shall use an agreed padding method if the bit length of the plaintext might not be a multiple of the parameter j. If no padding method is agreed, then the plaintext shall be a whole number of j-bit blocks unless the approach described in 10.4 is used.

The number of j-bit plaintext blocks (after padding if a padding method is in use) is q.

The variables employed by the Counter mode of operation when being used for encryption are

a) the input variables

1) A sequence of q plaintext variables $P_1, P_2, \ldots, P_q$, each of j bits.

2) A key K.

3) A starting variable SV of n bits. Refer to Annex B for security guidance related to the value of SV.

b) the intermediate results

1) A sequence of q block cipher input blocks $CTR_1, CTR_2, \ldots, CTR_q$, each of n bits.

2) A sequence of q block cipher output blocks $Y_1, Y_2, \ldots, Y_q$, each of n bits.

3) A sequence of q variables $E_1, E_2, \ldots, E_q$, each of j bits.

c) The output variables, i.e. a sequence of q ciphertext variables $C_1, C_2, \ldots, C_q$, each of j bits.

10.2 Encryption

If a padding method is to be used, then first pad the data string P to obtain a sequence of q plaintext blocks $P_1, P_2, ..., P_q$, each of j bits. Otherwise, simply divide the data string P into a sequence of q plaintext blocks $P_1, P_2, ..., P_q$, each of j bits.

The counter CTR is set to the starting variable

$$CTR_1 = SV$$

The operation of encrypting each plaintext variable employs the following four steps.

a) $Y_i = eK(CTR_i)$ (use of block cipher).

b) $E_i = j \sim Y_i$ (selection of leftmost j bits of Y_i).

c) $C_i = P_i \oplus E_i$ (generation of ciphertext variable).

d) $CTR_{i+1} = (CTR_i + 1) \bmod 2^n$ (generation of a new counter value CTR).

These steps are repeated for $i = 1, 2, ..., q$, ending with step (c) on the last cycle. The procedure is shown on the left side of Figure C.5. The plaintext and ciphertext variables have bits numbered from 1 to j.

The counter value is encrypted to give an output block Y_i and the leftmost j bits of this output block Y_i are used to encrypt the input value. The counter CTR then increases by one (modulo 2^n) to produce a new counter value.

10.3 Decryption

The variables employed for decryption are the same as those employed for encryption.

The counter CTR is set to the starting variable

$$CTR_1 = SV$$

The operation of decrypting each ciphertext variable employs the following four steps.

a) $Y_i = eK(CTR_i)$ (use of block cipher).

b) $E_i = j \sim Y_i$ (selection of leftmost j bits of Y_i).

c) $P_i = C_i \oplus E_i$ (generation of plaintext variable).

d) $CTR_{i+1} = (CTR_i + 1) \bmod 2^n$ (generation of a new counter value CTR).

These steps are repeated for $i = 1, 2, ..., q$, ending with step (c) on the last cycle. The procedure is shown on the right side of Figure C.5. The plaintext and ciphertext variables have bits numbered from 1 to j.

The counter value is encrypted to give an output block Y_i and the leftmost j bits of this output block Y_i are used to encrypt the input value. The counter CTR then increases by one (modulo 2^n) to produce a new counter value.

If a padding method is not in use, then P is equal to the concatenation of $P_1, P_2, ..., P_q$.

If a padding method is in use, apply the inverse of the agreed padding procedure to the sequence of q plaintext blocks $P_1, P_2, ..., P_q$ to obtain the decrypted plaintext string P.

NOTE Unique removal of padding is guaranteed because of the constraint on the choice of padding method given in Clause 5.

10.4 Avoiding ciphertext expansion

Using CTR mode it is possible to avoid ciphertext expansion in the case where the plaintext is not a whole number of j-bit blocks. With this approach, if the plaintext is not a whole number of j-bit blocks, then the plaintext is padded with p $(0 \leq p < j)$ padding bits so that the plaintext becomes a whole number of j-bit blocks but the padding is not included in the ciphertext. The actual value of the padding bits is irrelevant so they could be binary zeroes. The padded plaintext is encrypted as above but the final p bits of the final ciphertext block are discarded because they are not needed.

The decrypter then applies the same padding method to produce a ciphertext that is a whole number of j-bit blocks, decrypts the ciphertext as above and then discards the final p bits of the plaintext.

NOTE 1 One can equally formulate this approach without reference to padding by simply requiring that the final encryption variable E_q be truncated to be the length of the final plaintext/ciphertext bits. Indeed, Annex B of the third edition of this document noted that "j can be modified for the last portion of the plaintext. No padding is required when a plaintext block with z bits, $z < j$ is encrypted by bitwise addition with only the first z bits of the corresponding variable E".

NOTE 2 If a padding method has been agreed as in Clause 5, then this approach is not used. If a padding method has not been agreed, then this approach is used.

Annex A
(normative)

Object identifiers

This annex lists the object identifiers assigned to algorithms specified in this document.

```
ModesOfOperation {
iso(1) standard(0) modes-of-operation(10116) single-part(0)
asn1-module(0) algorithm-object-identifiers(0) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN
-- EXPORTS All; --
IMPORTS
BlockAlgorithms
FROM EncryptionAlgorithms-3 { iso(1) standard(0)
encryption-algorithms(18033) part3(3)
asn1-module(0) algorithm-object-identifiers(0) };
OID ::= OBJECT IDENTIFIER -- Alias
-- Synonyms --
is10116 OID ::= { iso(1) standard(0) modes-of-operation(10116) single-part(0)}
id-mode OID ::= { is10116 mode(1) }
id-pad OID ::= { is10116 pad(2) }
id-pad-null RELATIVE-OID ::= { 0 }       -- no padding algorithm identified
id-pad-1 RELATIVE-OID ::= { 1 }  -- padding according to method specified
-- in annex B, B.2.3
id-mode-ecb OID ::= { id-mode ecb(1) }
id-mode-cbc OID ::= { id-mode cbc(2) }
id-mode-cfb OID ::= { id-mode cfb(3) }
id-mode-ofb OID ::= { id-mode ofb(4) }
id-mode-ctr OID ::= { id-mode ctr(5) }
id-mode-cbc_cs1 OID ::= { id-mode cbc_cs1(6) }
id-mode-cbc_cs2 OID ::= { id-mode cbc_cs2(7) }
id-mode-cbc_cs3 OID ::= { id-mode cbc_cs3(8) }
-- Algorithm specifications --
ModeOfOperation ::= AlgorithmIdentifier {{ ModeOfOperationAlgorithms }}
ModeOfOperationAlgorithms ALGORITHM ::= {
{ OID id-mode-ecb PARMS EcbParameters } |
{ OID id-mode-cbc PARMS CbcParameters } |
{ OID id-mode-cfb PARMS CfbParameters } |
{ OID id-mode-ofb PARMS OfbParameters } |
{ OID id-mode-ctr PARMS CtrParameters }
{ OID id-mode-cbc_cs1 PARMS CbcParameters } |
{ OID id-mode-cbc_cs1 PARMS CbcParameters } |
{ OID id-mode-cbc_cs1 PARMS CbcParameters } ,
... -- expect additional algorithms –
}
PadAlgo ::= CHOICE {
specifiedPadAlgo     RELATIVE-OID,
generalPadAlgo       OID
}
EcbParameters ::= SEQUENCE {
bcAlgo BlockCipher  OPTIONAL,
padAlgo PadAlgo     DEFAULT specifiedPadAlgo:id-pad-null
}
CbcParameters ::= SEQUENCE {
m       INTEGER       DEFAULT 1,
bcAlgo        BlockCipher   OPTIONAL,
padAlgo       PadAlgo       DEFAULT specifiedPadAlgo:id-pad-1
}
CfbParameters ::= SEQUENCE {
r       INTEGER,      -- n<=r<=1024n where n is the cipher block length
k       INTEGER,      -- 1<=k<=n
j       INTEGER,      -- 1<=j<=k
bc      BlockCipher   OPTIONAL,
padAlgo       PadAlgo       DEFAULT specifiedPadAlgo:id-pad-null
}
OfbParameters ::= SEQUENCE {
j      INTEGER,      -- 1<=j<=n where n is the cipher block length
bc     BlockCipher   OPTIONAL,
padAlgo       PadAlgo       DEFAULT specifiedPadAlgo:id-pad-null
}
CtrParameters ::= SEQUENCE {
j      INTEGER,      -- 1<=j<=n where n is the cipher block length
bc     BlockCipher   OPTIONAL,
padAlgo       PadAlgo       DEFAULT specifiedPadAlgo:id-pad-null
}
-- Auxiliary definitions --
BlockCipher ::= AlgorithmIdentifier {{ BlockAlgorithms }}
-- 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 -- ModesOfOperation --
```

Annex B
(informative)

Properties of the modes of operation and important security guidance

B.1 General

As described above, these modes provide only protection of confidentiality. Proofs of security exist for the CBC mode, the CFB mode, the OFB mode and the CTR mode. The proofs assume that a block cipher cannot be distinguished from a pseudo-random function. The probability of this assumption being invalid increases dramatically as the number of processed blocks increases to $2^{n/2}$ and beyond. For this reason, the number of plaintext blocks encrypted under the same key should not exceed $2^{n/2}$ (see also ISO/IEC 29192-2:2012, Annex D and ISO/IEC JTC 1/SC 27, Standing Document 12 (SC 27 SD12) at http://www.jtc1sc27.din.de/sbe/SD12).

To simplify discussions in this annex, the modes in this document are treated as if they operate on a sequence of blocks of bits of length appropriate to the mode. That is, in this annex, the term ‘plaintext’ is used to refer to the data to be encrypted after any necessary padding has been applied.

In circumstances where padding is necessary, the padding method specified in Clause 5 is recommended. This padding method is not only simple to implement but it also resists certain attacks on CBC mode encryption [10], [11], [12] and [13]. For certain other padding methods, if an attacker is able to manipulate encrypted messages and is also able to detect whether or not an encrypted message causes a decrypting device to fail because the decryption process reveals an incorrectly formatted padded data string, then repeated probing of this type can be used to reveal information about the plaintext. If implementation of the mode uses padding, then verifying the authenticity of the ciphertext before decryption can prevent padding oracle attacks (see ISO/IEC 19772).

Implementations of this document can achieve authenticated encryption by use within Mechanism 5 (Encrypt-then-MAC) of ISO/IEC 19772:2009 and using a MAC algorithm from ISO/IEC 9797-1:2011.

B.2 Electronic Codebook (ECB) mode of operation

B.2.1 Properties and security guidance

Properties of the ECB mode are:

a) encryption or decryption of a block can be carried out independently of the other blocks;

b) reordering of the ciphertext blocks will result in the corresponding reordering of the plaintext blocks;

c) the same plaintext block always produces the same ciphertext block (for the same key) making it vulnerable to a “dictionary attack”, where a dictionary is built up containing corresponding plaintext and ciphertext blocks.

Binary data exchanged between computers, or people, may contain repetitions or commonly used sequences. In ECB mode, identical plaintext blocks produce (for the same key) identical ciphertext blocks.

For this reason, the ECB mode is not recommended for messages longer than one block. The use of ECB may only be specified in future International Standards for those special purposes where the repetition characteristic is acceptable (for example if input blocks never repeat), blocks have to be accessed individually or blocks have to be accessed randomly.

B.2.2 Error propagation

In the ECB mode, one or more bit errors within a single ciphertext block will only affect the decryption of the block in which the error(s) occur(s). Decryption of a ciphertext block with one or more error bits will result in an expected 50 % error probability for each plaintext bit in the corresponding plaintext block.

B.2.3 Synchronization

If block boundaries are lost between encryption and decryption (e.g. due to loss or insertion of a ciphertext bit), synchronization between the encryption and decryption operations will be lost until the correct block boundaries are re-established. The result of all decryption operations will be incorrect while the block boundaries are lost.

B.3 Cipher Block Chaining (CBC) mode of operation

B.3.1 Properties and security guidance

Properties of the CBC mode are:

a) the chaining operation makes the ciphertext blocks dependent on the current and the preceding plaintext blocks P_{i-m}, P_{i-2m}, P_{i-3m}, . . . and therefore rearranging ciphertext blocks does not result in a rearranging of the corresponding plaintext blocks; consequently, an adversary is not able to permute plaintext blocks simply by permuting ciphertext blocks;

b) the use of different *SV* values prevents the same plaintext encrypting to the same ciphertext;

c) selection of $m > 1$ enables the block cipher encryption operations to be performed in parallel. With use of parallel processing hardware such as a pipeline-oriented circuit, it facilitates high-throughput implementations;

d) decryption of any ciphertext block is possible without decrypting any of the preceding sequence of ciphertext blocks; that is, this mode has the "random access" property for ciphertext;

e) if the block cipher can be modelled by a pseudo-random permutation, and the *SV* has been randomly chosen, the CBC mode is mathematically proven to be secure in the sense that the encryption leaks no computationally non-trivial information about the plaintext (see Reference [9]).

NOTE It is a common misunderstanding that CBC mode provides data integrity — it does not.

The CBC mode produces the same ciphertext whenever the same plaintext is encrypted using the same key and starting variable.

If multiple plaintexts are encrypted under the same key and if none of the following procedures to generate the *SV* are used, security cannot be guaranteed.

— Generate the *SV* uniformly at random from the set of all n-bit strings with an unpredictable source of randomness (see ISO/IEC 18031).

— Set the first plaintext block to be a unique identifier for each plaintext out of the adversary's control, such as an incremented counter and set *SV* to the n-bit string consisting of only zeros.

B.3.2 Error propagation

In the CBC mode, one or more bit errors within a single ciphertext block will affect the decryption of two blocks (the block in which the error occurs and the m-th block after). An error in the i-th ciphertext block has the following effect on the resulting plaintext: the i-th plaintext block will have an expected 50 % error probability for each bit. The $(i+m)$-th plaintext block will have an error pattern equal to that in the i-th ciphertext block.

B.3.3 Synchronization

If block boundaries are lost between encryption and decryption (e.g. due to loss or insertion of a ciphertext bit), synchronization between the encryption and decryption operations will be lost until the correct block boundaries are re-established. The result of all decryption operations will be incorrect while the block boundaries are lost.

B.4 Cipher Feedback (CFB) mode of operation

B.4.1 Properties and security guidance

Properties of the CFB mode are:

a) the chaining operation makes the ciphertext variables dependent on the current and all but a certain number of immediately preceding plaintext variables. This number depends on the selection of r, k and j. Therefore, rearranging j-bit ciphertext variables does not result in a rearranging of the corresponding j-bit plaintext variables; consequently, an adversary is not able to permute plaintext blocks simply by permuting ciphertext blocks;

b) the use of different *SV* values prevents the same plaintext encrypting to the same ciphertext;

c) the encryption and decryption processes in the CFB mode both use only the encryption operation of the block cipher;

d) the strength of the CFB mode depends on the size of k (maximal if $j = k = n$) and the relative sizes of j, k, n and r;

NOTE A choice of $j < k$ will result in an increased probability of repeating occurrences of values of the input blocks. Such repeated occurrences will reveal linear relations between plaintext bits.

e) use of this mode will require approximately n/j times as many block cipher encryption operations than would ECB mode, and hence, selection of a small value for j will cause greater processing overheads;

f) selection of $r \geq n+k$ enables the pipelining and the continuous operation of the block cipher (assuming the use of parallel processing hardware, such as a pipeline-oriented circuit); the initial contents (*SV*) of the feedback buffer will only be completely replaced when the number of block cipher operations performed is greater than or equal to r / k; and

g) a security proof for CFB mode is given in Reference [8].

The CFB mode produces the same ciphertext whenever the same plaintext is encrypted using the same key and starting variable.

If multiple plaintexts are encrypted using the same key, then security cannot be guaranteed if the *SV* is not random for each plaintext, i.e. drawn uniformly at random from the set of all r bit strings (see ISO/IEC 18031).

B.4.2 Error propagation

In the CFB mode, errors in any j-bit unit of ciphertext will affect the decryption of succeeding ciphertext until the bits in error have been shifted out of the CFB feedback buffer. An error in the i-th ciphertext variable has the following effect on the resulting plaintext: the i-th plaintext variable will have an error pattern equal to that in the i-th ciphertext variable. The succeeding plaintext variables will have an expected 50 % error probability for each bit until all incorrectly received bits have been shifted out of the feedback buffer.

B.4.3 Synchronization

If j-bit boundaries are lost between encryption and decryption (e.g. due to loss or insertion of a ciphertext bit), cryptographic synchronization will be re-established r bits after j-bit boundaries are re-established. If a multiple of j bits are lost synchronization will be re-established automatically after r bits. Consequently, when $j = k = 1$, the CFB mode automatically re-establishes cryptographic synchronization after the loss or insertion of any number of ciphertext bits.

B.5 Output Feedback (OFB) mode of operation

B.5.1 Properties and security guidance

Properties of the OFB mode are:

a) the use of different *SV* values prevents the same plaintext encrypting to the same ciphertext, by producing different keystreams;

b) the encryption and decryption processes in the OFB mode both use only the encryption operation of the block cipher;

c) the OFB mode does not depend on the plaintext to generate the keystream used to add modulo 2 to the plaintext;

d) use of this mode will require approximately n/j times as many block cipher encryption operations than would ECB mode, and hence, selection of a small value for j will cause greater processing overheads; and

e) security proofs for the OFB mode are analogous to those for the CTR mode.

The OFB mode produces the same ciphertext whenever the same plaintext is encrypted using the same key and starting variable. Moreover, in the OFB mode, the same keystream (the sequence of intermediate results E_i) is produced when the same key and *SV* are used. Consequently, for security reasons a specific *SV* should be used only once for a given key.

If multiple plaintexts are encrypted using the same key and if none of the following procedures to generate the *SV* are used, security cannot be guaranteed.

— Generate the *SV* uniformly at random from the set of all n-bit strings with an unpredictable source of randomness (see ISO/IEC 18031).

— Set the *SV* to a unique identifier out of the adversary's control, such as an incremented counter.

B.5.2 Error propagation

The OFB mode does not extend ciphertext errors in the resultant plaintext output. Every bit in error in the ciphertext causes only one bit to be in error in the decrypted plaintext.

B.5.3 Synchronization

The OFB mode is not self-synchronizing. If the two operations of encryption and decryption get out of synchronization, the system needs to be re-initialized. Such a loss of synchronism might be caused by any number of inserted or lost ciphertext bits.

Each re-initialization should use a value of *SV* different from the *SV* values used before with the same key. The reason for this is that an identical bit stream would be produced each time for the same parameters. This would be susceptible to, for example, known plaintext and known ciphertext attacks.

B.6 Counter (CTR) mode of operation

B.6.1 Properties and security guidance

Properties of the Counter mode are:

a) the use of different *SV* values prevents the same plaintext encrypting to the same ciphertext by producing different keystreams;

b) the encryption and decryption processes in the Counter mode both use only the encryption operation of the block cipher;

c) the Counter mode does not depend on the plaintext to generate the keystream used to add modulo 2 to the plaintext;

d) in the Counter Mode, a ciphertext block C_i can be decrypted without decrypting the ciphertext block C_{i-1}; this is known as a random-access property;

e) selection of a small value of *j* will require approximately *n*/*j* times of cycles compared to ECB mode and thus cause greater processing overhead; and

f) under the same assumptions about the block cipher as in B.2.2, the Counter mode can also be proven to be secure [9].

The Counter mode produces the same ciphertext whenever the same plaintext is encrypted using the same key and starting variable *SV*. Moreover, in the Counter mode the same keystream (the sequence of intermediate results E_i) is produced when the same key and *SV* are used. Consequently, for security reasons, a specific *SV* should be used only once for a given key and the *SV* values should be selected so that any intermediate *CTR* value is not used more than once for a given key.

If multiple plaintexts are encrypted under the same key, then security can only be guaranteed if *SV*s are used only once and are chosen so that any intermediate *CTR* value is not used more than once. A uniformly randomly chosen *SV* (i.e. drawn uniformly at random from the set of all *n*-bit strings; see ISO/IEC 18031) is also sufficient for security but only in so far as collisions in the *SV*s and intermediate *CTR* values do not occur.

B.6.2 Error propagation

The Counter mode does not extend ciphertext errors in the resultant plaintext output. Every bit in error in the ciphertext causes only one bit to be in error in the decrypted plaintext.

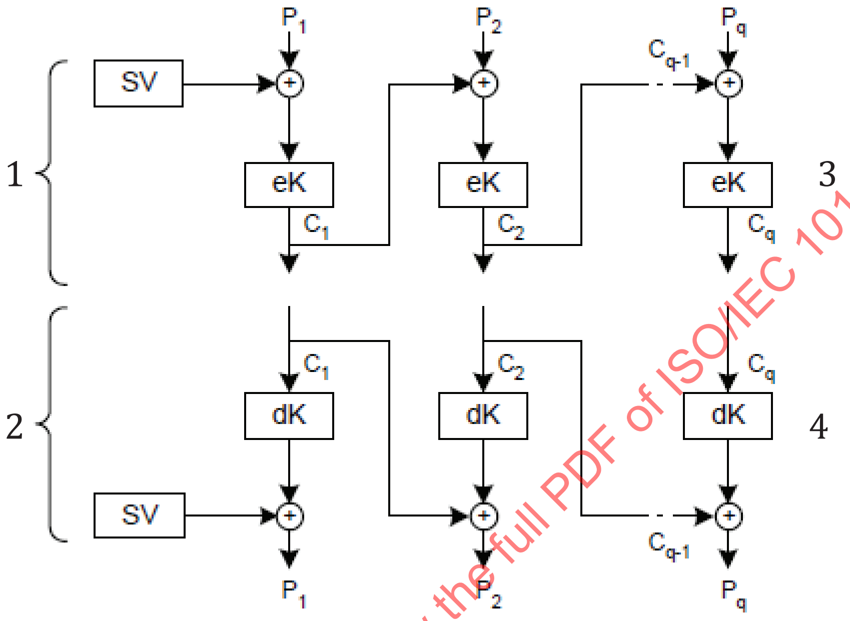
B.6.3 Synchronization

The Counter mode is not self-synchronizing. If the two operations of encryption and decryption get out of synchronization, the system needs to be re-initialized. Such a loss of synchronism might be caused by any number of inserted or lost ciphertext bits.

Each re-initialization should use a value of *SV* different from the *SV* values used before with the same key and should ensure that counter values are not re-used. The reason for this, as noted above, is that an identical bit stream would be produced each time for the same parameters. This would be susceptible to, for example, known plaintext and known ciphertext attacks.

Annex C
(informative)

Figures describing the modes of operation



Key

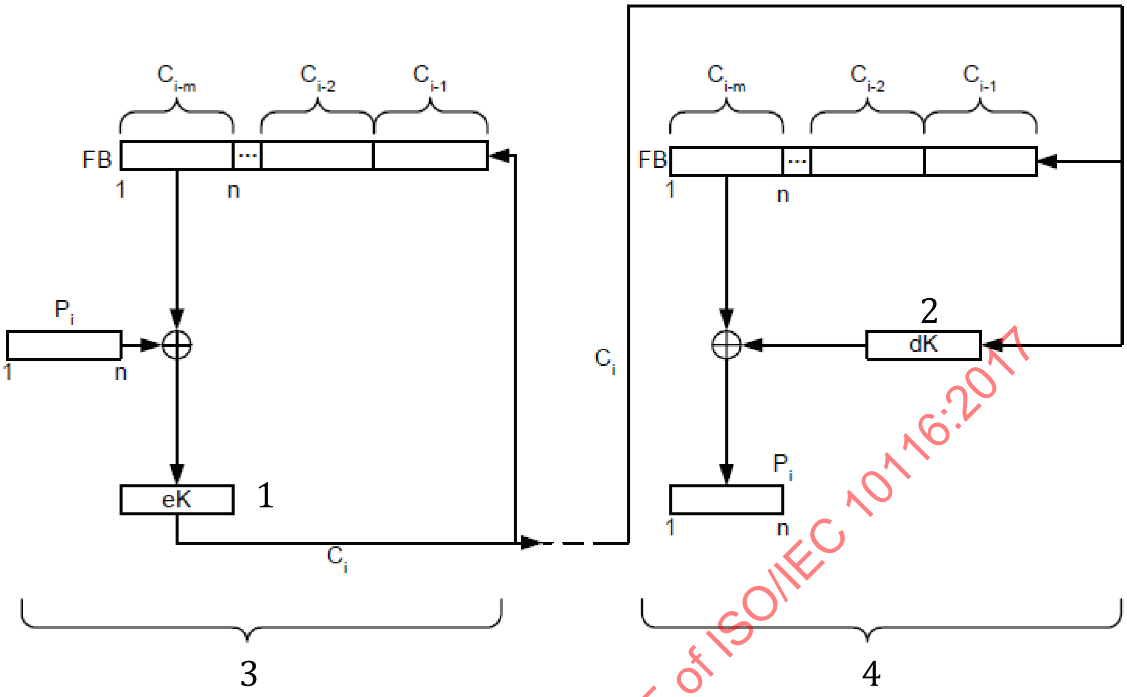
1 encryption

2 decryption

3 encryption algorithm

4 decryption algorithm

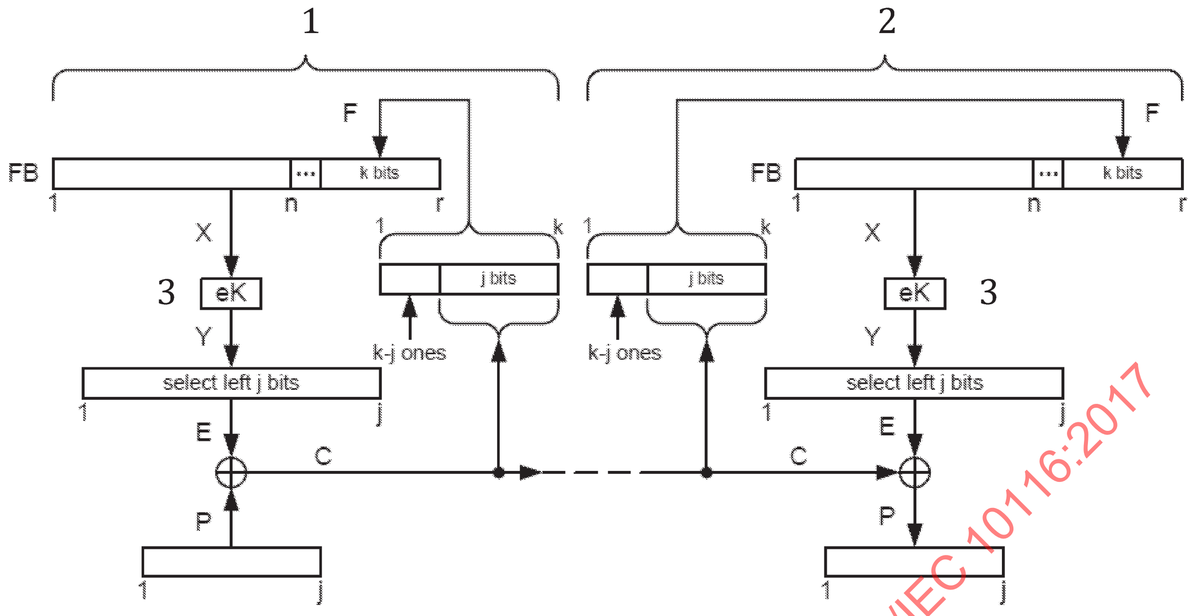
Figure C.1 — Cipher Block Chaining (CBC) mode of operation with *m* = 1



Key

1 encryption algorithm
2 decryption algorithm
3 encryption
4 decryption

Figure C.2 — Cipher Block Chaining (CBC) mode of operation



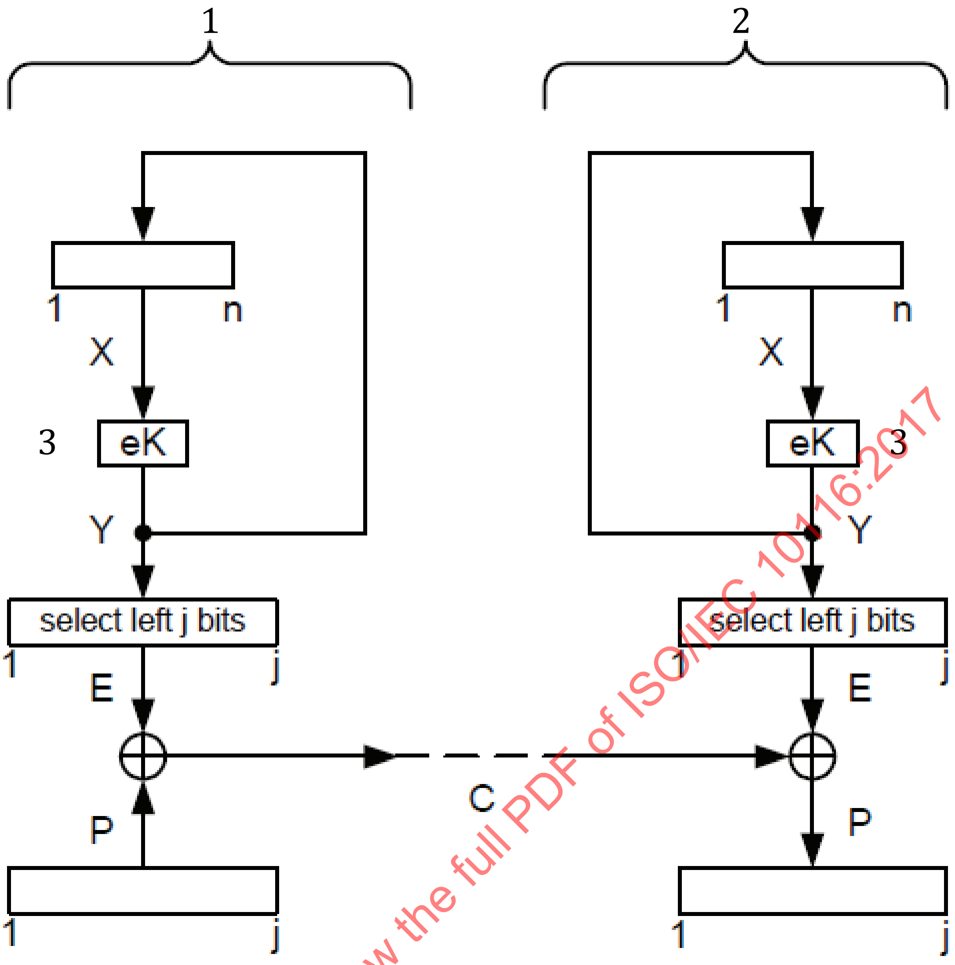
Key

1 encryption

2 decryption

3 encryption algorithm

Figure C.3 — Cipher Feedback (CFB) mode of operation



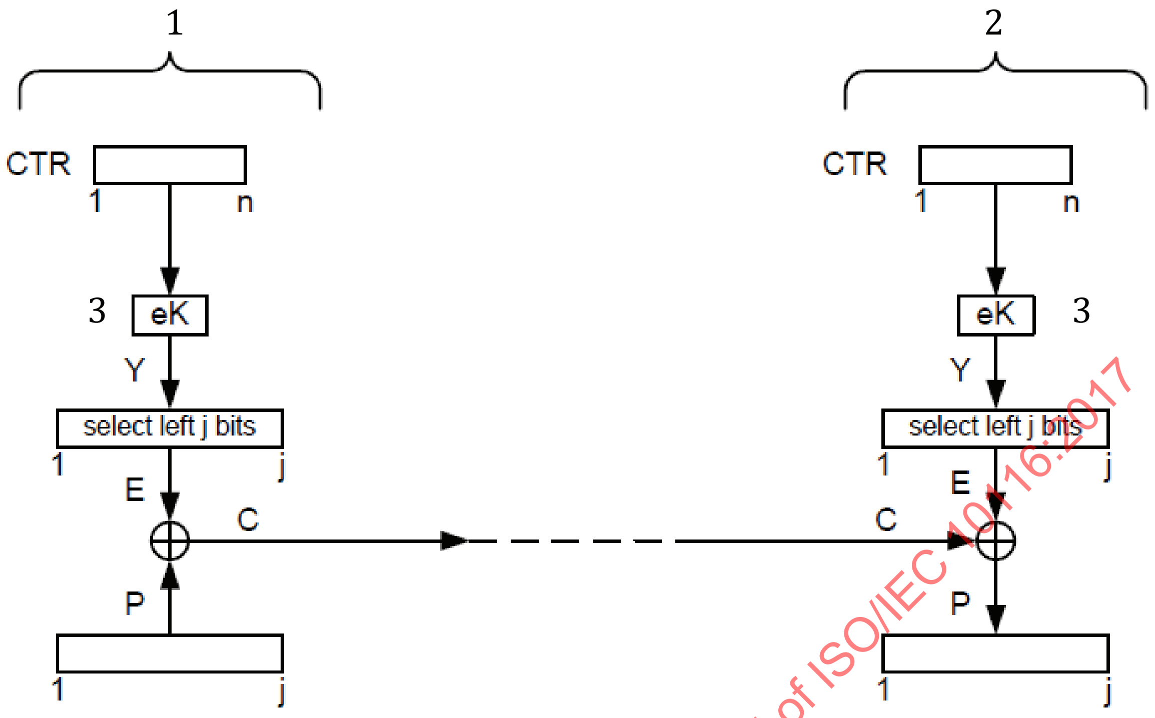
Key

1 encryption

2 decryption

3 encryption algorithm

Figure C.4 — Output Feedback (OFB) mode of operation



Key

1 encryption

2 decryption

3 encryption algorithm

Figure C.5 — Counter (CTR) mode of operation

Annex D
(informative)

Numerical examples for the modes of operation

D.1 General

This annex gives examples for the encryption and decryption of a message using the modes of operation specified in this document. The block cipher used in D.2 is the Triple Data Encryption Algorithm (TDEA) with the value of $n = 64$. D.3 gives examples for the Advanced Encryption Standard (AES) with $n = 128$. The TDEA and the AES are standardized in ISO/IEC 18033-3.

D.2 Triple Data Encryption Algorithm (TDEA)

D.2.1 General

The examples use the following parameters:

a) The cryptographic keys are:

K_1 = 0123456789ABCDEF;

K_2 = 23456789ABCDEF01;

K_3 = 456789ABCDEF0123.

b) The starting variable is 1234567890ABCDEF.

c) The plaintext is the 7-bit ASCII code for “Now is the time for all” (in hexadecimal notation 4E6F772069732074 68652074696D6520 666F7220616C6C20). For CFB mode, the plaintext is the 7-bit ASCII code for “Now is the” (in hexadecimal notation 4E6F77206973207468 65).

The input and output of the DEA functional blocks are given sequentially. For example, in the following table, the three steps are finished sequentially in three clock cycles. If the output of DEA$_1$ is 3FA40E8A984D4815, then it is the input of DEA$_2$, and the output of DEA$_2$ is 0663D1B37C48090C, which is the input of DEA$_3$. The output of TDEA encryption is the output of DEA$_3$.

D.2.2 ECB Mode

D.2.2.1 ECB Mode, encryption

Examples for the ECB mode of encryption are as follows.

P_1 = 4E6F772069732074

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	4E6F772069732074	3FA40E8A984D4815
DEA$_2$ – dK_2	3FA40E8A984D4815	0663D1B37C48090C
DEA$_3$ – eK_3	0663D1B37C48090C	314F8327FA7A09A8

C_1 = `314F8327FA7A09A8`

P_2 = `68652074696D6520`

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	68652074696D6520	6A271787AB8883F9
DEA$_2$ – dK_2	6A271787AB8883F9	95CAE06A9FF4D6D2
DEA$_3$ – eK_3	95CAE06A9FF4D6D2	4362760CC13BA7DA

C_2 = `4362760CC13BA7DA`

P_3 = `666F7220616C6C20`

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	666F7220616C6C20	893D51EC4B563B53
DEA$_2$ – dK_2	893D51EC4B563B53	5FED3EB05419F67C
DEA$_3$ – eK_3	5FED3EB05419F67C	FF55C5F80FAAAC45

C_3 = `FF55C5F80FAAAC45`

D.2.2.2 ECB Mode, decryption

Examples for the ECB mode of decryption are as follows.

C_1 = `314F8327FA7A09A8`

	block cipher input block	block cipher output block
DEA$_3$ – dK_3	314F8327FA7A09A8	0663D1B37C48090C
DEA$_2$ – eK_2	0663D1B37C48090C	3FA40E8A984D4815
DEA$_1$ – dK_1	3FA40E8A984D4815	4E6F772069732074

P_1 = `4E6F772069732074`

C_2 = `4362760CC13BA7DA`

	block cipher input block	block cipher output block
DEA$_3$ – dK_3	4362760CC13BA7DA	95CAE06A9FF4D6D2
DEA$_2$ – eK_2	95CAE06A9FF4D6D2	6A271787AB8883F9
DEA$_1$ – dK_1	6A271787AB8883F9	68652074696D6520

P_2 = `68652074696D6520`

C_3 = `FF55C5F80FAAAC45`

	block cipher input block	block cipher output block
DEA$_3$ – dK_3	FF55C5F80FAAAC45	5FED3EB05419F67C
DEA$_2$ – eK_2	5FED3EB05419F67C	893D51EC4B563B53
DEA$_1$ – dK_1	893D51EC4B563B53	666F7220616C6C20

P_3 = `666F7220616C6C20`

D.2.3 CBC Mode

D.2.3.1 CBC mode, encryption

Examples for the CBC mode of encryption with $m = 1$ are as follows.

SV = 1234567890ABCDEF = C_0

$P_1 \oplus C_0$ = 5C5B2158F9D8ED9B

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	5C5B2158F9D8ED9B	E5C7CDDE872BF27C
DEA$_2$ – dK_2	E5C7CDDE872BF27C	EDFA50E493DBFECC
DEA$_3$ – eK_3	EDFA50E493DBFECC	F3C0FF026C023089

C_1 = F3C0FF026C023089

$P_2 \oplus C_1$ = 9BA5DF76056F55A9

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	9BA5DF76056F55A9	5A80BE0F562EE625
DEA$_2$ – dK_2	5A80BE0F562EE625	B92F8D3BB74CE43D
DEA$_3$ – eK_3	B92F8D3BB74CE43D	656FBB169DEF7EDB

C_2 = 656FBB169DEF7EDB

$P_3 \oplus C_2$ = 0300C936FC8312FB

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	0300C936FC8312FB	F1F58BA6135C2B1E
DEA$_2$ – dK_2	F1F58BA6135C2B1E	3374E4F8B29A3026
DEA$_3$ – eK_3	3374E4F8B29A3026	30BA36075D6F0176

C_3 = 30BA36075D6F0176

D.2.3.2 CBC mode, decryption

Examples for the CBC mode of decryption with $m = 1$ are as follows.

C_1 = F3C0FF026C023089

	block cipher input block	block cipher output block
DEA$_3$ – dK_3	F3C0FF026C023089	EDFA50E493DBFECC
DEA$_2$ – eK_2	EDFA50E493DBFECC	E5C7CDDE872BF27C
DEA$_1$ – dK_1	E5C7CDDE872BF27C	5C5B2158F9D8ED9B

$P_1 = dK_1(eK_2(dK_3(C_1))) \oplus C_0$ = 4E6F772069732074

C_2 = 656FBB169DEF7EDB

	block cipher input block	block cipher output block
DEA$_3$ – dK_3	656FBB169DEF7EDB	B92F8D3BB74CE43D
DEA$_2$ – eK_2	B92F8D3BB74CE43D	5A80BE0F562EE625
DEA$_1$ – dK_1	5A80BE0F562EE625	9BA5DF76056F55A9

$P_2 = dK_1(eK_2(dK_3(C_2))) \oplus C_1$ = 68652074696D6520

C_3 = 30BA36075D6F0176

	block cipher input block	block cipher output block
DEA$_3$ – dK_3	30BA36075D6F0176	3374E4F8B29A3026
DEA$_2$ – eK_2	3374E4F8B29A3026	F1F58BA6135C2B1E
DEA$_1$ – dK_1	F1F58BA6135C2B1E	0300C936FC8312FB

$P_3 = dK_1(eK_2(dK_3(C_3))) \oplus C_2$ = 666F7220616C6C20

D.2.4 CFB Mode

D.2.4.1 CFB Mode, encryption

Examples for the CFB mode of encryption are as follows. For this example, the parameters $j = k = 8$ and $r = n$ have been chosen.

SV = 1234567890ABCDEF

P_1 = 4E

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	1234567890ABCDEF	BD661569AE874E25
DEA$_2$ – dK_2	BD661569AE874E25	553A74ED589EC819
DEA$_3$ – eK_3	553A74ED589EC819	A011B07C73633375

$C_1 = P_1 \oplus E_1$ = 4E $\oplus$ A0 = EE

P_2 = 6F

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	34567890ABCDEFEE	A99C77AD5C012000
DEA$_2$ – dK_2	A99C77AD5C012000	67AC5DE460707687
DEA$_3$ – eK_3	67AC5DE460707687	F4FD50973F59A489

$C_2 = P_2 \oplus E_2$ = 6F $\oplus$ F4 = 9B

P_3 = 77

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	567890ABCDEFEE9B	05A76EE816060C18
DEA$_2$ – dK_2	05A76EE816060C18	E30ACAF870F4ED25
DEA$_3$ – eK_3	E30ACAF870F4ED25	73BFA8827618CC1C

$C_3 = P_3 \oplus E_3$ = 77 $\oplus$ 73 = 04

P_4 = 20

	block cipher input block	block cipher output block
DEA$_1$ – eK_1	7890ABCDEFEE9B04	9CB2ADF4743DE3A6
DEA$_2$ – dK_2	9CB2ADF4743DE3A6	1B4FD9CE1AF12640
DEA$_3$ – eK_3	1B4FD9CE1AF12640	DF97078C6539370F

$C_4 = P_4 \oplus E_4$ = 20 $\oplus$ DF = FF