ISO/IEC 3rd CD 10116

Information technology —
Security techniques —

Modes of operation for an n-bit block cipher

4.5 selection of bits:

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 is defined only when $1 \leq j \leq m$.

4.6 shift operation:

A "shift function" S_k is defined as follows: Given an m-bit variable X and a k-bit variable F where $1 \leq k \leq m$, the effect of the shift function $S_k(X \mid F)$ is to produce the m-bit variable

$$S_k(X \mid F) = (x_{k+1}, x_{k+2}, \ldots, x_m, f_1, f_2, \ldots, f_k) \qquad (k < m)$$

$$S_k(X \mid F) = (f_1, f_2, \ldots, f_k) \qquad (k = m)$$

The effect is to shift the bits of array X left by k places, discarding $x_1, x_2, \ldots, x_k$, and to place the array F in the rightmost k places of X. When $k = m$ the effect is to totally replace X by F.

4.7 *I*(*k*):

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

Electronic Codebook (ECB) Mode

The input variables
1) A sequence of *q* plaintext blocks $P_1, P_2, \ldots, P_q$, each of *n* bits.
2) A key *K*.

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

Encipherment
The ECB mode of encipherment operates as follows:
$Ci = e_K(P_i)$ for $i = 1, 2, \ldots, q$.

Decipherment (*2)
The ECB mode of decipherment operates as follows:
$Pi = d_K(C_i)$ for $i = 1, 2, \ldots, q$.

Properties of the ECB mode

a) encipherment or decipherment 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 with corresponding plaintext and ciphertext blocks.

Padding requirements

Only multiples of *n* bits can be enciphered or deciphered. Other lengths need to be padded to a *n*-bit boundary.

Error propagation

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

Block boundaries

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

Cipher Block Chaining (CBC) Mode

The one parameter, i.e. the number m of stored ciphertext blocks, where $m \leq 1024$.

The input variables
1) A sequence of q plaintext blocks $P_1, P_2, \ldots, P_q$, each of n bits.
2) A key K.
3) A sequence of m starting variables $SV_1, SV_2, \ldots, SV_m$ each of n bits.

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

Encipherment (*1)

The CBC mode of encipherment operates as follows:

$$C_i = e_K(P_i \oplus SV_i), \quad 1 \leq i \leq m$$

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

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

Decipherment

The CBC mode of decipherment operates as follows:

$$P_i = d_K(C_i) \oplus SV_i, \quad 1 \leq i \leq m$$

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

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

Properties of the CBC mode

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, however, rearranging a portion of ciphertext blocks results in the rearrangement of the corresponding plaintext blocks after a sequence of corrupt plaintext blocks;
b) the use of different SV values prevents the same plaintext enciphering to the same ciphertext;
c) selection of $m > 1$ enables the block cipher encryption operations to be performed in parallel to facilitate 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) assuming the block cipher is ideal, i.e. modeled by a pseudo random function and the SV has been randomly chosen, the CBC mode is mathematically proven to be secure in the sense that the encipherment leaks no computationally non-trivial information about the plaintext.

Padding requirements

Only multiples of *n* bits can be enciphered or deciphered. Other lengths need to be padded to a *n*-bit boundary. If this is not acceptable, the last variable can be treated in a special way. Two examples of a special treatment are given below.

A first possibility to treat an incomplete last variable (i.e. a variable P_q of $j < n$ bits where q should be greater than 1) is to encipher it in OFB mode as described below:
a) encipherment

$$C_q = P_q \oplus (j \sim eK(C_{q-1}))$$

b) decipherment

$$P_q = C_q \oplus (j \sim eK(C_{q-1}))$$

However, this last variable is vulnerable to a "chosen plaintext attack" if the SV is not secret or if it is used more than once with the same key.

A second possibility is known as "ciphertext-stealing". Suppose that the last two plaintext variables are P_{q-1} and P_q, where P_{q-1} is an n-bit block and P_q is a variable of $j < n$ bits and q should be greater than 1.

a) encipherment
Let C_{q-1} be the ciphertext block derived from P_{q-1} using the method described in (*1).
Then set $C_q = e_K(S_j(C_{q-1} \mid P_q))$ The last two ciphertext variables are then $j \sim C_{q-1}$ and C_q.

b) decipherment
C_q needs to be deciphered first, resulting in the variable P_q and the right-most n - j bits of C_{q-1}

$$S_j(C_{q-1} \mid P_q) = d_K(C_q)$$

The complete block C_{q-1} is now available and P_{q-1} can be derived using the method described in ECB (*2).
The two trailing ciphertext variables are deciphered in reverse order which makes this solution less suited for hardware implementations.

Ciphertext stealing may leak information about the last plaintext of a message under a given key. When the "stolen" ciphertext bits are identical in two ciphertexts, which can easily occur if *n*-j is small, then identical final plaintext blocks will produce identical final ciphertext blocks.

Error propagation

In the CBC mode, one or more bit errors within a single ciphertext block will affect the decipherment 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. If errors occur in a variable of less than *n* bits, error propagation depends on the chosen method of special treatment. In the first example the deciphered short block will have those bits in error that correspond directly to the ciphertext bits in error.

Synchronization

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

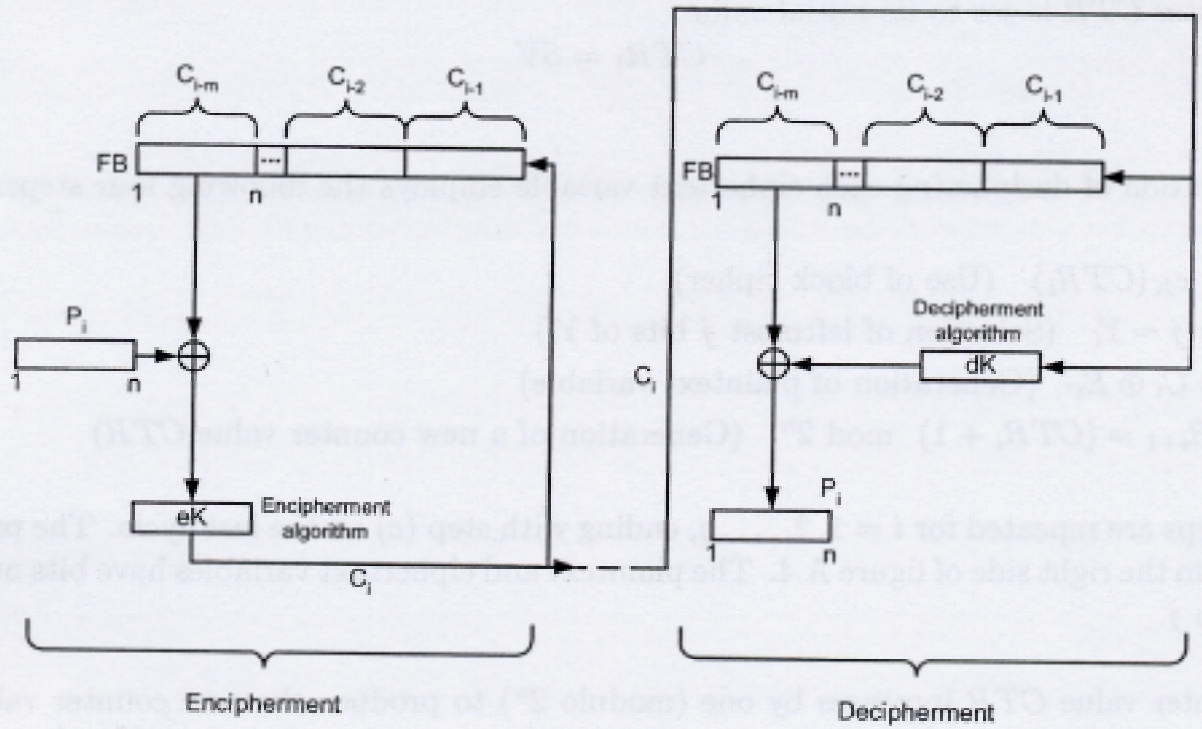

Figure A.1 – The cipher block chaining (CBC) mode of operation

Cipher Feedback (CFB) Mode

Three parameters define a CFB mode of operation:
— the size of feedback buffer, r, where $n \leq r \leq 1024n$
— the size of feedback variable, k, where $1 \leq k \leq n$
— the size of plaintext variable, j, where $1 \leq j \leq k$

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 r bits.

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 - 1 feedback buffer contents $FB_1, FB_2, \ldots, FB_{q-1}$ each of r bits.

The output variables, i.e. a sequence of q ciphertext C_1, C_2, C_q, each of j bits.

Encipherment

The feedback buffer FB is set to its initial value

$FB_1 = SV$

The operation of enciphering each plaintext variable employs the following six steps:

a) $X_i = n \sim FB_i$ (Selection of leftmost *n* bits of FB)

b) $Y_i = e_K(X_i)$ (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) $F_i = S_j(I(k) \mid C_i)$ (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.

Decipherment

The variables employed for decipherment are the same as those employed for encipherment.

The feedback buffer FB is set to its initial value

$FB_1 = SV$

The operation of deciphering each ciphertext variable employs the following six steps:

a) $X_i = n \sim FB_i$ (Selection of leftmost *n* bits of FB)

b) $Y_i = e_K(X_i)$ (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) $F_i = S_j(I(k) \mid C_i)$ (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.

Properties of the CFB mode

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.
b) the use of different SV values prevents the same plaintext enciphering to the same ciphertext;
c) the encipherment and decipherment processes in the CFB mode both use only the encipherment 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;

Padding requirements

Only multiples of j bits can be enciphered or deciphered. Other lengths need to be padded to a j-bit boundary. However, usually j will be chosen equal to such a size that no padding will be required, e.g. j can be modified for the last portion of the plaintext.

Error propagation

In the CFB mode, errors in any j-bit unit of ciphertext will affect the decipherment 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.

Synchronization

If j-bit boundaries are lost between encipherment and decipherment (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 bi

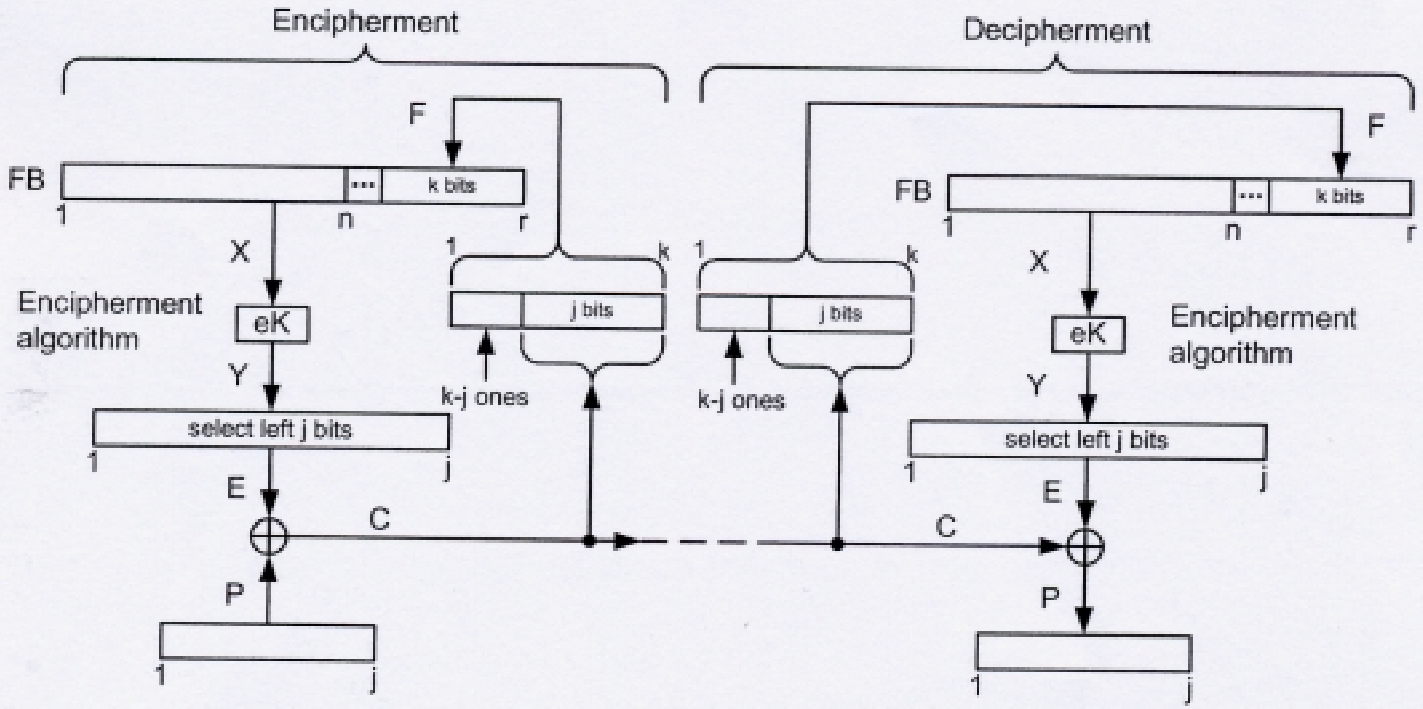

Figure A.2 – The cipher feedback (CFB) mode of operation

Output Feedback (OFB) Mode

The one parameter, i.e. the size of the plaintext variable j, where $1 \leq j \leq n$.

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.

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.

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

Encipherment

The input block X is set to its initial value

$$X_1 = SV$$

The operation of enciphering each plaintext variable employs the following four steps:

a) $Y_i = e_K(X_i)$ (Use of block cipher)
b) $E_i = j \sim Y_i$ (Selection of leftmost j bits)
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.

Decipherment

The variables employed for decipherment are the same as those employed for encipherment. The input block X is set to its initial value

$X_1 = SV$

The operation of deciphering each ciphertext variable employs the following five steps:

a) $Y_i = e_K(X_i)$ (Use of block cipher)
b) $E_i = j \sim Y_i$ (Selection of leftmost *j* bits)
c) $P_i = C_i \oplus E_i$ (Generation of plaintext variable)
d) $X_{i+1} = Y_i$ (Shift function on FB)

These steps are repeated for i = 1,2, ... ,*q*, ending with step (c) on the last cycle.

Properties of the OFB mode

a) the use of different SV values prevents the same plaintext enciphering to the same ciphertext, by producing different key streams;
b) the encipherment and decipherment processes in the OFB mode both use only the encipherment operation of the block cipher;
c) the OFB mode does not depend on the plaintext to generate the key stream 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.

Padding requirements

Only multiples of j bits can be enciphered or deciphered. Other lengths need to be padded to a j-bit boundary. However, usually j will be chosen equal to such a size that no padding will be required, e.g. j can be modified for the last portion of the plaintext. There is no padding 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.

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 deciphered plaintext.

Synchronization

The OFB mode is not self-synchronizing. If the two operations of encipherment and decipherment 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 for example to known plaintext and known ciphertext attacks.

Also overlapping sequences of block cipher input blocks while using the same key should be prevented as these make the OFB mode vulnerable to, for example, “known plaintext” and “known ciphertext” attacks.

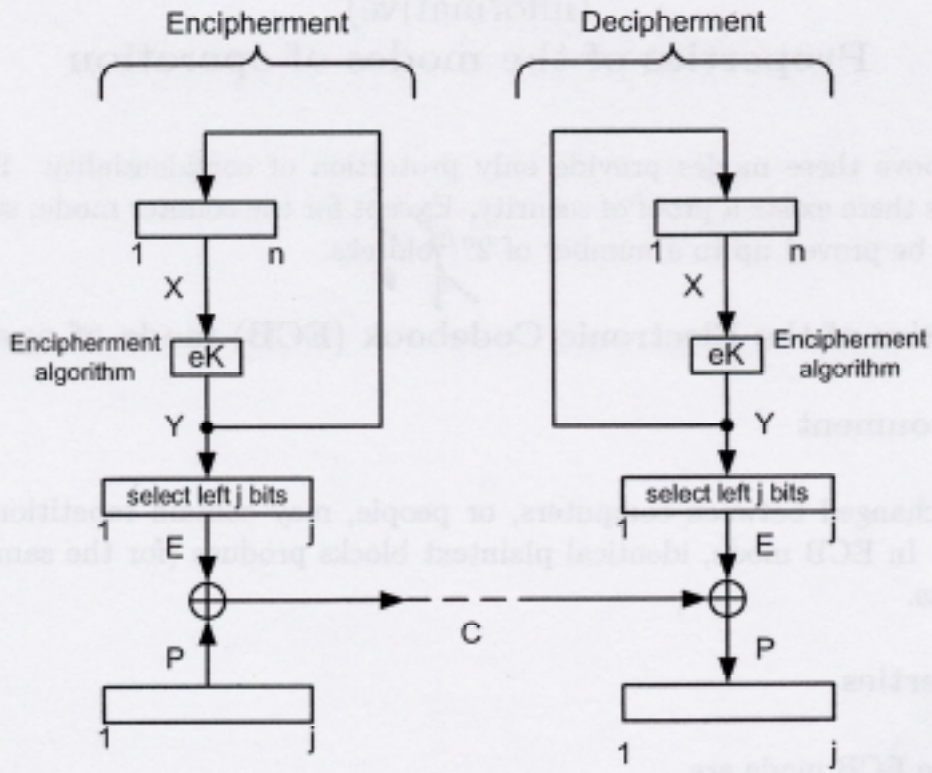

Figure A.3 – The output feedback (OFB) mode of operation

Counter (CTR) Mode

The one parameter, i.e. the size of plaintext variable, j, where $1 \leq j \leq n$.

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.

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.

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

Encipherment

The counter CTR is set to its initial value

$$CTR_1 = SV$$

The operation of enciphering each plaintext variable employs the following four steps:

a) $Y_i = e_K(CTR_i)$ (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.

Decipherment

The variables employed for decipherment are the same as those employed for encipherment.
The counter CTR is set to its initial value

$CTR_1 = SV$

The operation of deciphering each ciphertext variable employs the following four steps:

a) $Y_i = e_K(CTR_i)$ (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.

Properties of the Counter Mode

a) the use of different SV values prevents the same plaintext enciphering to the same ciphertext, by producing different key streams;
b) the encipherment and decipherment processes in the Counter Mode both use only the encipherment operation of the block cipher;
c) the Counter Mode does not depend on the plaintext to generate the key stream used to add modulo 2 to the plaintext;
d) selection of a small value of *j* will require approximately $\frac{n}{j}$ times of cycles compared to ECB mode and thus cause greater processing overhead;
e) under the same assumptions about the block cipher as in Properties of CBC the counter mode can also be proven to be secure.

Padding requirements

Only multiples of j bits can be enciphered or deciphered. Other lengths need to be padded to a j-bit boundary. However, usually j will be chosen equal to such a size, that no padding will be required, e.g. j can be modified for the last portion of the plaintext. There is no padding 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.

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 deciphered plaintext.

Synchronization

The Counter Mode is not self-synchronizing. If the two operations of encipherment and decipherment 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.

Also overlapping sequences of counter values while using the same key should be prevented as these make the counter mode also vulnerable to for example “known plaintext” and known “ciphertext attacks”.

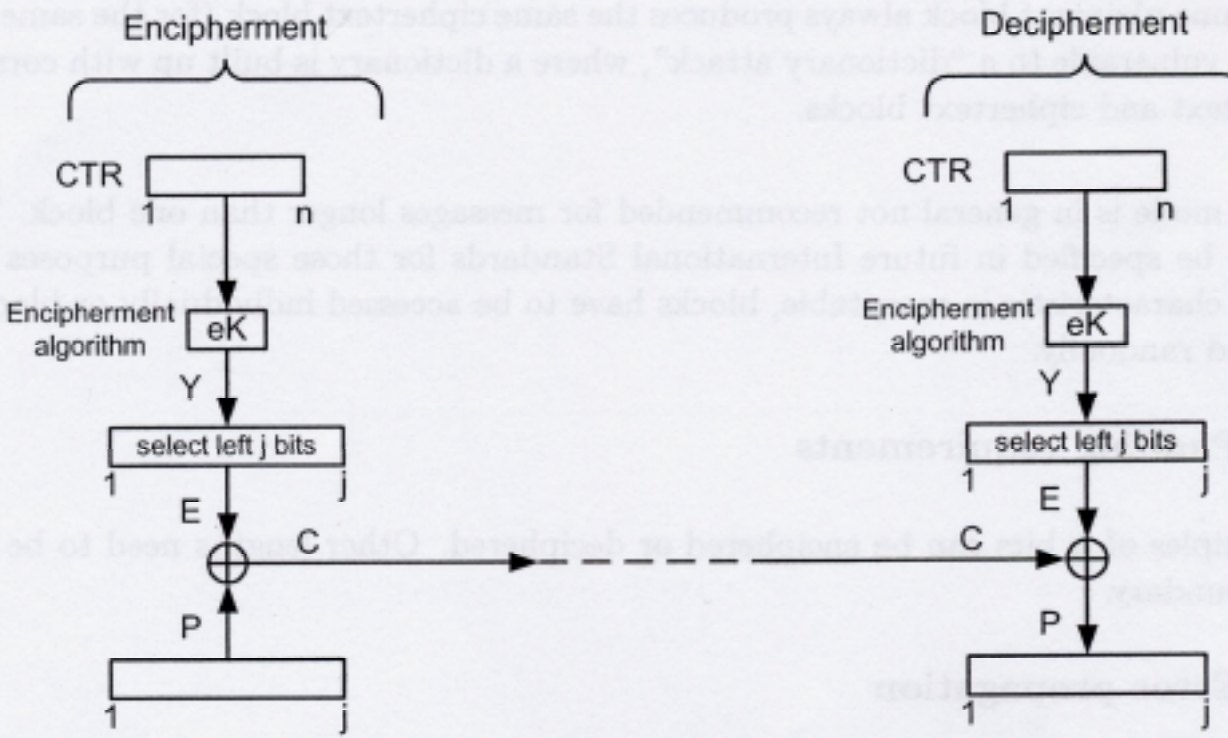

Figure A.4 – The counter (CTR) mode of operation