XOR con R_c

k = 128 bits

```
EXPANDED KEY----------------------------------------------------------------
00000000626262629bf99bf99069f20bee87757e7ff88df3ec14996a2135acc60e3b9751b18a1d4cb43e236f
00000000636363639bf99bf99069f20bee87757e7ff88df3ec14996a2135acc60e3b9751b18a1d4cb43e236f
00000000636363639bf99bf99069f20bee87757e7ff88df3ec14996a2135acc60e3b9751b18a1d4cb43e236f
00000000636363639bf99bf99069f20bee87757e7ff88df3ec14996a2135acc60e3b9751b18a1d4cb43e236f
--------  --------  --------  --------  --------  --------  --------  --------  --------  --------
   00        01        02        03        04        05        06        07        08        09        0a
----------------------------------------------------------------------------------------------------
```

```
---INITIAL STATE---
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
----AddRoundKey----
KEY:
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
STATE:
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
-----ApplyBytes-----
63 63 63 63
63 63 63 63
63 63 63 63
63 63 63 63
----ApplyShiftRows----
63 63 63 63
63 63 63 63
63 63 63 63
63 63 63 63
----ApplyMixColumns----
63 63 63 63
63 63 63 63
63 63 63 63
63 63 63 63
----AddRoundKey----
KEY:
62 62 62 62
63 63 63 63
63 63 63 63
63 63 63 63
STATE:
01 01 01 01
00 00 00 00
00 00 00 00
00 00 00 01
-----ApplyBytes-----
7c 7c 7c 7c
63 63 63 63
63 63 63 63
63 63 63 63
----ApplyShiftRows----
7c 7c 7c 7c
63 63 63 63
63 63 63 63
63 63 63 63
----ApplyMixColumns----
5d 5d 5d 5d
7c 7c 7c 7c
7c 7c 7c 7c
42 42 42 42
```

```
----AddRoundKey----
KEY:
9b f9 9b f9
98 fb 98 fb
98 fb 98 fb
c9 aa c9 aa
STATE:
c6 a4 c6 a4
e4 87 e4 87
e4 87 e4 87
8b e8 8b e8
-----ApplyBytes-----
b4 49 b4 49
69 17 69 17
69 17 69 17
3d 9b 3d 9b
----ApplyShiftRows----
b4 49 b4 49
17 69 17 69
69 17 69 17
9b 3d 9b 3d
----ApplyMixColumns----
b8 03 b8 03
ba 9f ba 9f
c7 49 c7 49
94 df 94 df
----AddRoundKey----
KEY:
90 69 f2 0b
97 6c f4 0f
34 cf 57 ac
50 fa 33 99
STATE:
28 6a 4a 08
2d f3 4e 90
f3 86 90 e5
c4 25 a7 46
-----ApplyBytes-----
34 02 d6 30
d8 0d 2f 60
0d 44 60 d9
1c 3f 5c 5a
----ApplyShiftRows----
34 02 d6 30
0d 2f 60 d8
60 d9 0d 44
5a 1c 3f 5c
----ApplyMixColumns----
45 b0 25 0b
d4 30 3e 0b
17 a0 ed 84
85 c8 72 74
```

Attenzione: l'uso di chiavi più lunghe non necessariamente rende il sistema più robusto. → più lunga è la chiave più servono cicli.

co · ca

---

oss Aes con un numero ridotto di cicli di codifica è attaccabile.

→ || Mb/s: è utile riciclare la matematica di AES.

$\mathbb{F}_{256}$ $\quad$ $\frac{\mathbb{F}_{256}[x]}{(x^4+1)}$ $\quad$ MIXCOLUMNS

prodotti MATRICE/vettore.

→ per costruzione di funzioni Hash

→ per costruzioni (su campi più grandi ma sempre finiti) di crittosistemi basati su DLOG.

→ → || $\mathbb{F}_{p^n}^*$ è ciclico $\forall n$, p primo

↑ DI SOLITO PER DLOG SI USA $\mathbb{Z}_p^*$ p primo

o $\mathbb{F}_{2^n}^*$ n grande, possibilmente 2^n-1 primo.

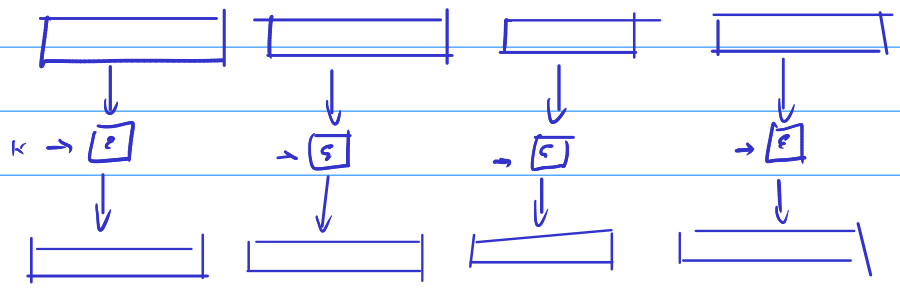
↑

gli elementi sono stringhe di bit e la somma è uno XOR.

~~$\mathbb{F}_{2^n}$~~ Come si usa un cifrario a blocchi.

Un cifrario a blocchi deve agire su di un testo che è composto da più unità di lunghezza un blocco.

ECB:



ELECTRONIC CODEBOOK MODE. / sostituzione monoalfabetica.

$b = 128$ bits.

NIST Special Publication 800-38A
2001 Edition

Recommendation for Block Cipher Modes of Operation

Methods and Techniques

Morris Dworkin

National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

C O M P U T E R S E C U R I T Y

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

December 2001

U.S. Department of Commerce
Donald L. Evans, Secretary

Technology Administration
Phillip J. Bond, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
Arden L. Bement, Jr., Director

Reports on Information Security Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Special Publication 800-38A 2001 ED
Natl. Inst. Stand. Technol. Spec. Publ. 800-38A 2001 ED, 66 pages (December 2001)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2001

For sale by the Superintendent of Documents, U.S. Government Printing Office
Internet: bookstore.gpo.gov — Phone: (202) 512-1800 — Fax: (202) 512-2250
Mail: Stop SSOP, Washington, DC 20402-0001

Abstract

This recommendation defines five confidentiality modes of operation for use with an underlying symmetric key block cipher algorithm: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR). Used with an underlying block cipher algorithm that is approved in a Federal Information Processing Standard (FIPS), these modes can provide cryptographic protection for sensitive, but unclassified, computer data.

KEY WORDS: Computer security; cryptography; data security; block cipher; encryption; Federal Information Processing Standard; mode of operation.

Table of Contents

1 PURPOSE 1

2 AUTHORITY 1

3 INTRODUCTION 1

4 DEFINITIONS, ABBREVIATIONS, AND SYMBOLS 3

4.1 Definitions and Abbreviations 3
4.2 Symbols 5
4.2.1 Variables 5
4.2.2 Operations and Functions 5

5 PRELIMINARIES 7

5.1 Underlying Block Cipher Algorithm 7
5.2 Representation of the Plaintext and the Ciphertext 7
5.3 Initialization Vectors 8
5.4 Examples of Operations and Functions 8

6 BLOCK CIPHER MODES OF OPERATION 9

6.1 The Electronic Codebook Mode 9
6.2 The Cipher Block Chaining Mode 10
6.3 The Cipher Feedback Mode 11
6.4 The Output Feedback Mode 13
6.5 The Counter Mode 15

APPENDIX A: PADDING 17

APPENDIX B: GENERATION OF COUNTER BLOCKS 18

B.1 The Standard Incrementing Function 18
B.2 Choosing Initial Counter Blocks 19

APPENDIX C: GENERATION OF INITIALIZATION VECTORS 20

APPENDIX D: ERROR PROPERTIES 21

APPENDIX E: MODES OF TRIPLE DES 23

APPENDIX F: EXAMPLE VECTORS FOR MODES OF OPERATION OF THE AES 24

F.1 ECB Example Vectors 24
F.1.1 ECB-AES128.Encrypt 24
F.1.2 ECB-AES128.Decrypt 24
F.1.3 ECB-AES192.Encrypt 25
F.1.4 ECB-AES192.Decrypt 25
F.1.5 ECB-AES256.Encrypt 26
F.1.6 ECB-AES256.Decrypt 26
F.2 CBC Example Vectors 27
F.2.1 CBC-AES128.Encrypt 27
F.2.2 CBC-AES128.Decrypt 27
F.2.3 CBC-AES192.Encrypt 28
F.2.4 CBC-AES192.Decrypt 28

F.2.5 CBC-AES256.Encrypt *28*
F.2.6 CBC-AES256.Decrypt *29*
F.3 CFB EXAMPLE VECTORS 29
F.3.1 CFB1-AES128.Encrypt *29*
F.3.2 CFB1-AES128.Decrypt *31*
F.3.3 CFB1-AES192.Encrypt *33*
F.3.4 CFB1-AES192.Decrypt *34*
F.3.5 CFB1-AES256.Encrypt *36*
F.3.6 CFB1-AES256.Decrypt *37*
F.3.7 CFB8-AES128.Encrypt *39*
F.3.8 CFB8-AES128.Decrypt *41*
F.3.9 CFB8-AES192.Encrypt *42*
F.3.10 CFB8-AES192.Decrypt *44*
F.3.11 CFB8-AES256.Encrypt *46*
F.3.12 CFB8-AES256.Decrypt *48*
F.3.13 CFB128-AES128.Encrypt *50*
F.3.14 CFB128-AES128.Decrypt *50*
F.3.15 CFB128-AES192.Encrypt *50*
F.3.16 CFB128-AES192.Decrypt *51*
F.3.17 CFB128-AES256.Encrypt *51*
F.3.18 CFB128-AES256.Decrypt *52*
F.4 OFB EXAMPLE VECTORS 52
F.4.1 OFB-AES128.Encrypt *52*
F.4.2 OFB-AES128.Decrypt *53*
F.4.3 OFB-AES192.Encrypt *53*
F.4.4 OFB-AES192.Decrypt *54*
F.4.5 OFB-AES256.Encrypt *54*
F.4.6 OFB-AES256.Decrypt *55*
F.5 CTR EXAMPLE VECTORS 55
F.5.1 CTR-AES128.Encrypt *55*
F.5.2 CTR-AES128.Decrypt *56*
F.5.3 CTR-AES192.Encrypt *56*
F.5.4 CTR-AES192.Decrypt *57*
F.5.5 CTR-AES256.Encrypt *57*
F.5.6 CTR-AES256.Decrypt *57*
APPENDIX G: REFERENCES 59

Table of Figures

Figure 1: The ECB Mode 9
Figure 2: The CBC Mode 10
Figure 3: The CFB Mode 12
Figure 4: The OFB Mode 14
Figure 5: The CTR Mode 16

1 Purpose

This publication provides recommendations regarding modes of operation to be used with symmetric key block cipher algorithms.

2 Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Computer Security Act of 1987 (Public Law 100-235) and the Information Technology Management Reform Act of 1996, specifically 15 U.S.C. 278 g-3(a)(5). This is not a guideline within the meaning of 15 U.S.C. 278 g-3 (a)(5).

This recommendation is neither a standard nor a guideline, and as such, is neither mandatory nor binding on Federal agencies. Federal agencies and non-government organizations may use this recommendation on a voluntary basis. It is not subject to copyright.

Nothing in this recommendation should be taken to contradict standards and guidelines that have been made mandatory and binding upon Federal agencies by the Secretary of Commerce under his statutory authority. Nor should this recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, the Director of the Office of Management and Budget, or any other Federal official.

Conformance testing for implementations of the modes of operation that are specified in this recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of the NIST and the Communications Security Establishment of the Government of Canada. An implementation of a mode of operation must adhere to the requirements in this recommendation in order to be validated under the CMVP.

3 Introduction

This recommendation specifies five confidentiality modes of operation for symmetric key block cipher algorithms, such as the algorithm specified in FIPS Pub. 197, the Advanced Encryption Standard (AES) [2]. The modes may be used in conjunction with any symmetric key block cipher algorithm that is approved by a Federal Information Processing Standard (FIPS). The five modes—the Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes—can provide data confidentiality.

Two FIPS publications already approve confidentiality modes of operation for two particular block cipher algorithms. FIPS Pub. 81 [4] specifies the ECB, CBC, CFB, and OFB modes of the Data Encryption Standard (DES). FIPS Pub. 46-3 [3] approves the seven modes that are specified in ANSI X9.52 [1]. Four of these modes are equivalent to the ECB, CBC, CFB, and OFB modes with the Triple DES algorithm (TDEA) as the underlying block cipher; the other

three modes in ANSI X9.52 are variants of the CBC, CFB, and OFB modes of Triple DES that use interleaving or pipelining.

Thus, there are three new elements in this recommendation: 1) the extension of the four confidentiality modes in FIPS Pub 81 for use with any FIPS-approved block cipher; 2) the revision of the requirements for these modes; and 3) the specification of an additional confidentiality mode, the CTR mode, for use with any FIPS-approved block cipher.

4 Definitions, Abbreviations, and Symbols

4.1 Definitions and Abbreviations

Bit	A binary digit: 0 or 1.
Bit Error	The substitution of a ‘0’ bit for a ‘1’ bit, or vice versa.
Bit String	An ordered sequence of 0’s and 1’s.
Block Cipher	A family of functions and their inverse functions that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.
Block Size	The number of bits in an input (or output) block of the block cipher.
CBC	Cipher Block Chaining.
CFB	Cipher Feedback.
Ciphertext	Encrypted data.
Confidentiality Mode	A mode that is used to encipher plaintext and decipher ciphertext. The confidentiality modes in this recommendation are the ECB, CBC, CFB, OFB, and CTR modes.
CTR	Counter.
Cryptographic Key	A parameter used in the block cipher algorithm that determines the forward cipher operation and the inverse cipher operation.
Data Block (Block)	A sequence of bits whose length is the block size of the block cipher.
Data Segment (Segment)	In the CFB mode, a sequence of bits whose length is a parameter that does not exceed the block size.
Decryption (Deciphering)	The process of a confidentiality mode that transforms encrypted data into the original usable data.
ECB	Electronic Codebook.
Encryption (Enciphering)	The process of a confidentiality mode that transforms usable data into an unreadable form.

Exclusive-OR	The bitwise addition, modulo 2, of two bit strings of equal length.
FIPS	Federal Information Processing Standard.
Forward Cipher Function (Forward Cipher Operation)	One of the two functions of the block cipher algorithm that is selected by the cryptographic key.
Initialization Vector (IV)	A data block that some modes of operation require as an additional initial input.
Input Block	A data block that is an input to either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Inverse Cipher Function (Inverse Cipher Operation)	The function that reverses the transformation of the forward cipher function when the same cryptographic key is used.
Least Significant Bit(s)	The right-most bit(s) of a bit string.
Mode of Operation (Mode)	An algorithm for the cryptographic transformation of data that features a symmetric key block cipher algorithm.
Most Significant Bit(s)	The left-most bit(s) of a bit string.
Nonce	A value that is used only once.
Octet	A group of eight binary digits.
OFB	Output Feedback.
Output Block	A data block that is an output of either the forward cipher function or the inverse cipher function of the block cipher algorithm.
Plaintext	Usable data that is formatted as input to a mode.

4.2 Symbols

4.2.1 Variables

b	The block size, in bits.
j	The index to a sequence of data blocks or data segments ordered from left to right.
n	The number of data blocks or data segments in the plaintext.
s	The number of bits in a data segment.
u	The number of bits in the last plaintext or ciphertext block.
C_j	The j^{th} ciphertext block.
$C^{\#}_j$	The j^{th} ciphertext segment.
C^{*}_n	The last block of the ciphertext, which may be a partial block.
I_j	The j^{th} input block.
IV	The initialization vector.
K	The secret key.
O_j	The j^{th} output block.
P_j	The j^{th} plaintext block.
$P^{\#}_j$	The j^{th} plaintext segment.
P^{*}_n	The last block of the plaintext, which may be a partial block.
T_j	The j^{th} counter block.

4.2.2 Operations and Functions

$X \mid Y$	The concatenation of two bit strings X and Y.
$X \oplus Y$	The bitwise exclusive-OR of two bit strings X and Y of the same length.
$CIPH_K(X)$	The forward cipher function of the block cipher algorithm under the key K applied to the data block X.

$CIPH^{-1}_{K}(X)$	The inverse cipher function of the block cipher algorithm under the key K applied to the data block X.
$LSB_m(X)$	The bit string consisting of the m least significant bits of the bit string X.
$MSB_m(X)$	The bit string consisting of the m most significant bits of the bit string X.
$[x]_m$	The binary representation of the non-negative integer x, in m bits, where $x<2^m$.

5 Preliminaries

5.1 Underlying Block Cipher Algorithm

This recommendation assumes that a FIPS-approved symmetric key block cipher algorithm has been chosen as the underlying algorithm, and that a secret, random key, denoted K, has been established among all of the parties to the communication. The cryptographic key regulates the functioning of the block cipher algorithm and, thus, by extension, regulates the functioning of the mode. The specifications of the block cipher and algorithms and the modes are public, so the security of the mode depends, at a minimum, on the secrecy of the key.

A confidentiality mode of operation of the block cipher algorithm consists of two processes that are inverses of each other: encryption and decryption. Encryption is the transformation of a usable message, called the plaintext, into an unreadable form, called the ciphertext; decryption is the transformation that recovers the plaintext from the ciphertext.

For any given key, the underlying block cipher algorithm of the mode also consists of two functions that are inverses of each other. These two functions are often called encryption and decryption, but in this recommendation, those terms are reserved for the processes of the confidentiality modes. Instead, as part of the choice of the block cipher algorithm, one of the two functions is designated as the forward cipher function, denoted $CIPH_K$; the other function is then called the inverse cipher function, denoted $CIPH^{-1}_K$. The inputs and outputs of both functions are called input blocks and output blocks. The input and output blocks of the block cipher algorithm have the same bit length, called the block size, denoted b.

5.2 Representation of the Plaintext and the Ciphertext

For all of the modes in this recommendation, the plaintext must be represented as a sequence of bit strings; the requirements on the lengths of the bit strings vary according to the mode:

For the ECB and CBC modes, the total number of bits in the plaintext must be a multiple of the block size, b; in other words, for some positive integer n, the total number of bits in the plaintext must be nb. The plaintext consists of a sequence of n bit strings, each with bit length b. The bit strings in the sequence are called data blocks, and the plaintext is denoted $P_1, P_2, \ldots, P_n$.

For the CFB mode, the total number of bits in the plaintext must be a multiple of a parameter, denoted s, that does not exceed the block size; in other words, for some positive integer n, the total number of bits in the message must be ns. The plaintext consists of a sequence of n bit strings, each with bit length s. The bit strings in the sequence are called data segments, and the plaintext is denoted $P^{\#}_1, P^{\#}_2, \ldots, P^{\#}_n$.

For the OFB and CTR modes, the plaintext need not be a multiple of the block size. Let n and u denote the unique pair of positive integers such that the total number of bits in the message is $(n\text{-}1)b+u$, where $1 \leq u \leq b$. The plaintext consists of a sequence of n bit strings, in which the bit length of the last bit string is u, and the bit length of the other bit strings is b. The sequence is denoted $P_1, P_2, \ldots, P_{n-1}, P^{*}_n$, and the bit strings are called data blocks, although the last bit string,

P^*_n , may not be a complete block.

For each mode, the encryption process transforms every plaintext data block or segment into a corresponding ciphertext data block or segment with the same bit length, so that the ciphertext is a sequence of data blocks or segments. The ciphertext is denoted as follows: for the ECB and CBC modes, $C_1, C_2,\ldots, C_n$; for the CFB mode, $C^\#_1, C^\#_2,\ldots, C^\#_n$; and, for the OFB and CTR modes, $C_1, C_2,\ldots, C_{n-1}, C^*_n$, where C^*_n may be a partial block.

The formatting of the plaintext, including in some cases the appending of padding bits to form complete data blocks or data segments, is outside the scope of this recommendation. Padding is discussed in Appendix A.

5.3 Initialization Vectors

The input to the encryption processes of the CBC, CFB, and OFB modes includes, in addition to the plaintext, a data block called the initialization vector (IV), denoted *IV*. The IV is used in an initial step in the encryption of a message and in the corresponding decryption of the message.

The IV need not be secret; however, for the CBC and CFB modes, the IV for any particular execution of the encryption process must be unpredictable, and, for the OFB mode, unique IVs must be used for each execution of the encryption process. The generation of IVs is discussed in Appendix C.

5.4 Examples of Operations and Functions

The concatenation operation on bit strings is denoted | ; for example, 001 | 10111 = 00110111.

Given bit strings of equal length, the exclusive-OR operation, denoted $\oplus$, specifies the addition, modulo 2, of the bits in each bit position, i.e., without carries. Thus, 10011 $\oplus$ 10101= 00110, for example.

The functions LSB_s and MSB_s return the s least significant bits and the s most significant bits of their arguments. For example, $LSB_3(111011010) = 010$, and $MSB_4(111011010) = 1110$.

Given a positive integer m and a non-negative (decimal) integer x that is less than 2^m, the binary representation of x in m bits is denoted $[x]_m$. For example, $[45]_8 = 00101101$.

6 Block Cipher Modes of Operation

The mathematical specifications of the five modes are given in Sections 6.1-6.5, along with descriptions, illustrations, and comments on the potential for parallel processing.

6.1 The Electronic Codebook Mode

The Electronic Codebook (ECB) mode is a confidentiality mode that features, for a given key, the assignment of a fixed ciphertext block to each plaintext block, analogous to the assignment of code words in a codebook. The Electronic Codebook (ECB) mode is defined as follows:

ECB Encryption: $C_j = CIPH_K(P_j)$ for $j = 1 \ldots n$.

ECB Decryption: $P_j = CIPH^{-1}_K(C_j)$ for $j = 1 \ldots n$.

In ECB encryption, the forward cipher function is applied directly and independently to each block of the plaintext. The resulting sequence of output blocks is the ciphertext.

In ECB decryption, the inverse cipher function is applied directly and independently to each block of the ciphertext. The resulting sequence of output blocks is the plaintext.

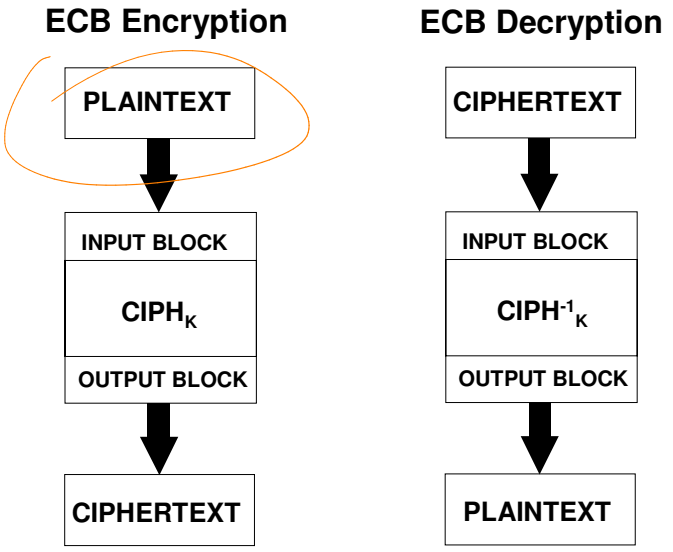


Figure 1: The ECB Mode

In ECB encryption and ECB decryption, multiple forward cipher functions and inverse cipher functions can be computed in parallel.

In the ECB mode, under a given key, any given plaintext block always gets encrypted to the

same ciphertext block. If this property is undesirable in a particular application, the ECB mode should not be used.

The ECB mode is illustrated in Figure 1.

6.2 The Cipher Block Chaining Mode

The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features the combining ("chaining") of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an IV to combine with the first plaintext block. The IV need not be secret, but it must be unpredictable; the generation of such IVs is discussed in Appendix C. Also, the integrity of the IV should be protected, as discussed in Appendix D. The CBC mode is defined as follows:

CBC Encryption:

$$C_1 = CIPH_K(P_1 \oplus IV);$$
$$C_j = CIPH_K(P_j \oplus C_{j-1}) \quad \text{for } j = 2 \ldots n.$$

CBC Decryption:

$$P_1 = CIPH^{-1}_K(C_1) \oplus IV;$$
$$P_j = CIPH^{-1}_K(C_j) \oplus C_{j-1} \quad \text{for } j = 2 \ldots n.$$

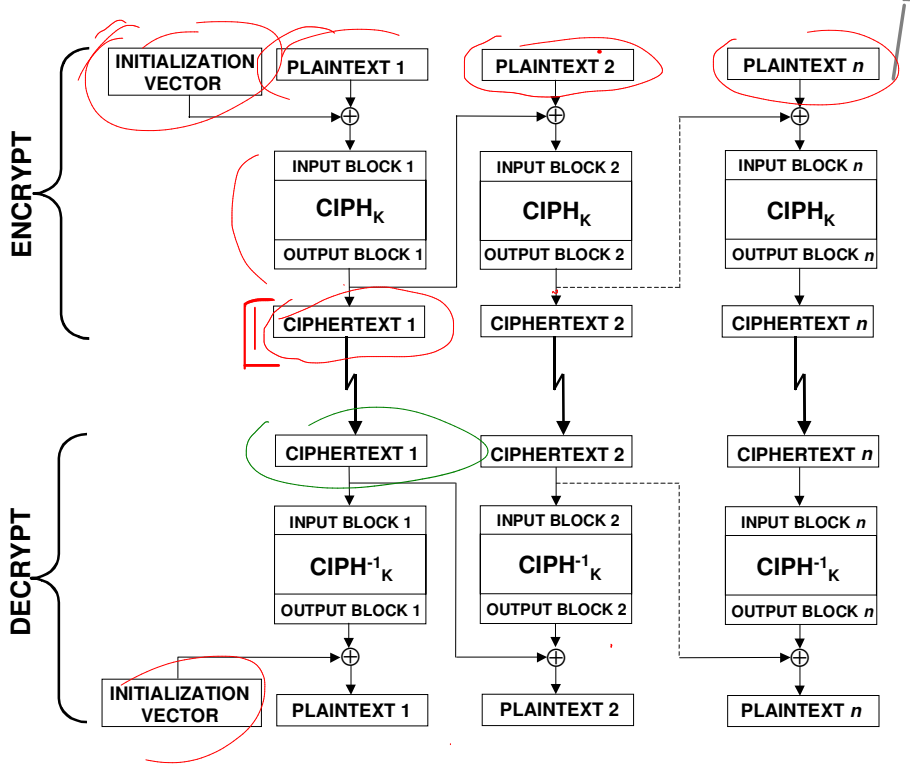
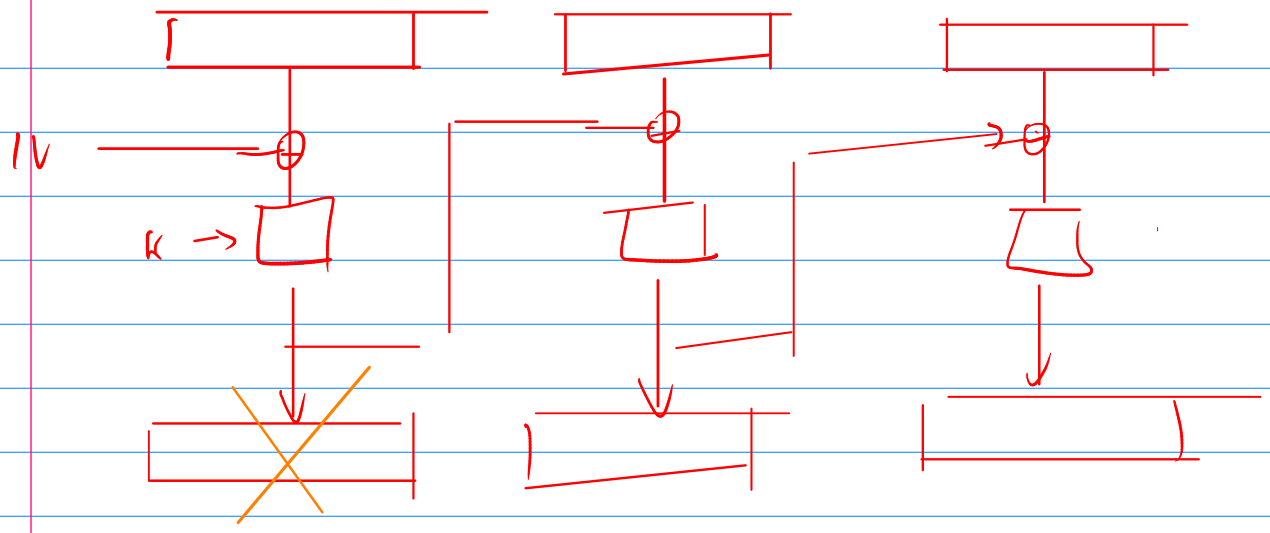


Figure 2: The CBC Mode

In CBC encryption, the first input block is formed by exclusive-ORing the first block of the plaintext with the IV. The forward cipher function is applied to the first input block, and the



→ la codifica è seriale.

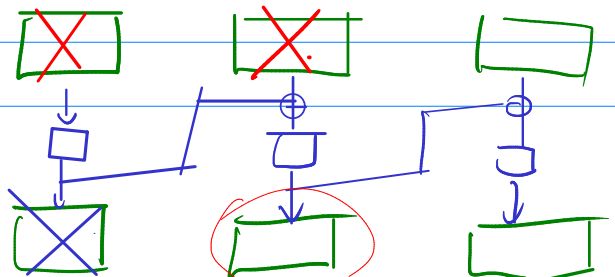
1, 2, 3 ——— n

per codificare il blocco i, mi serve la codifica del blocco i-1

→ la decodifica è parallela.

→ per decodificare il blocco i mi serve la codifica del blocco i-1

→ in presenza di errori, si perdono al più 2 blocchi



N.B: Se si deve aggiungere un blocco in coda non si devono modificare i blocchi precedenti.

Se si modifica un blocco in posizione i, tutti i blocchi da i+1 in poi devono essere rilasciati

resulting output block is the first block of the ciphertext. This output block is also exclusive-ORed with the second plaintext data block to produce the second input block, and the forward cipher function is applied to produce the second output block. This output block, which is the second ciphertext block, is exclusive-ORed with the next plaintext block to form the next input block. Each successive plaintext block is exclusive-ORed with the previous output/ciphertext block to produce the new input block. The forward cipher function is applied to each input block to produce the ciphertext block.

In CBC decryption, the inverse cipher function is applied to the first ciphertext block, and the resulting output block is exclusive-ORed with the initialization vector to recover the first plaintext block. The inverse cipher function is also applied to the second ciphertext block, and the resulting output block is exclusive-ORed with the first ciphertext block to recover the second plaintext block. In general, to recover any plaintext block (except the first), the inverse cipher function is applied to the corresponding ciphertext block, and the resulting block is exclusive-ORed with the previous ciphertext block.

In CBC encryption, the input block to each forward cipher operation (except the first) depends on the result of the previous forward cipher operation, so the forward cipher operations cannot be performed in parallel. In CBC decryption, however, the input blocks for the inverse cipher function, i.e., the ciphertext blocks, are immediately available, so that multiple inverse cipher operations can be performed in parallel.

The CBC mode is illustrated in Figure 2.

6.3 The Cipher Feedback Mode

The Cipher Feedback (CFB) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input block. The IV need not be secret, but it must be unpredictable; the generation of such IVs is discussed in Appendix C.

The CFB mode also requires an integer parameter, denoted s, such that $1 \le s \le b$. In the specification of the CFB mode below, each plaintext segment ($P^{\#}_j$) and ciphertext segment ($C^{\#}_j$) consists of s bits. The value of s is sometimes incorporated into the name of the mode, e.g., the 1-bit CFB mode, the 8-bit CFB mode, the 64-bit CFB mode, or the 128-bit CFB mode.

The CFB mode is defined as follows:

CFB Encryption:

$$I_1 = IV;$$
$$I_j = LSB_{b-s}(I_{j-1}) \,|\, C^{\#}_{j-1} \quad \text{for } j = 2 \ldots n;$$
$$O_j = CIPH_K(I_j) \quad \text{for } j = 1, 2 \ldots n;$$
$$C^{\#}_j = P^{\#}_j \oplus MSB_s(O_j) \quad \text{for } j = 1, 2 \ldots n.$$

CFB Decryption:

$$I_1 = IV;$$
$$I_j = LSB_{b-s}(I_{j-1}) \,|\, C^{\#}_{j-1} \quad \text{for } j = 2 \ldots n;$$

$$O_j = CIPH_K(I_j) \qquad \text{for } j = 1, 2 \ldots n;$$
$$P^{\#}_j = C^{\#}_j \oplus MSB_s(O_j) \qquad \text{for } j = 1, 2 \ldots n.$$

In CFB encryption, the first input block is the IV, and the forward cipher operation is applied to the IV to produce the first output block. The first ciphertext segment is produced by exclusive-ORing the first plaintext segment with the s most significant bits of the first output block. (The remaining b-s bits of the first output block are discarded.) The b-s least significant bits of the IV are then concatenated with the s bits of the first ciphertext segment to form the second input block. An alternative description of the formation of the second input block is that the bits of the first input block circularly shift s positions to the left, and then the ciphertext segment replaces the s least significant bits of the result.

The process is repeated with the successive input blocks until a ciphertext segment is produced from every plaintext segment. In general, each successive input block is enciphered to produce an output block. The s most significant bits of each output block are exclusive-ORed with the corresponding plaintext segment to form a ciphertext segment. Each ciphertext segment (except the last one) is "fed back" into the previous input block, as described above, to form a new input block. The feedback can be described in terms of the individual bits in the strings as follows: if $i_1 i_2 \ldots i_b$ is the jth input block, and $c_1 c_2 \ldots c_s$ is the jth ciphertext segment, then the $(j+1)^{th}$ input block is $i_{s+1} i_{s+2} \ldots i_b \, c_1 c_2 \ldots c_s$.

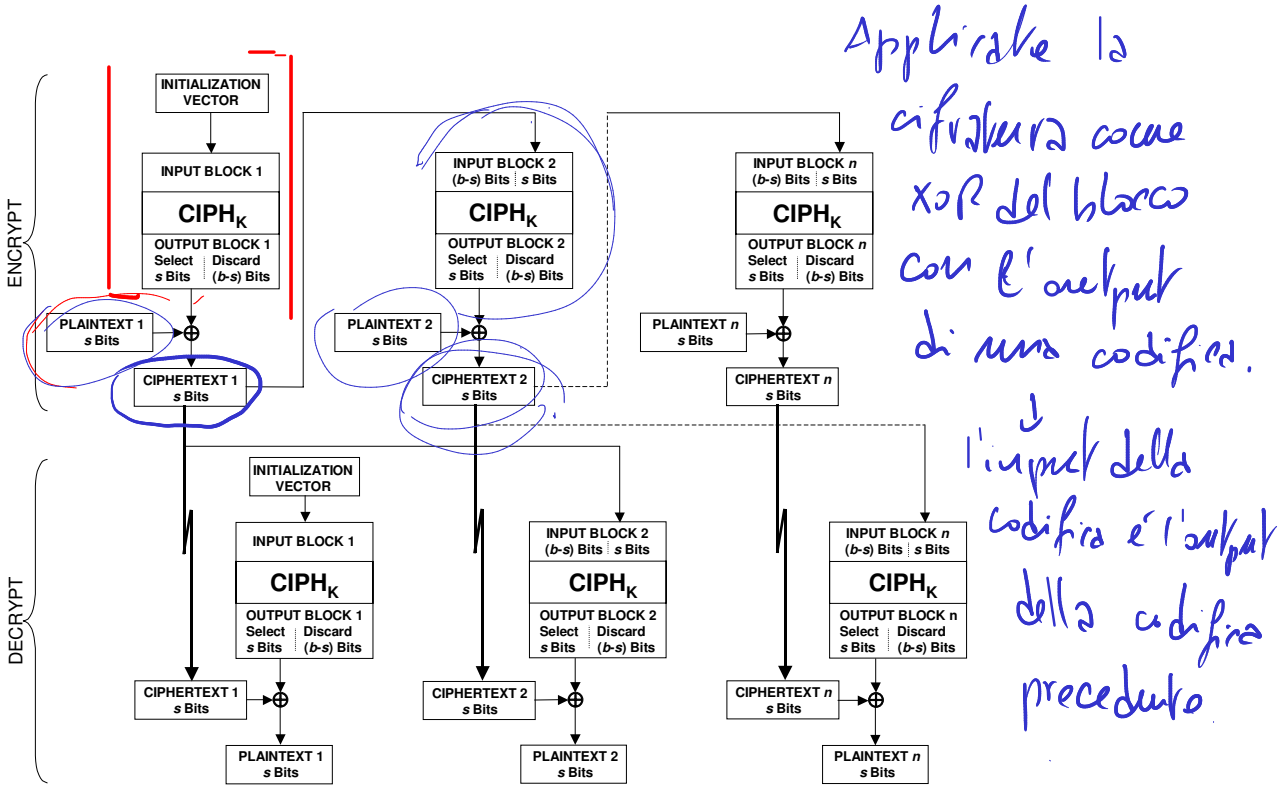
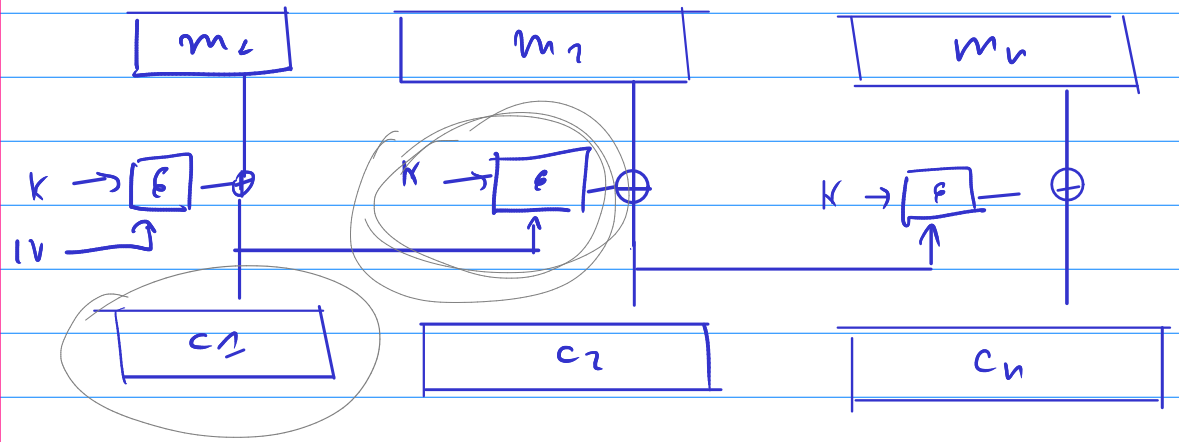


Figure 3: The CFB Mode

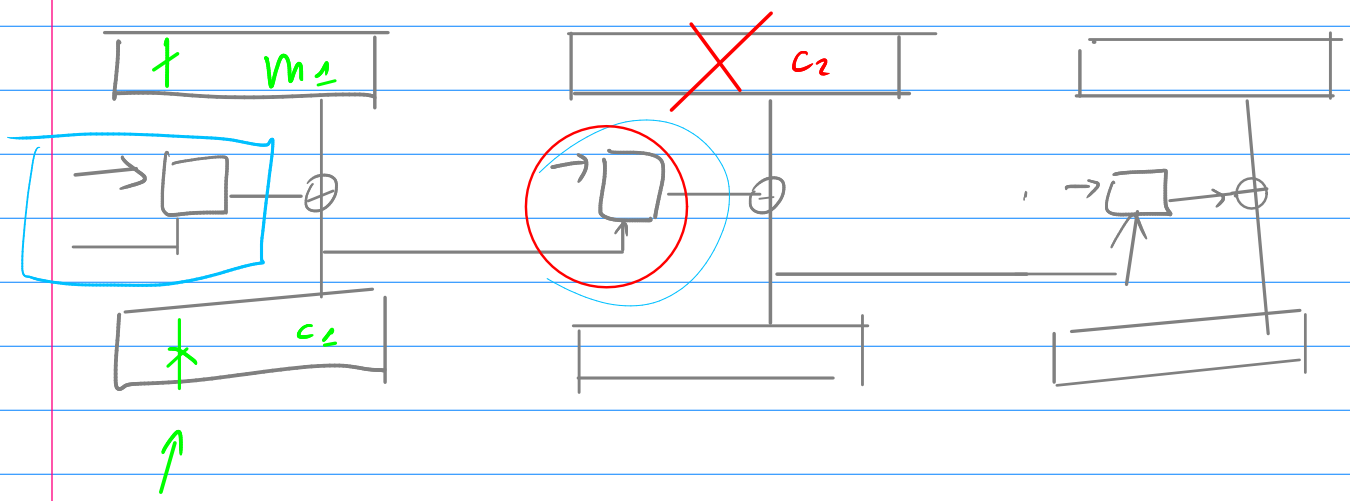
In CFB decryption, the IV is the first input block, and each successive input block is formed as in CFB encryption, by concatenating the b-s least significant bits of the previous input block with



OSS 1) NON SERVE IMPLEMENTARE LA DECODIFICA DEL CIFRARIO A BLOCCHI.

OSS 2) Un errore di 1 bit altera.

1 blocco + 1 bit.



the *s* most significant bits of the previous ciphertext. The *forward cipher* function is applied to each input block to produce the output blocks. The *s* most significant bits of the output blocks are exclusive-ORed with the corresponding ciphertext segments to recover the plaintext segments.

In CFB encryption, like CBC encryption, the input block to each forward cipher function (except the first) depends on the result of the previous forward cipher function; therefore, multiple forward cipher operations cannot be performed in parallel. In CFB decryption, the required forward cipher operations can be performed in parallel if the input blocks are first constructed (in series) from the IV and the ciphertext.

The CFB mode is illustrated in Figure 3.

6.4 The Output Feedback Mode

Stream cipher.

The Output Feedback (OFB) mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The OFB mode requires that the IV is a nonce, i.e., the IV must be unique for each execution of the mode under the given key; the generation of such IVs is discussed in Appendix C. The OFB mode is defined as follows:

OFB Encryption:

$I_1 = IV;$
$I_j = O_{j-1}$ for $j = 2 \ldots n;$
$O_j = CIPH_K(I_j)$ for $j = 1, 2 \ldots n;$
$C_j = P_j \oplus O_j$ for $j = 1, 2 \ldots n\text{-}1;$
$C^*_n = P^*_n \oplus MSB_u(O_n).$

OFB Decryption:

$I_1 = IV;$
$I_j = O_{j-1}$ for $j = 2 \ldots n;$
$O_j = CIPH_K(I_j)$ for $j = 1, 2 \ldots n;$
$P_j = C_j \oplus O_j$ for $j = 1, 2 \ldots n\text{-}1;$
$P^*_n = C^*_n \oplus MSB_u(O_n).$

In OFB encryption, the IV is transformed by the forward cipher function to produce the first output block. The first output block is exclusive-ORed with the first plaintext block to produce the first ciphertext block. The forward cipher function is then invoked on the first output block to produce the second output block. The second output block is exclusive-ORed with the second plaintext block to produce the second ciphertext block, and the forward cipher function is invoked on the second output block to produce the third output block. Thus, the successive output blocks are produced from applying the forward cipher function to the previous output blocks, and the output blocks are exclusive-ORed with the corresponding plaintext blocks to produce the ciphertext blocks. For the last block, which may be a partial block of *u* bits, the most significant *u* bits of the last output block are used for the exclusive-OR operation; the remaining *b*-*u* bits of the last output block are discarded.

In OFB decryption, the IV is transformed by the *forward cipher* function to produce the first

output block. The first output block is exclusive-ORed with the first ciphertext block to recover the first plaintext block. The first output block is then transformed by the forward cipher function to produce the second output block. The second output block is exclusive-ORed with the second ciphertext block to produce the second plaintext block, and the second output block is also transformed by the forward cipher function to produce the third output block. Thus, the successive output blocks are produced from applying the forward cipher function to the previous output blocks, and the output blocks are exclusive-ORed with the corresponding ciphertext blocks to recover the plaintext blocks. For the last block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the exclusive-OR operation; the remaining b-u bits of the last output block are discarded.

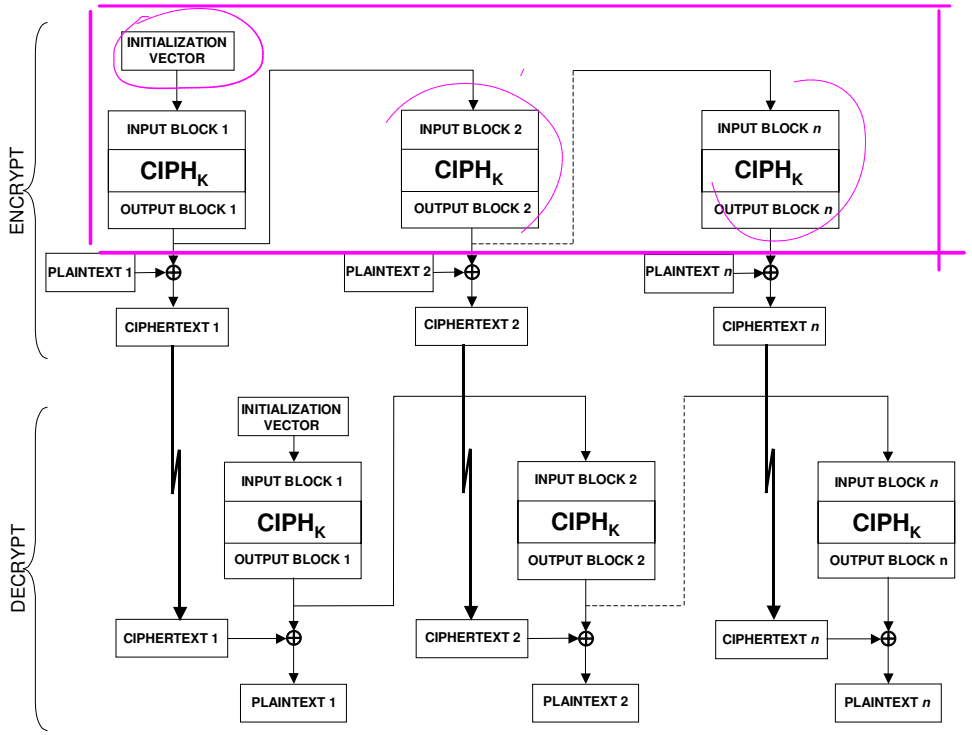
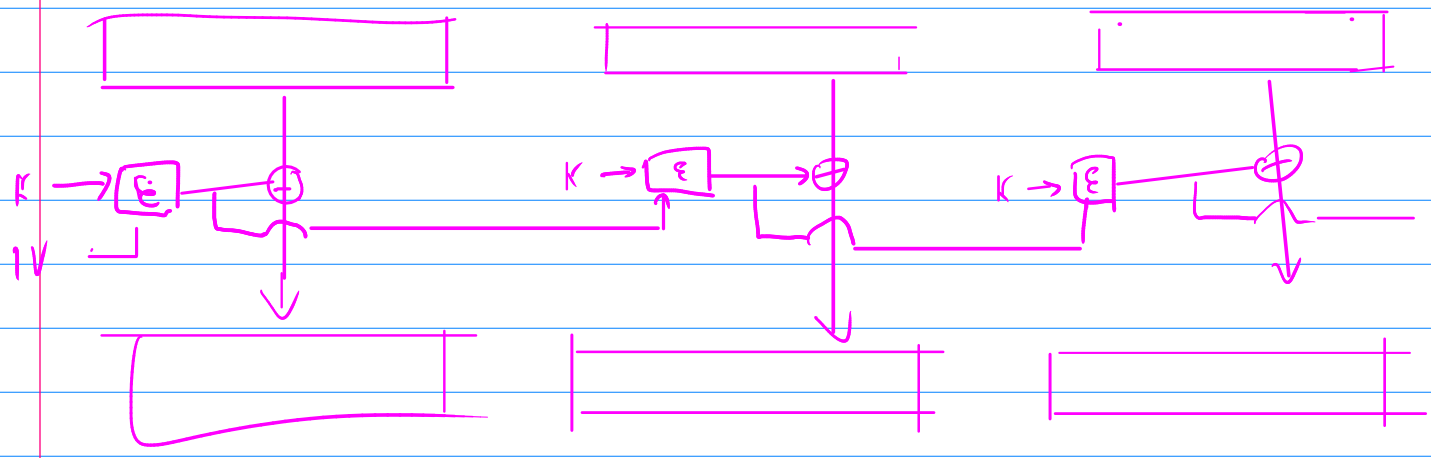


Figure 4: The OFB Mode

In both OFB encryption and OFB decryption, each forward cipher function (except the first) depends on the results of the previous forward cipher function; therefore, multiple forward cipher functions cannot be performed in parallel. However, if the IV is known, the output blocks can be generated prior to the availability of the plaintext or ciphertext data.

The OFB mode requires a unique IV for every message that is ever encrypted under the given key. If, contrary to this requirement, the same IV is used for the encryption of more than one message, then the confidentiality of those messages may be compromised. In particular, if a plaintext block of any of these messages is known, say, the jth plaintext block, then the jth output of the forward cipher function can be determined easily from the jth ciphertext block of the message. This information allows the jth plaintext block of any other message that is encrypted



Il cifrario a blocchi è visto come cifrario a flusso → si genera uno stream di bits ottenuti iterando la cifratura su un vettore di partenza IV quante volte serve e poi applicando sul testo mediante uno XOR.

VANTAGGI: 1) Un errore di 1 bit nel testo cifrato è un errore di 1 bit nella decodifica.

2) Lo stream può essere precalcolato.

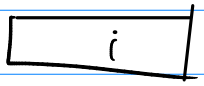
3) Riscrivere un blocco non influenza gli altri blocchi.

SVANTAGGIO FONDAMENTALE

Conoscere un blocco in posizione i

in chiaro vuol dire conoscere

$\forall$ blocco in posizione i



$$c_i = m_i \oplus \mathcal{E}^i(IV, k)$$

$$\mathcal{E}^i(IV, k)$$

2) In posizione $i+1$ abbiamo

$$c_{i+1} = m_{i+1} \oplus \mathcal{E}^{i+1}(IV, k) =$$

$$= \mathcal{E}\left(\mathcal{E}^i(IV, k) \oplus k\right)$$

ALCUNI BLOCCHI IN INGRESSO POTREBBERO ESSERE "DEBOLI"

→ || → le loro orbita sotto l'azione di $\mathcal{E}(\cdot, k)$ è

cont'd.

Calcolare lo stream fino al blocco i richiede calcolare ∀ valore < i. di esso.

using the same IV to be easily recovered from the *j*th ciphertext block of that message.

Confidentiality may similarly be compromised if *any* of the input blocks to the forward cipher function for the encryption of a message is designated as the IV for the encryption of another message under the given key.

The OFB mode is illustrated in Figure 4.

6.5 The Counter Mode

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct. In this recommendation, the counters for a given message are denoted $T_1, T_2, \ldots, T_n$. Methods for generating counters are discussed in Appendix B. Given a sequence of counters, $T_1, T_2, \ldots, T_n$, the CTR mode is defined as follows:

CTR Encryption:

$$O_j = CIPH_K(T_j) \quad \text{for } j = 1, 2 \ldots n;$$
$$C_j = P_j \oplus O_j \quad \text{for } j = 1, 2 \ldots n\text{-}1;$$
$$C^*_n = P^*_n \oplus MSB_u(O_n).$$

CTR Decryption:

$$O_j = CIPH_K(T_j) \quad \text{for } j = 1, 2 \ldots n;$$
$$P_j = C_j \oplus O_j \quad \text{for } j = 1, 2 \ldots n\text{-}1;$$
$$P^*_n = C^*_n \oplus MSB_u(O_n).$$

In CTR encryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-ORed with the corresponding plaintext blocks to produce the ciphertext blocks. For the last block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the exclusive-OR operation; the remaining b-u bits of the last output block are discarded.

In CTR decryption, the forward cipher function is invoked on each counter block, and the resulting output blocks are exclusive-ORed with the corresponding ciphertext blocks to recover the plaintext blocks. For the last block, which may be a partial block of u bits, the most significant u bits of the last output block are used for the exclusive-OR operation; the remaining b-u bits of the last output block are discarded.

In both CTR encryption and CTR decryption, the forward cipher functions can be performed in parallel; similarly, the plaintext block that corresponds to any particular ciphertext block can be recovered independently from the other plaintext blocks if the corresponding counter block can be determined. Moreover, the forward cipher functions can be applied to the counters prior to the availability of the plaintext or ciphertext data.

CTR mode $m_i \oplus E(i,k)$ $E(H(i),k)$

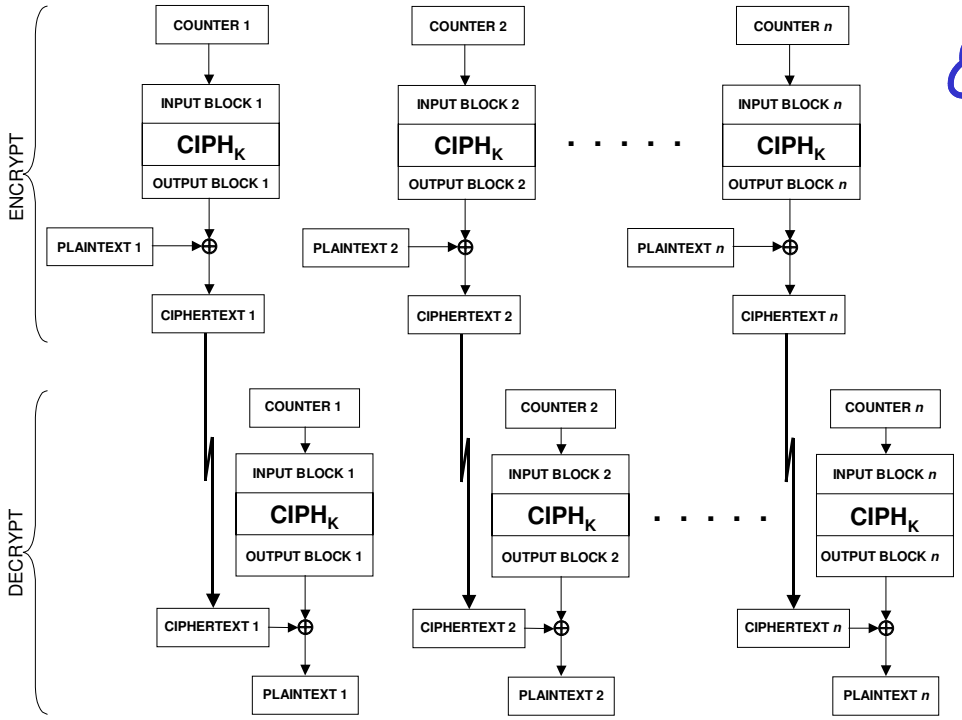
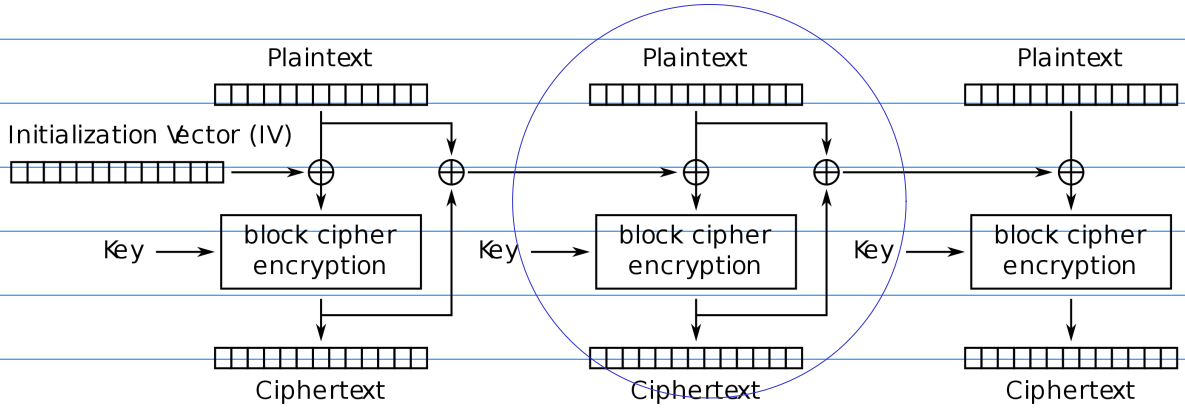


Figure 5: The CTR Mode

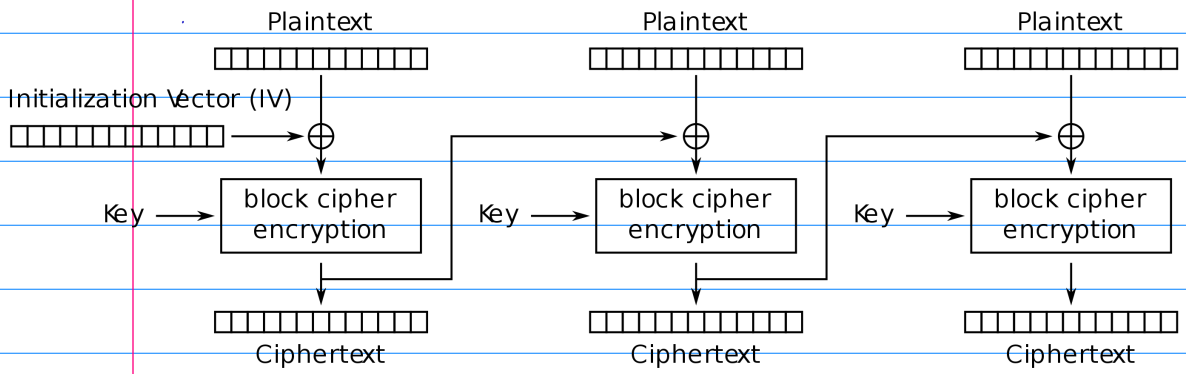
The CTR mode is illustrated in Figure 5.

CTR è parallelo in lettura ed in scrittura.

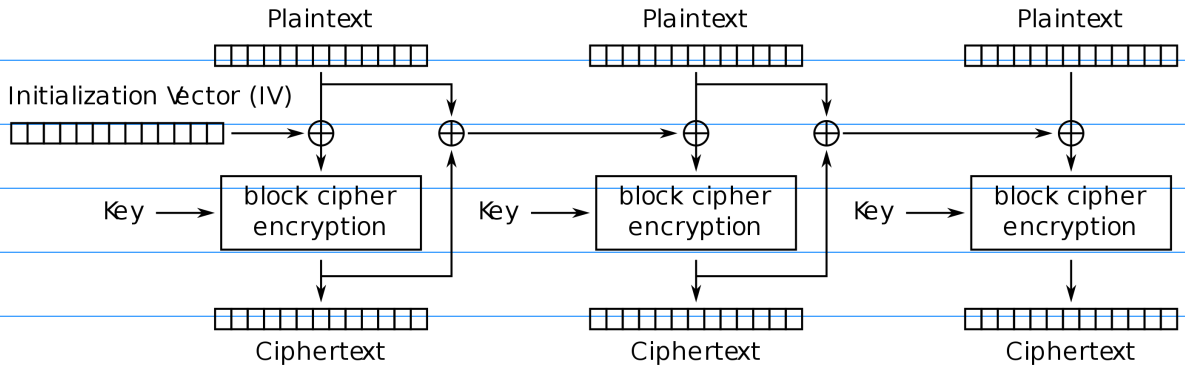


Propagating Cipher Block Chaining (PCBC) mode encryption

non funziona in presenza di errori ed è completamente reversibile.



Cipher Block Chaining (CBC) mode encryption

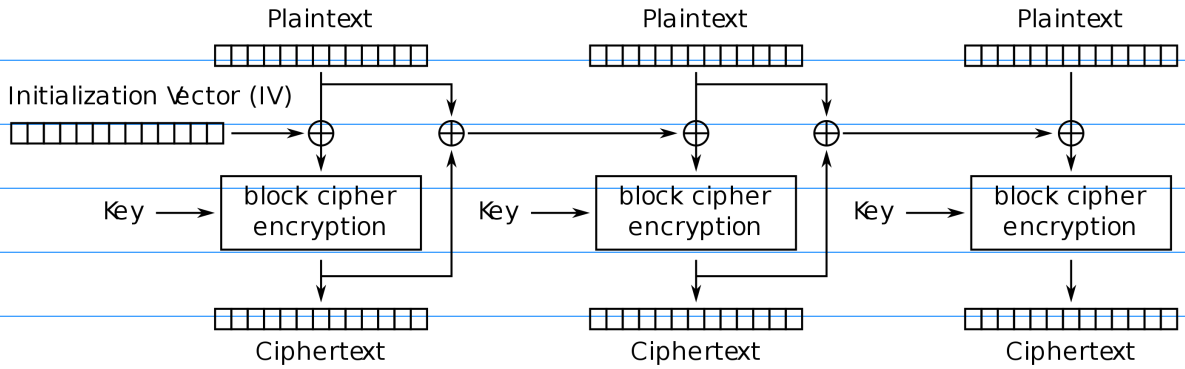


Propagating Cipher Block Chaining (PCBC) mode encryption

C_2

$m_2 = D(c_2) \oplus c_1 \oplus m_1 =$

$=$ ⨁



Propagating Cipher Block Chaining (PCBC) mode encryption

$$m_2 = D(c_2) \oplus c_1 \oplus \boxed{m_1} =$$

$$= D(c_2) \oplus c_1 \oplus \boxed{D(c_1) \oplus IV}$$

per decodificare c_i ci serve avere decodificato c_{i-1}

$\Rightarrow$ c_i è decodificabile $\Leftrightarrow$ abbiamo m_j e c_j per $j < i$.

⇒ un errore in posizione k rende il resto non leggibile $\forall j > k$

Inoltre codifica e decodifica sono possibili solamente in modo seriale.

La cosa che più piacere è che un bit alterato nello stream altera irreversibilmente la decodifica dei bits successivi.

Appendix A: Padding

For the ECB, CBC, and CFB modes, the plaintext must be a sequence of one or more complete data blocks (or, for CFB mode, data segments). In other words, for these three modes, the total number of bits in the plaintext must be a positive multiple of the block (or segment) size.

If the data string to be encrypted does not initially satisfy this property, then the formatting of the plaintext must entail an increase in the number of bits. A common way to achieve the necessary increase is to append some extra bits, called padding, to the trailing end of the data string as the last step in the formatting of the plaintext. An example of a padding method is to append a single '1' bit to the data string and then to pad the resulting string by as few '0' bits, possibly none, as are necessary to complete the final block (segment). Other methods may be used; in general, the formatting of the plaintext is outside the scope of this recommendation.

For the above padding method, the padding bits can be removed unambiguously, provided the receiver can determine that the message is indeed padded. One way to ensure that the receiver does not mistakenly remove bits from an unpadded message is to require the sender to pad every message, including messages in which the final block (segment) is already complete. For such messages, an entire block (segment) of padding is appended. Alternatively, such messages can be sent without padding if, for every message, the existence of padding can be reliably inferred, e.g., from a message length indicator.

Appendix B: Generation of Counter Blocks

The specification of the CTR mode requires a unique counter block for each plaintext block that is ever encrypted under a given key, across all messages. If, contrary to this requirement, a counter block is used repeatedly, then the confidentiality of all of the plaintext blocks corresponding to that counter block may be compromised. In particular, if any plaintext block that is encrypted using a given counter block is known, then the output of the forward cipher function can be determined easily from the associated ciphertext block. This output allows any other plaintext blocks that are encrypted using the same counter block to be easily recovered from their associated ciphertext blocks.

There are two aspects to satisfying the uniqueness requirement. First, an incrementing function for generating the counter blocks from any initial counter block can ensure that counter blocks do not repeat within a given message. Second, the initial counter blocks, T_1, must be chosen to ensure that counters are unique across all messages that are encrypted under the given key.

B.1 The Standard Incrementing Function

In general, given the initial counter block for a message, the successive counter blocks are derived by applying an incrementing function. As in the above specifications of the modes, n is the number of blocks in the given plaintext message, and b is the number of bits in the block.

The standard incrementing function can apply either to an entire block or to a part of a block. Let m be the number of bits in the specific part of the block to be incremented; thus, m is a positive integer such that $m \leq b$. Any string of m bits can be regarded as the binary representation of a non-negative integer x that is strictly less than 2^m. The standard incrementing function takes $[x]_m$ and returns $[x+1 \bmod 2^m]_m$.

For example, let the standard incrementing function apply to the five least significant bits of eight bit blocks, so that b=8 and m=5 (unrealistically small values); let * represent each unknown bit in this example, and let ***11110 represent a block to be incremented. The following sequence of blocks results from four applications of the standard incrementing function:

* * * 1 1 1 1 0
* * * 1 1 1 1 1
* * * 0 0 0 0 0
* * * 0 0 0 0 1
* * * 0 0 0 1 0.

Counter blocks in which a given set of m bits are incremented by the standard incrementing function satisfy the uniqueness requirement *within the given message* provided that $n \leq 2^m$. Whether the uniqueness requirement for counter blocks is satisfied across all messages that are encrypted under a given key then depends on the choices of the initial counter blocks for the messages, as discussed in the next section.

This recommendation permits the use of any other incrementing function that generates n unique strings of m bits in succession from the allowable initial strings. For example, if the initial string of m bits is not the "zero" string, i.e., if it contains at least one '1' bit, then an incrementing function can be constructed from a linear feedback shift register that is specialized to ensure a sufficiently large period; see Ref. [5] for information about linear feedback shift registers.

B.2 Choosing Initial Counter Blocks

The initial counter blocks, T_1, for each message that is encrypted under the given key must be chosen in a manner than ensures the uniqueness of all the counter blocks across all the messages. Two examples of approaches to choosing the initial counter blocks are given in this section.

In the first approach, for a given key, all plaintext messages are encrypted sequentially. Within the messages, the same fixed set of m bits of the counter block is incremented by the standard incrementing function. The initial counter block for the initial plaintext message may be any string of b bits. The initial counter block for any subsequent message can be obtained by applying the standard incrementing function to the fixed set of m bits of the final counter block of the previous message. In effect, all of the plaintext messages that are ever encrypted under the given key are concatenated into a single message; consequently, the total number of plaintext blocks must not exceed 2^m. Procedures should be established to ensure the maintenance of the state of the final counter block of the latest encrypted message, and to ensure the proper sequencing of the messages.

A second approach to satisfying the uniqueness property across messages is to assign to each message a unique string of $b/2$ bits (rounding up, if b is odd), in other words, a message nonce, and to incorporate the message nonce into every counter block for the message. The leading $b/2$ bits (rounding up, if b is odd) of each counter block would be the message nonce, and the standard incrementing function would be applied to the remaining m bits to provide an index to the counter blocks for the message. Thus, if N is the message nonce for a given message, then the jth counter block is given by $T_j = N \mid [j]_m$, for $j = 1 \ldots n$. The number of blocks, n, in any message must satisfy $n < 2^m$. A procedure should be established to ensure the uniqueness of the message nonces.

This recommendation allows other methods and approaches for achieving the uniqueness property. Validation that an implementation of the CTR mode conforms to this recommendation will typically include an examination of the procedures for assuring the uniqueness of counter blocks within messages and across all messages that are encrypted under a given key.

Appendix C: Generation of Initialization Vectors

The CBC, CFB, and OFB modes require an initialization vector as input, in addition to the plaintext. An IV must be generated for each execution of the encryption operation, and the same IV is necessary for the corresponding execution of the decryption operation. Therefore, the IV, or information that is sufficient to calculate the IV, must be available to each party to the communication.

The IV need not be secret, so the IV, or information sufficient to determine the IV, may be transmitted with the ciphertext.

For the CBC and CFB modes, the IVs must be unpredictable. In particular, for any given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV.

There are two recommended methods for generating unpredictable IVs. The first method is to apply the forward cipher function, under the same key that is used for the encryption of the plaintext, to a nonce. The nonce must be a data block that is unique to each execution of the encryption operation. For example, the nonce may be a counter, as described in Appendix B, or a message number. The second method is to generate a random data block using a FIPS-approved random number generator.

For the OFB mode, the IV need not be unpredictable, but it must be a nonce that is unique to each execution of the encryption operation. For example, the nonce may be a counter, as described in Appendix B, or a message number.

If, contrary to this requirement, the same IV is used for the OFB encryption of more than one message, then the confidentiality of those messages may be compromised. In particular, if a plaintext block of any of these messages is known, say, the jth plaintext block, then the jth output of the forward cipher function can be determined easily from the jth ciphertext block of the message. This information allows the jth plaintext block of any other message that is encrypted using the same IV to be easily recovered from the jth ciphertext block of that message.

Confidentiality may similarly be compromised if *any* of the input blocks to the forward cipher function for the OFB encryption of a message is designated as the IV for the encryption of another message under the given key. One consequence of this observation is that IVs for the OFB mode should not be generated by invoking the block cipher on another IV.

Validation that an implementation of the CBC, CFB, or OFB mode conforms to this recommendation will typically include an examination of the procedures for assuring the unpredictability or uniqueness of the IV.

Appendix D: Error Properties

A bit error is the substitution of a '0' bit for a '1' bit, or vice versa. This appendix contains a discussion of the effects of bit errors in ciphertext blocks (or segments), counter blocks, and IVs on the modes in this recommendation. Insertion or deletion of bits into ciphertext blocks (or segments) is also discussed.

For any confidentiality mode, if there are any bit errors in a single ciphertext block (or segment), then the decryption of that ciphertext block (or segment) will be incorrect, i.e., it will differ from the original plaintext block (or segment). In the CFB, OFB, and CTR modes, the bit error(s) in the decrypted ciphertext block (or segment) occur in the same bit position(s) as in the ciphertext block (or segment); the other bit positions are not affected. In the ECB and CBC modes, a bit error may occur, independently, in any bit position of the decrypted ciphertext block, with an expected error rate of fifty percent, depending on the strength of the underlying block cipher.

For the ECB, OFB, and CTR modes, bit errors within a ciphertext block do not affect the decryption of any other blocks. In the CBC mode, any bit positions that contain bit errors in a ciphertext block will also contain bit errors in the decryption of the succeeding ciphertext block; the other bit positions are not affected. In the CFB mode, bit errors in a ciphertext segment affect the decryption of the next b/s (rounded up to the nearest integer) ciphertext segments. A bit error may occur, independently, in any bit position in these decrypted segments, with an expected error rate of fifty percent.

Similarly, for the CTR mode, if there is a bit error in a counter block, then a bit error may occur, independently, in any bit position of the decryption of the corresponding ciphertext, with an expected error rate of fifty percent.

Bit errors in IVs also affect the decryption process. In the OFB mode, bit errors in the IV affect the decryption of every ciphertext block. In the CFB mode, bit errors in the IV affect, at a minimum, the decryption of the first ciphertext segment, and possibly successive ciphertext segments, depending on the bit position of the rightmost bit error in the IV. (In general, a bit error in the ith most significant bit position affects the decryptions of the first i/s (rounding up) ciphertext segments.) For both the OFB and CFB modes, a bit error may occur, independently, in any bit position of the affected ciphertext blocks (or segments), with an expected error rate of fifty percent. In the CBC mode, if bit errors occur in the IV, then the first ciphertext block will be decrypted incorrectly, and bit errors will occur in exactly the same bit positions as in the IV; the decryptions of the other ciphertext blocks are not affected.

Consequently, for the CBC mode, the decryption of the first ciphertext block is vulnerable to the (deliberate) introduction of bit errors in specific bit positions of the IV if the integrity of the IV is not protected. Similarly, for the OFB and CTR modes, the decryption of any ciphertext block is vulnerable to the introduction of specific bit errors into that ciphertext block if its integrity is not protected. The same property also holds for the ciphertext segments in the CFB mode; however, for every ciphertext segment except the last one, the existence of such bit errors may be detected by their randomizing effect on the decryption of the succeeding ciphertext segment.

Table D.1 summarizes the effects of bit errors in a ciphertext block or IV on the decryption of the ciphertext for each of the five confidentiality modes.

Table D.1e five confidentiality modes.

Table D.2: Summary of Effect of Bit Errors on Decryption

Mode	Effect of Bit Errors in C_j	Effect of Bit Errors in the IV
ECB	RBE in the decryption of C_j	Not applicable
CBC	RBE in the decryption of C_j SBE in the decryption of C_{j+1}	SBE in the decryption of C_1
CFB	SBE in the decryption of C_j RBE in the decryption of $C_{j+1},\ldots,C_{j+b/s}$	RBE in the decryption of $C_1, C_2, \ldots, C_j$ for some j between 1 and b/s
OFB	SBE in the decryption of C_j	RBE in the decryption of $C_1, C_2, \ldots, C_n$
CTR	SBE in the decryption of C_j	Not applicable *

RBE: random bit errors, i.e., bit errors occur independently in any bit position with an expected probability of ½.
SBE: specific bit errors, i.e., bit errors occur in the same bit position(s) as the original bit error(s).
* Bit errors in the jth counter block, T_j, result in RBE in the decryption of C_j.

The deletion or insertion of bits into a ciphertext block (or segment) spoils the synchronization of the block (or segment) boundaries; in effect, bit errors may occur in the bit position of the inserted or deleted bit, and in every subsequent bit position. Therefore, the decryptions of the subsequent ciphertext blocks (or segments) will almost certainly be incorrect until the synchronization is restored. When the 1-bit CFB mode is used, then the synchronization is automatically restored b+1 positions after the inserted or deleted bit. For other values of s in the CFB mode, and for the other confidentiality modes in this recommendation, the synchronization must be restored externally.

Appendix E: Modes of Triple DES

FIPS Pub 46-3 [FIPS 46-3] specifies the Data Encryption Standard (DES) algorithm and approves its three-fold, compound operation that is specified in ANSI X9.52 [1]: the Triple Data Encryption Algorithm (TDEA). Essentially, the TDEA consists of the application of the forward DES algorithm, i.e, DES encryption, under one key, followed by the application of the inverse DES algorithm, i.e., DES decryption, under a second key, followed by the application of the forward DES algorithm under a third key. The TDEA is often called Triple DES.

FIPS Pub 46-3 also approves the seven modes of operation of Triple DES that are specified in ANSI X9.52. Four of those modes are equivalent to modes in this recommendation with the TDEA as the underlying block cipher. In particular, the TECB, TCBC, and TOFB modes in ANSI X9.52 are equivalent to the ECB, CBC, and OFB modes in this recommendation, with the TDEA as the underlying block cipher; the TCFB mode in ANSI X9.52 is equivalent to the CFB mode in this recommendation, with the TDEA as the underlying block cipher, provided that the possible choices of the parameter *s* (the segment size) are restricted to three values: 1, 8, and 64. The remaining three modes in ANSI X9.52 are TCBC-I, TCFB-P, and TOFB-I; they are mode variants that allow for interleaving or pipelining; this recommendation does not provide analogues of these three modes.

The Triple DES *modes* in ANSI X9.52 should not be used as the underlying block cipher algorithm for the modes in this recommendation. However, the Triple DES *algorithm*, i.e., TDEA, as described above, may be used as the underlying block cipher algorithm for the six modes in this recommendation. One of the resulting modes of Triple DES is new, i.e., not specified in ANSI X9.52: the CTR mode of the TDEA.

Appendix G: References

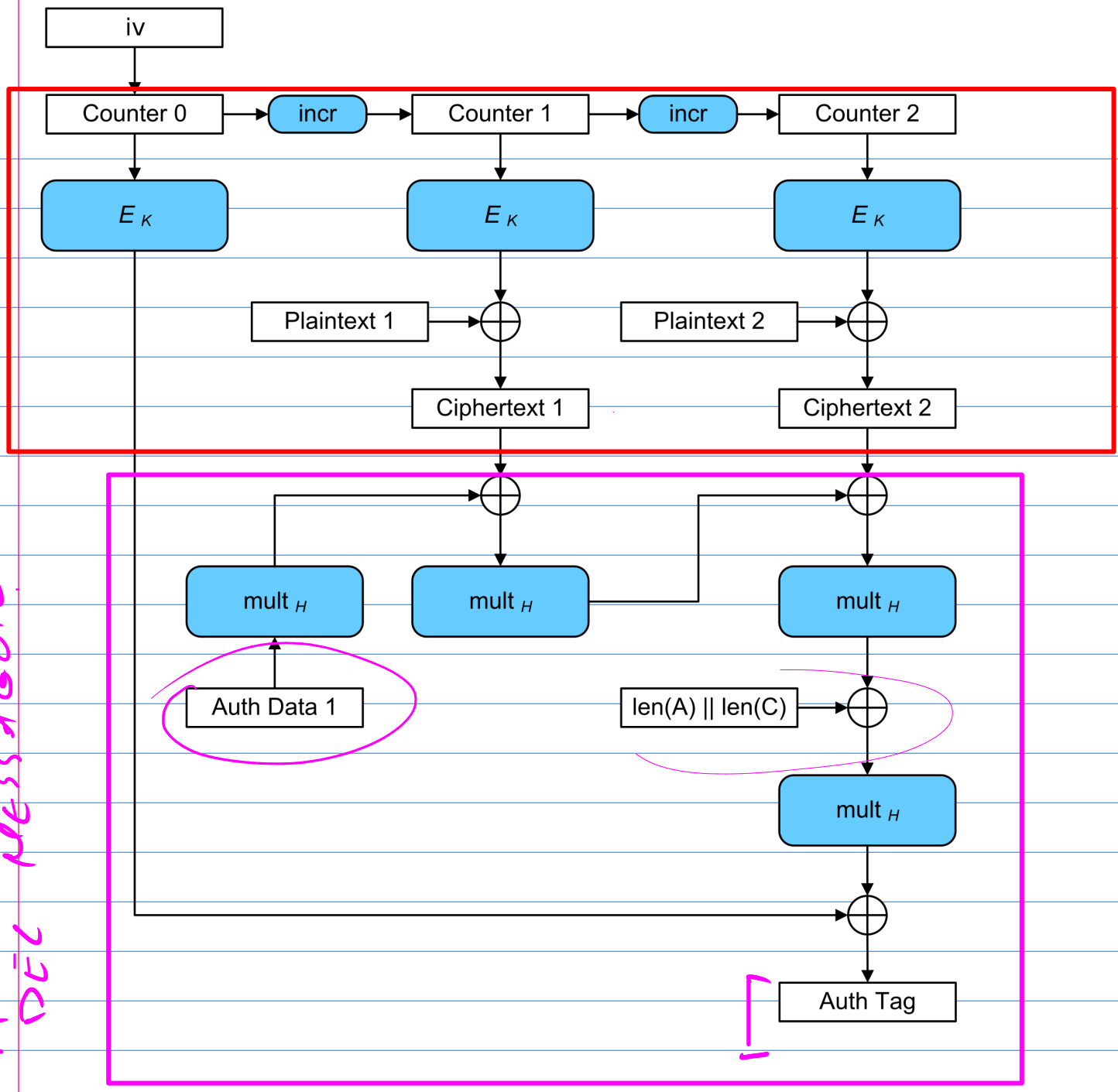
[1] American National Standard for Financial Services X9.52-1998, "Triple Data Encryption Algorithm Modes of Operation." American Bankers Association, Washington, D.C., July 29, 1998.

[2] FIPS Publication 197, "Advanced Encryption Standard (AES)." U.S. DoC/NIST, November 26, 2001.

[3] FIPS Publication 46-3, "Data Encryption Standard (DES)." U.S. DoC/NIST, October 25, 1999.

[4] FIPS Publication 81, "DES Modes of Operation." U.S. DoC/NIST, December 1980.

[5] A. Menezes, P. van Oorschot, and S. Vanstone, "Handbook of Applied Cryptography." CRC Press, New York, 1997.



CTR

AUTENTICAZIONE DEL MESSAGGIO

1) Codificare un messaggio con una chiave k

2) Generare contestualmente un tag di autenticazione → un blocco che attesti che il messaggio è

Visto effettivamente codificato da qualcuno che conosceva la chiave K.

per la costruzione del TAG.

consideriamo i blocchi di AES.

sono sequenze di 128 bits.

(visti come array di 16 bytes).

possiamo costruire il campo

$$\mathbb{F}_{2^{128}} \cong \frac{\mathbb{F}_2[x]}{(p(x))}$$

con $\deg p(x) = 16$

$$\mathbb{F}_{256} = \frac{\mathbb{Z}_2[x]}{(m(x))}$$

$$\deg m(x) = 8$$

Ogni blocco di AES può essere letto come un elemento di un campo ove la somma è lo XOR.

$$H = E(\underline{0}^{16}, K)$$

parto dal reg. iniziale

$$T_0 \rightarrow T_0 \cdot H + C_1 = T_1$$

$$T_1 \rightarrow T_2 \cdot H + C_2 \dots$$

$$T_n = T_{n-1} \cdot H + C_n$$

TAG FINALE:

$$\left[T^* = \left(T_n + [\text{Rappr. lunghezza } C \text{ ed } A] \right) \cdot H + \varepsilon(IV, k) \right]$$

Nel calcolo di T^* stesso di conoscere la chiave k ed è legata di testi cifrati. |

---

codifica di dati per dischi rigidi.

ABBIAMO DEI PREREQUISITI.

1) Accesso casuale ai dati (R/W)
2) Bassa latenza
3) Resistenza agli errori.

↳ pensare ad una sfida cipher

prendiamo CTR mode per un cifrario a blocchi: vediamo il disco come un array con un indice lineare. Il blocco i è enumerizzato come $E(i,k) \oplus m_i$

ph: se riscriviamo il blocco viene usata la stessa codifica per i dati.

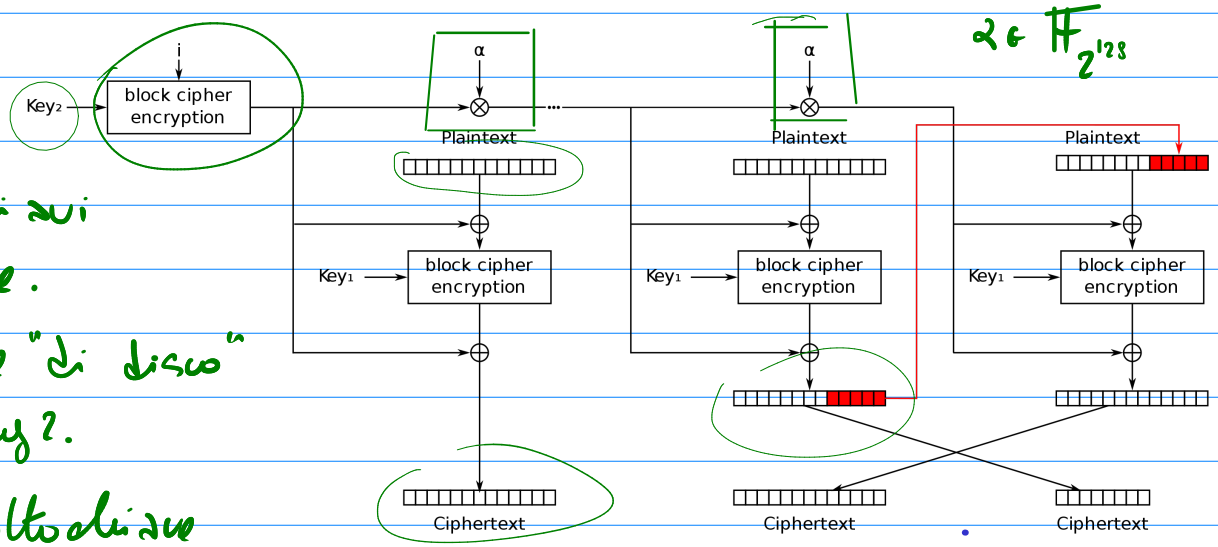
$$E(i,k).$$

$$m \longmapsto c = m \oplus E(i,k)$$

ricavo m da c conoscendo $E(i,k)$.

"Assomiglia a CTR mode ma in cui non si usa lo XOR direttamente"

XTS mode



$\alpha \in \mathbb{F}_{2^{128}}$

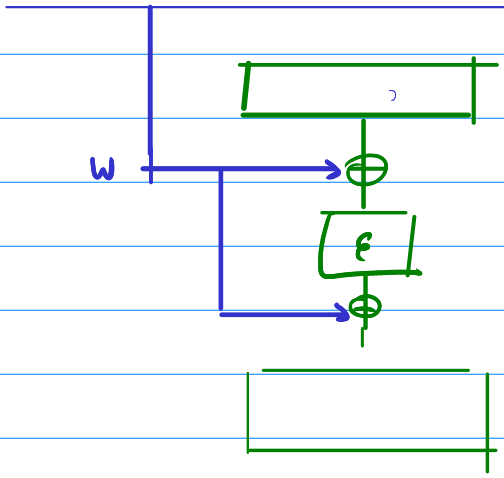
XEX with tweak and ciphertext stealing (XTS) mode encryption

più chiavi coinvolte.

1) chiave "di disco" key2.

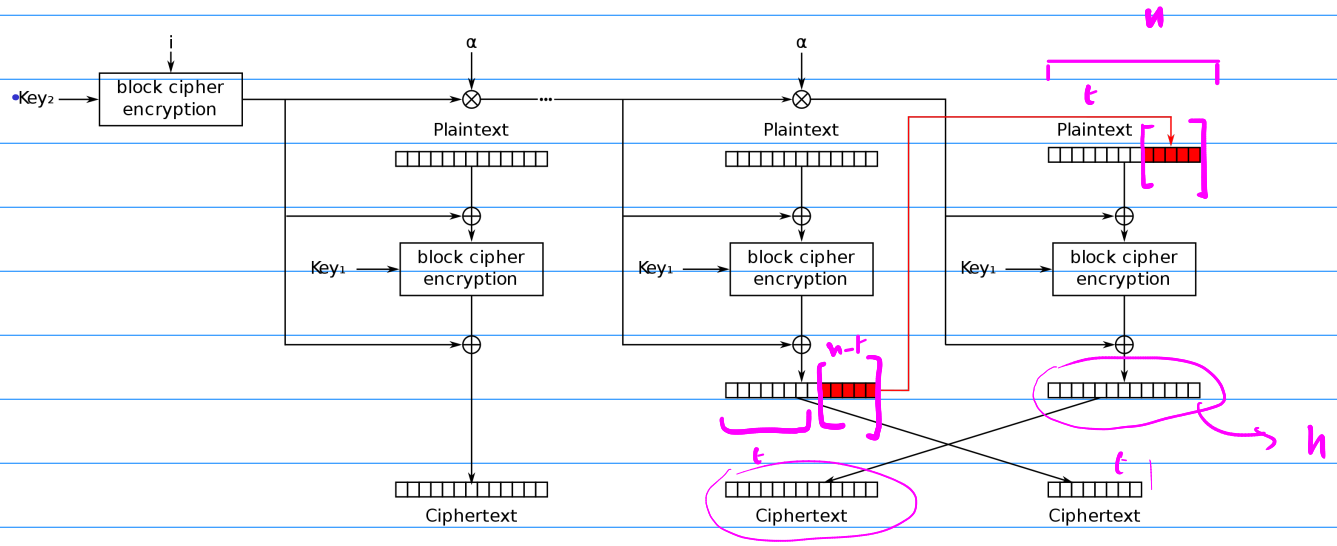
2) una sottochiave key1 relativa i files o comunque le zone del disco stesso.

XEX



$m = 0 \Rightarrow c = E(w, k_2) \oplus w$

$c' = E(m \oplus w, k_2) \oplus w$



XEX with tweak and ciphertext stealing (XTS) mode encryption

Trucco per avere che nell'ultimo blocco ci siano esattamente t bits reali anche se la codifica lavora su blocchi di lunghezza $n > t$