Agenzia per la
Cybersicurezza Nazionale

LINEE GUIDA FUNZIONI CRITTOGRAFICHE

Cifrari a Blocchi e Modalità di Funzionamento

MAGGIO 2026

Versione	Data di pubblicazione	Note
1.0	**11/07/2024**	**Prima pubblicazione**
1.1	**12/05/2026**	**Aggiunta sezione “Scopo del documento”, ulteriori modifiche minori**

Sommario

pag.

Indice delle figure

pag.

Indice delle tabelle

Scopo del documento

Questo documento costituisce parte della serie "Linee Guida Funzioni Crittografiche" e fornisce le raccomandazioni di ACN in merito ai **cifrari a blocchi** e alle loro **modalità di funzionamento**, da adottare in tutti i contesti in cui si necessita di garantire la confidenzialità dei dati. Per ulteriori informazioni sulle Linee Guida e sulle utilità delle funzioni crittografiche in base ai contesti specifici, si faccia riferimento al documento introduttivo della serie [1].

Ogni documento tiene in considerazione le minacce presenti al giorno della sua pubblicazione. Data la diversa natura dei sistemi informativi di destinazione, non è possibile garantire che queste raccomandazioni possano essere utilizzate senza adattamenti specifici. In qualsiasi caso, la pertinenza dell'attuazione delle soluzioni proposte deve essere sottoposta, preventivamente, a valutazione e validazione da parte dei responsabili della sicurezza dei sistemi informativi di destinazione.

I contenuti delle Linee Guida sono indirizzati a sviluppatori, produttori di dispositivi e fornitori di servizi digitali, al fine di promuovere l'utilizzo di primitive crittografiche e relativi parametri di configurazione sicuri fin dalla fase di progettazione di prodotti, reti, applicazioni e servizi. Questi documenti sono altresì rivolti a security manager, referenti e responsabili della cybersicurezza di soggetti pubblici e privati affinché verifichino che i sistemi utilizzati dalla propria organizzazione siano conformi alle raccomandazioni fornite.

La legge 28 giugno 2024, n. 90 relativa a "Disposizioni in materia di rafforzamento della cybersicurezza nazionale e di reati informatici", all'articolo 9, stabilisce a tal fine che *«le strutture di cui all'articolo 8 della presente legge nonché quelle che svolgono analoghe funzioni per i soggetti di cui all'articolo 1, comma 2-bis, del decreto-legge 21 settembre 2019, n. 105, convertito, con modificazioni, dalla legge 18 novembre 2019, n. 133, e al decreto legislativo 18 maggio 2018, n. 65, verificano che i programmi e le applicazioni informatiche e di comunicazione elettronica in uso, che utilizzano soluzioni crittografiche, rispettino le linee guida sulla crittografia nonché quelle sulla conservazione delle password adottate dall'Agenzia per la cybersicurezza nazionale e dal Garante per la protezione dei dati personali e non comportino vulnerabilità note, atte a rendere disponibili e intellegibili a terzi i dati cifrati»*.

Il documento è stato curato dal Centro Nazionale di Crittografia istituito presso ACN.

Lista dei simboli matematici utilizzati

$\{0,1\}$	Campo binario dei valori assumibili da un singolo bit	Δ_B	Differenziale tra due vettori, cioè $B \oplus B' = \Delta_B$
$\{0,1\}^n$	Spazio vettoriale delle stringhe binarie di lunghezza n	$\ll k$	Rotazione a sinistra di k posizioni con reinserimento
$\oplus$	Operazione XOR, ovvero la somma bit a bit tra stringhe binarie	C^*	Blocco ottenuto scartando gli ultimi d bit dal blocco C
$\|$	Concatenazione di stringhe	$\otimes$	Moltiplicazione nel campo finito con 2^{128} elementi

1 Introduzione

La richiesta più immediata che viene posta alla crittografia è quella di assicurare la confidenzialità dei messaggi scambiati tra due o più interlocutori: gli algoritmi di cifratura e decifratura entrano in gioco quando si vuole mantenere la segretezza di una comunicazione nei confronti di altre entità potenzialmente malevole.
I cifrari a blocchi, nati per assolvere a tale funzione, sono tra i metodi di crittografia più diffusi al giorno d'oggi e sono ampiamente impiegati nell'ambito delle comunicazioni digitali. Assieme ai cifrari a flusso, trattati nel documento dedicato [2], costituiscono i cifrari simmetrici: sistemi che utilizzano in fase di cifratura e di decifratura la stessa chiave, la quale deve perciò essere nota solamente ai legittimi interlocutori ma tenuta segreta a chiunque altro.
La caratteristica principale dei cifrari a blocchi, come evidenzia anche il nome, è quella di operare su porzioni del testo in chiaro di lunghezza fissata, chiamate appunto blocchi. Questa peculiarità li rende adattabili a molti sistemi ed esistono diverse modalità di funzionamento che descrivono come si utilizzi la funzione di cifratura.
Il documento presenta la seguente struttura: nel capitolo 2 si introducono le diverse tipologie di cifrari a blocchi e le definizioni principali. In seguito, nel capitolo 3, vengono descritte le principali tipologie di attacco, evidenziandone anche le relazioni matematiche. I cifrari a blocchi raccomandati e le modalità di funzionamento vengono trattati, rispettivamente, nel capitolo 4 e nel capitolo 5, mentre la pratica del padding viene descritta in dettaglio nel capitolo 6. Infine, le raccomandazioni e le conclusioni sono raccolte nel capitolo 7.

2 Cifrari a blocchi

Un **cifrario a blocchi** è un sistema crittografico simmetrico che permette a due interlocutori di scambiarsi dei messaggi assicurandone la **confidenzialità**, ossia impedendo a un attore terzo che venga in possesso del dato scambiato di conoscere il contenuto della conversazione.
Per poter soddisfare queste caratteristiche, i cifrari a blocchi utilizzano una stringa K lunga k bit, detta **chiave**, che deve essere nota solamente agli interlocutori. La funzione di cifratura prende un input di lunghezza n e restituisce un cifrato della stessa lunghezza. Dato che la lunghezza in input è solitamente minore di quella dell'intero messaggio da cifrare, esso viene prima suddiviso in porzioni di lunghezza n, chiamate **blocchi**, e poi cifrato blocco per blocco secondo una delle modalità presentate nel capitolo 5.
Più nel dettaglio, l'algoritmo di cifratura di un cifrario a blocchi può essere formalizzato dal punto di vista matematico come una funzione tra stringhe binarie

$$E : \{0,1\}^k \times \{0,1\}^n \longrightarrow \{0,1\}^n .$$

Se si fissa la chiave K, la funzione di cifratura può essere rappresentata tramite una funzione invertibile (o permutazione) tra due spazi della stessa dimensione n specificata tra i parametri del cifrario, in formule

$$E_K : \{0,1\}^n \longrightarrow \{0,1\}^n .$$

In altre parole, i blocchi in chiaro vengono trasformati univocamente tramite E_K in blocchi cifrati, in modo tale che solo chi è in possesso della chiave possa ricostruire il messaggio di partenza tramite la permutazione inversa, cioè applicando la funzione di decifratura D_K.
Nelle sezioni seguenti si descrivono le componenti dei cifrari a blocchi e le loro strutture principali.

È fondamentale che la chiave venga scambiata precedentemente in modo sicuro e che venga conservata in maniera adeguata: deve essere praticamente impossibile per un attaccante intercettarla o ricostruirla [3, 4]. Per ulteriori informazioni riguardo ai metodi di scambio della chiave, si rimanda al documento dedicato*.

*In fase di pubblicazione.

2.1 Utilizzo della chiave

La chiave segreta di un cifrario a blocchi, detta **master key**, viene utilizzata in una funzione di **key schedule** per generare più chiavi intermedie, dette **chiavi di round**.
I dettagli di questa procedura sono solitamente pubblici, per cui la sicurezza del cifrario viene affidata alla sola segretezza della master key. I primi cifrari a blocchi moderni utilizzavano chiavi di 64 bit, lunghezza troppo breve per resistere ad attacchi effettuati con i computer odierni. Per questo motivo, la maggior parte dei cifrari attuali utilizza chiavi con una lunghezza minima di 128 bit e, in alcuni casi, sono previste versioni con lunghezze della chiave maggiori. Negli schemi più comuni, le chiavi di round ottenute tramite il key schedule vengono combinate con gli stati intermedi tramite uno XOR ($\oplus$).

2.2 Round

La maggior parte dei cifrari a blocchi ha una struttura di **algoritmo iterativo**, nel quale le stesse operazioni vengono ripetute più volte. In particolare, queste sono organizzate in **round** sequenziali, per cui l'output alla fine di un round diventa l'input del round successivo. Il numero r di round di un cifrario è uno dei parametri globali del cifrario stesso. Talvolta, tale valore varia in base alle diverse versioni dello stesso cifrario, ma viene sempre fissato prima del suo utilizzo.
In ogni round del cifrario viene utilizzata almeno una delle chiavi di round ottenute tramite il key schedule a partire dalla master key. Solitamente vengono generate r o $r+1$ chiavi di round, in quanto si può optare per l'esecuzione, alla fine o all'inizio del cifrario, di un round anomalo chiamato **whitening**. Questo round particolare consiste nel calcolare lo XOR con una delle chiavi di round ed effettuare solo alcune delle operazioni previste in un round completo (in alcuni casi, nessuna). Lo scopo di questo round extra è di incrementare ulteriormente la sicurezza del cifrario.

2.3 S-box

Le **S-box** servono a introdurre confusione tra i dati. Si tratta di funzioni altamente non lineari che solitamente vengono rappresentate come tabelle, dette **look-up table**, in cui vengono riportati gli output in base a tutti i possibili input. Ad esempio, la Tabella 1 rappresenta la look-up table relativa alla S-box del cifrario PRESENT [5], contenente input e output in formato esadecimale. In generale, l'utilizzo di look-up table nell'implementazione di un cifrario viene sconsigliato poiché potrebbero creare delle vulnerabilità alla sicurezza, tramite attacchi di tipo side-channel, come descritto nella sezione 3.3. In generale, le S-box utilizzate all'interno di un cifrario possono essere tutte uguali, oppure variare in base all'ordine in cui vengono applicate. Poiché gli utilizzi di più S-box sono tra loro indipendenti, dati diversi possono essere elaborati parallelamente in modo da ridurre il costo computazionale.

Da un punto di vista matematico, una S-box è una funzione booleana vettoriale [6] che deve possedere determinate caratteristiche per poter resistere agli attacchi più comuni, riportati in seguito. Una corretta specifica di un cifrario deve riportare queste informazioni, in modo da permettere alla comunità scientifica di analizzare la presenza di eventuali vulnerabilità prima che il cifrario venga approvato per l'uso.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Tabella 1 - S-box di PRESENT

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Tabella 2 - P-box di DES

2.4 Mixing layer

A differenza delle S-box, il **mixing layer** o **P-box** è, solitamente, una funzione lineare, cioè una permutazione dei bit all'interno dello stato. Lo scopo di questa componente è quello di coinvolgere più S-box nel successivo round di cifratura, diffondendo l'informazione contenuta nel blocco ad ogni round. A differenza delle S-box, le P-box consistono in un riordinamento dei bit dello stato e sono spesso rappresentati da vettori in cui vengono indicate le posizioni dei nuovi bit. In Tabella 2 è riportata la P-box di DES (Data Encryption Standard) [7], standard dismesso nel 2002, la quale opera su 32 bit. La tabella si interpreta in questo modo: il primo bit del nuovo stato assumerà il valore del 16-esimo bit dello stato precedente, il secondo quello del settimo e così via. Finita la prima riga, si ha la 17-esima posizione che assume il valore della numero 2 e si continua.

2.5 Cifrario di Feistel

Una delle più comuni strutture per i cifrari a blocchi è quella dei **cifrari di Feistel** [8], la quale prende il nome da Horst Feistel che la progettò nel 1973 per il cifrario Lucifer di IBM. La struttura di un cifrario di Feistel è rappresentata in Figura 1 ed è descritta di seguito. Siano $K_1, K_2, \ldots, K_r$ le chiavi di round e F una funzione non lineare (non necessariamente invertibile). Inoltre, il messaggio iniziale e gli stati intermedi si considerano divisi in due parti L_i e R_i, con $0 \leq i \leq r$. Partendo dal testo in chiaro $M = L_0 \,\|\, R_0$, ogni round i viene svolto nel seguente modo:

- si lavora su L_{i-1}, R_{i-1} provenienti dal round precedente, solitamente di uguale lunghezza;
- la nuova parte sinistra assume il valore della vecchia parte destra, cioè $L_i = R_{i-1}$;
- la nuova parte destra è lo XOR tra la vecchia parte sinistra e la parte destra combinata con la chiave di round tramite la funzione F, cioè $R_i = L_{i-1} \oplus F\left(R_{i-1}, K_i\right)$.

Una volta effettuati tutti i round, l'ultimo output $L_r \,\|\, R_r$ rappresenta il testo cifrato.
Il grande vantaggio di questo tipo di cifrari è che, come si osserva in Figura 1, la fase di cifratura e quella di decifratura presentano la stessa struttura. L'unica differenza consiste nell'ordine di utilizzo delle chiavi di round, che nella decifratura è inverso rispetto a quello nella cifratura. Alcuni esempi di cifrari di Feistel sono Data Encryption Standard (DES) [7, 9], Camellia [9, 10], Twofish [11], Kasumi [12].

2.6 Substitution-Permutation Network (SPN)

Più recentemente rispetto ai cifrari di Feistel è stata introdotta una struttura sequenziale chiamata **Substitution-Permutation Network** (SPN). L'idea alla base di questa tipologia di cifrari è la diretta applicazione dei principi stabiliti da Claude Shannon [13], implementando la confusione dell'informazione tramite S-box (che corrispondono a sostituzioni) e ottenendo la diffusione mediante P-box (cioè permutazioni).
Preliminarmente, si applica il whitening, che prevede lo XOR tra il blocco B del testo in chiaro e la prima chiave di round. Il primo round prende in input il valore risultante, mentre ogni round successivo lavora sull'output del round precedente. Nel round i si effettuano le seguenti operazioni:

- l'input viene diviso in parti di uguale lunghezza (detti **brick**);
- le singole parti vengono utilizzate come input di più S-box, applicate in parallelo;
- gli output risultanti vengono concatenati per formare un unico input per un mixing layer P;
- viene calcolato lo XOR tra il risultato parziale e la chiave di round K_i.

Solitamente, l'ultimo round differisce dai precedenti in quanto non prevede la P-box.
Il processo di cifratura è schematizzato nella parte sinistra di Figura 2. La decifratura, illustrata nella parte destra, avviene eseguendo le operazioni a ritroso ma, a differenza dei cifrari di Feistel, si devono applicare le funzioni inverse ed è quindi fondamentale che S-box e P-box siano invertibili. Alcuni cifrari famosi che utilizzano questa struttura sono Advanced Encryption Standard (AES) [9, 14], Serpent [15], PRESENT [5].

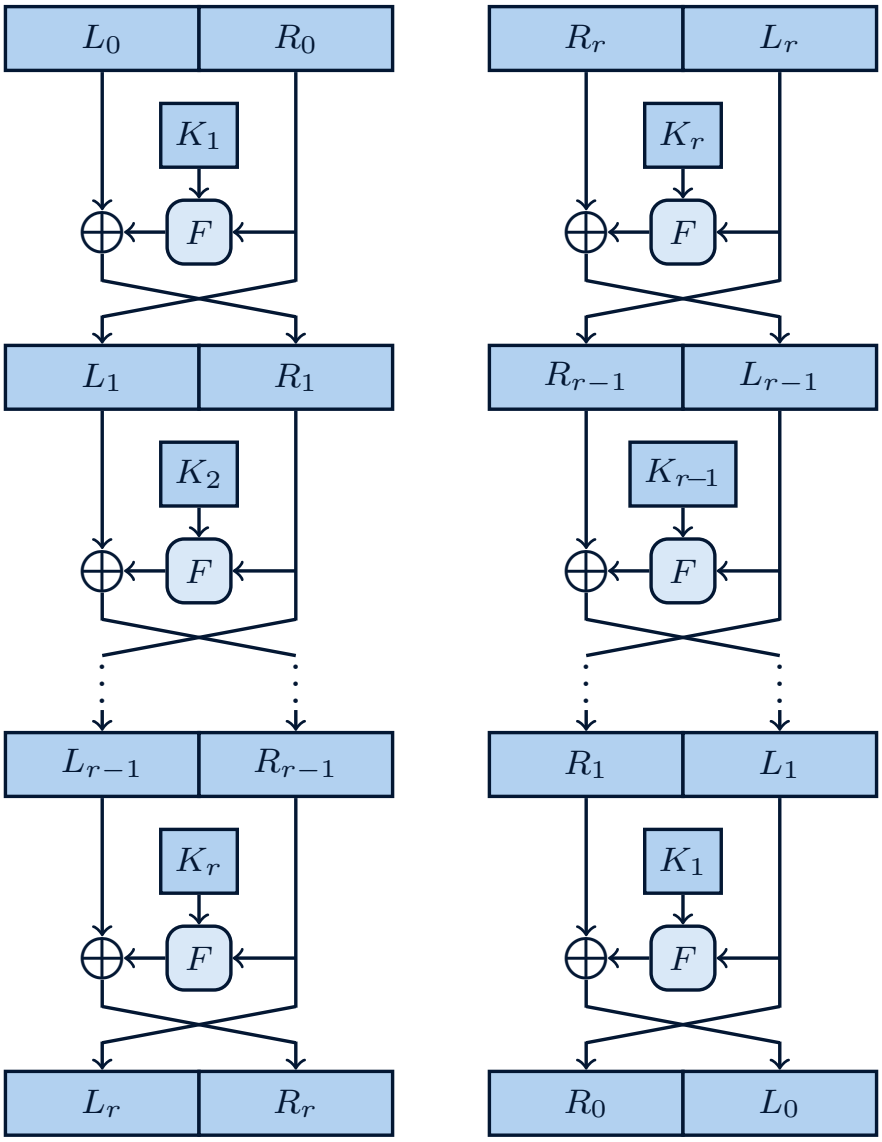

Figura 1 - Cifrario di Feistel, cifratura a sinistra e decifratura a destra

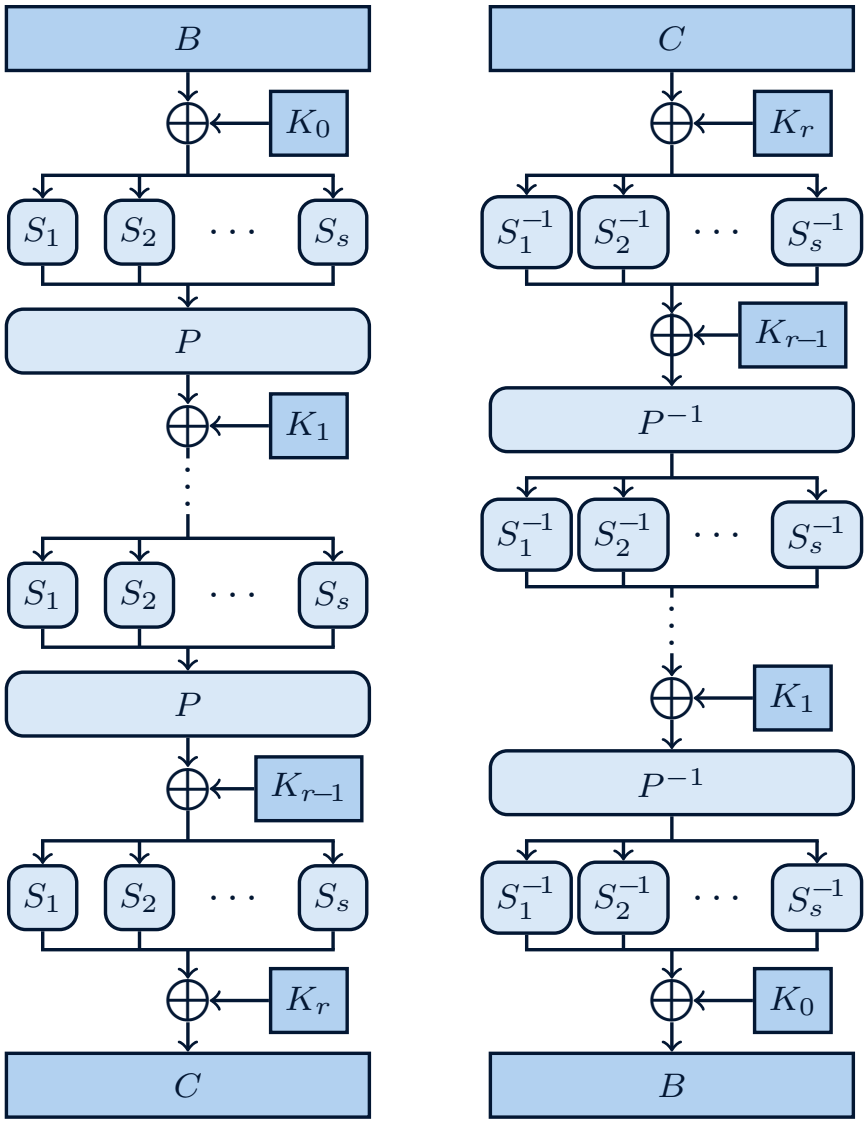

Figura 2 - Schema di una SPN, cifratura a sinistra e decifratura a destra

2.7 Cifratura autenticata

Gli algoritmi di cifratura autenticata sono particolari meccanismi crittografici che permettono di ottenere, oltre alla confidenzialità, sia l'autenticazione del mittente sia la verifica dell'integrità del messaggio, combinando le funzioni dei cifrari a blocchi e dei codici di autenticazione del messaggio (MAC) [4]. L'autenticazione è solitamente ottenuta attraverso la generazione di un **tag**, creato a partire dal messaggio ed eventualmente da dei dati aggiuntivi, attraverso l'utilizzo di una chiave segreta condivisa, solitamente diversa da quella utilizzata dal cifrario a blocchi. Il tag viene quindi inviato insieme al messaggio cifrato così che il destinatario, utilizzando la chiave segreta condivisa, possa verificare l'integrità del messaggio e assicurare l'identità del mittente, per poi decifrare il messaggio tramite l'utilizzo della chiave di cifratura.
Oltre ad algoritmi specifici, la cifratura autenticata può essere ottenuta adottando delle specifiche modalità di funzionamento per i cifrari a blocchi.
A meno di casi eccezionali, si raccomanda sempre l'utilizzo di una soluzione che permetta la cifratura autenticata, anche quando si vuole assicurare solamente la confidenzialità di un messaggio.
Per maggiori dettagli e per le raccomandazioni relative alla cifratura autenticata si rimanda al documento dedicato [16].

2.8 Sistemi lightweight

Le richieste di memoria e di capacità computazionali necessarie per l'implementazione di un cifrario a blocchi sono, a volte, troppo onerose per sistemi con risorse limitate, come ad esempio dispositivi IoT o smart card. In questi casi sono necessari dei cifrari dedicati che riescano comunque a garantire un livello di sicurezza adeguato ai dati da proteggere. Questo tipo di cifrari è chiamato **lightweight**. Una recente competizione del NIST [17] ha selezionato come nuovo standard lightweight Ascon [18], descritto nel dettaglio nel documento dedicato alla cifratura autenticata [16]. Oltre a questo, PRESENT [5] è stato uno dei primi esempi di cifratura lightweight, ma sono state sviluppate diverse soluzioni, come lo standard sudcoreano HIGHT [19].

Se memoria e potenza computazionale lo permettono, si raccomanda di utilizzare i cifrari raccomandati in questo documento. I cifrari lightweight sono valide soluzioni solo in caso di **limitazioni obbligate**, ma in questi contesti se ne consiglia l'utilizzo unitamente a un metodo per la **verifica di integrità e autenticazione**.

3 Attacchi ai cifrari a blocchi

I cifrari a blocchi sono tra i cifrari più utilizzati e una delle prime tecnologie implementate nella crittografia moderna. Per questi motivi, sono sorte numerose strategie di attacco che sfruttano principalmente le debolezze relative alle S-box e alle funzioni booleane che le rappresentano. Oltre a questi **attacchi teorici**, esistono attacchi alle implementazioni dei cifrari che sfruttano le caratteristiche tecniche dell'hardware, chiamati **attacchi side-channel**. Nel seguito verranno descritte brevemente le principali strategie di attacco e le proprietà che cifrari e S-box devono rispettare per resistere a tali attacchi.

3.1 Crittoanalisi lineare

La crittoanalisi lineare, attribuita a Mitsuru Matsui per via di un suo attacco al cifrario FEAL nel 1992 [20], è una forma di crittoanalisi nel contesto **Known Plaintext Attack** (KPA) che sfrutta possibili relazioni lineari tra i bit del testo in chiaro, del testo cifrato e di una chiave di round, al fine di ricavare quest'ultima. La crittoanalisi lineare si è rapidamente diffusa ed è stata applicata a numerosi cifrari, come ad esempio a DES [21], in un attacco dello stesso Matsui negli anni '90.
La crittoanalisi lineare si concentra sullo studio delle relazioni del tipo

$$b_1 \oplus \ldots \oplus b_l \oplus c_1 \oplus \ldots \oplus c_m = k_1 \oplus \ldots \oplus k_n,$$

dove $b_1, \ldots, b_l$ sono alcuni bit del testo in chiaro, $c_1, \ldots, c_m$ sono alcuni bit del testo cifrato e $k_1, \ldots, k_n$ sono alcuni bit della chiave di round. L'obiettivo è trovarne una che valga con una probabilità che sia il più distante possibile da quella casuale che, trattandosi di una relazione tra bit, è pari al 50%. Una volta trovata una relazione di questo tipo, essa può essere utilizzata per recuperare alcuni bit della chiave di round. Il processo viene quindi ripetuto fino a trovare un numero di relazioni che consenta di scoprire i restanti bit della chiave tramite forza bruta. L'attacco viene poi iterato riducendo successivamente i round del cifrario, fino a ricavare tutte le chiavi di round.
Affinché l'attacco abbia successo, l'attaccante deve riuscire ad ottenere una quantità sufficiente di coppie di testo in chiaro e corrispondente testo cifrato, come solitamente avviene nel contesto KPA.
La resistenza alla crittoanalisi lineare è principalmente condizionata dalla scelta della S-box, in quanto uniche componenti non lineari all'interno del cifrario. In particolare, la capacità di una funzione booleana di resistere agli attacchi lineari è misurata dalla sua **nonlinearità**, cioè la distanza minima della funzione da una qualsiasi approssimazione affine (cioè una funzione polinomiale al più di primo grado). Se si utilizza una S-box con nonlinearità molto bassa, è possibile sostituire tale funzione con la sua approssimazione affine e ottenere così un attacco lineare che consente di ricavare le chiavi di round.

3.2 Crittoanalisi differenziale

La crittoanalisi differenziale è, insieme a quella lineare, uno dei metodi di crittoanalisi più diffusi applicabili ai cifrari a blocchi, ma anche a cifrari a flusso o a funzioni di hash.
La scoperta di questa tecnica viene attribuita a Eli Biham e Adi Shamir, che la adoperarono nei primi anni '90 per attaccare diversi cifrari, tra cui DES [22].
La crittoanalisi differenziale studia la propagazione di differenze in input nelle corrispondenti differenze in output. In particolare, fissata una differenza in input Δ_B, si possono considerare due testi in chiaro B, B' che differiscono di Δ_B, cioè $B' = B \oplus \Delta_B$, con rispettivi testi cifrati C e C' aventi una differenza in output $\Delta_C = C \oplus C'$. L'attacco ha successo se si riesce a trovare un **differenziale**, ovvero una coppia di differenze (Δ_B, Δ_C), che si verifica frequentemente al variare delle coppie di testi in chiaro B, B'. Infatti, a causa delle componenti non lineari, la propagazione di una differenza in input non avviene in modo deterministico e quindi alcune differenze risultano più probabili di altre.
Ottenuta una coppia di differenze in input e in output statisticamente rilevante, è possibile effettuare un **Chosen Plaintext Attack** (CPA), ossia un attacco che richiede la conoscenza di un certo numero di coppie testo in chiaro-testo cifrato, dove il testo in chiaro viene scelto dall'attaccante. In particolare, l'attaccante avrà bisogno di conoscere la cifratura di una serie di coppie di testi che differiscono di Δ_B. Come nella crittoanalisi lineare, effettuando con successo un attacco differenziale si otterrà una chiave di round, riducendo di volta in volta i round del cifrario.
Come per il caso lineare, la resistenza agli attacchi differenziali deriva da una corretta scelta delle componenti non lineari del cifrario, ovvero, nel caso dei cifrari a blocchi, delle S-box. Per misurare l'efficacia di questo tipo di attacchi si considera la seguente definizione: sia F una funzione booleana con input di n bit e output di m bit, F si dice **δ-uniforme differenzialmente** (differentially δ-uniform) se l'equazione

$$F(x \oplus a) \oplus F(x) = b$$

ammette al più un numero di soluzioni pari a δ, al variare di a tra tutti i possibili vettori non nulli di n bit e al variare di b tra tutti i possibili vettori di m bit.

Il valore δ è sempre pari e può assumere il valore minimo 2 solo in caso di permutazioni ($n = m$).
Più il valore di δ è piccolo, più la funzione booleana è resistente agli attacchi differenziali, per cui le funzioni ottimali sono quelle con $\delta = 2$, che vengono chiamate **Almost Perfect Nonlinear** (APN). Si può dimostrare che è sempre possibile trovare funzioni APN quando la dimensione $n = m$ è dispari, mentre il caso pari, di gran lunga più utile alle applicazioni crittografiche, è molto più complesso. Infatti, non esistono funzioni APN di dimensione 4, se ne conosce una sola di dimensione 6, mentre per permutazioni di dimensione 8 non esistono al momento dimostrazioni dell'esistenza o meno di funzioni APN. Per ulteriori approfondimenti sulle funzioni booleane si rimanda a [23].

Tanto per l'attacco lineare quanto per quello differenziale, la probabilità di successo può essere ridotta con un mixing layer che effettua una diffusione adeguata. Infatti, nel caso di attacco lineare, questa abbassa notevolmente la probabilità di trovare una relazione lineare efficiente, mentre, nel caso di attacco differenziale, rende difficile recuperare un differenziale significativo.

3.3 Attacchi side-channel

Un cifrario può essere sicuro dal punto di vista teorico, ma avere dei problemi in una sua implementazione.
Gli attacchi che sfruttano tali debolezze vengono detti **side-channel**.
A differenza degli attacchi introdotti finora, che si basano sulle proprietà matematiche del sistema e quindi sono rivolti al cifrario in generale, questi si applicano a particolari implementazioni del cifrario.
È quindi fondamentale porre molta attenzione nelle implementazioni ad-hoc o fai-da-te, ed è preferibile adottare le implementazioni presenti in librerie approvate in quanto già testate e sicure.

Al fine di ottenere un livello di sicurezza adeguato, le implementazioni dei cifrari a blocchi utilizzano spesso delle contromisure per far fronte agli attacchi di tipo side-channel. È quindi fondamentale porre molta attenzione nelle implementazioni ad-hoc o fai-da-te, ed è preferibile adottare le implementazioni presenti nelle librerie crittografiche in quanto già testate e sicure.

Il principio alla base degli attacchi side-channel è che gli algoritmi lasciano irrimediabilmente una traccia quando le componenti fisiche vengono utilizzate, come ad esempio quando si accede alla memoria o si richiedono particolari tempi di calcolo [24].
Alcuni dei principali attacchi side-channel sono:

- **attacco alla cache** (cache attack), basato sul monitorare gli accessi alla memoria cache in un sistema che utilizza risorse condivise. Un caso esemplare è quello delle S-box implementate come look-up table, da evitare in quanto la cache permetterebbe di rivelare a quali parti della tabella vengono effettuati gli accessi;
- **ricerca di dati fantasma** (data remanence), ovvero studiare eventuali dati rimasti anche in seguito alla cancellazione;
- **attacco a cronometro** (timing attack), nel quale l'attaccante ha accesso ai tempi di esecuzione delle operazioni interne al cifrario;
- **attacco di monitoraggio dei consumi** (power-monitoring attack), che, invece di guardare i tempi, considera il consumo elettrico della macchina;
- **attacco elettromagnetico** (electromagnetic attack), sempre basato sulle informazioni energetiche, ma specifico sull'energia elettromagnetica irradiata;
- **attacco ottico** (optical attack), che sfrutta una videocamera per catturare immagini ad alta risoluzione, in modo da identificare variazioni dello stato dei componenti della macchina;
- **crittoanalisi acustica** (acoustic cryptanalysis), simile al precedente, ma incentrato sulle variazioni acustiche;
- **analisi differenziale dei malfunzionamenti** (differential fault analysis), nella quale si inducono problemi alle componenti fisiche per studiare le variazioni provocate al sistema;
- **attacco su malfunzionamenti indotti dal software** (software-initiated fault attacks), analogo alla pratica precedente, ma sfrutta errori provocati dal software.

3.4 Altri attacchi

Esistono ulteriori attacchi che risultano efficienti in situazioni particolari dipendenti dalle scelte effettuate in fase di costruzione del cifrario.
Ad esempio, il numero di round in una SPN deve essere tale da impedire un tipo di CPA noto come **Square attack** [25] o **crittoanalisi integrale** (integral cryptanalysis) [26], che sfrutta la struttura di questo tipo di cifrari e alcune proprietà matematiche dei campi finiti per ricavare le chiavi di round.
Esistono, inoltre, delle strategie più complesse della crittoanalisi differenziale, che sfruttano la combinazione di più di un singolo differenziale per effettuare un attacco al cifrario a blocchi e che prendono il nome di **attacchi a boomerang** [27, 28, 29].
Un altro caso è quello dei **related-key attacks**, applicabili a cifrari le cui chiavi di round sono legate da un qualche vincolo matematico, generate quindi da un key schedule inadeguato. Esempi comuni di questa vulnerabilità sono la presenza di ripetizioni costanti nei bit delle chiavi di round [30], oppure l'utilizzo di chiavi di round che variano di quantità definite [31].

Minaccia quantistica

Come la maggior parte della crittografia simmetrica, i cifrari a blocchi risultano suscettibili agli attacchi perpetrati da un computer quantistico tramite l'**algoritmo di Grover** [32], che tuttavia garantisce al più un aumento quadratico della velocità degli attacchi di forza bruta. Quindi, un raddoppio delle dimensioni della chiave è sufficiente a garantire lo stesso livello di sicurezza degli standard attuali.

4 Advanced Encryption Standard (AES)

Nel 1997 il NIST lanciò una competizione pubblica [33] per scegliere il successore di DES (Data Encryption Standard) [7], un cifrario a blocchi di tipo Feistel che utilizzava chiavi da soli 64 bit. La selezione del nuovo cifrario si concluse nel 2001 e fu vinta dall'algoritmo **Rijndael**, che da allora assunse il nome di **AES** (Advanced Encryption Standard) [14]. In seguito alla standardizzazione, AES divenne la scelta più utilizzata a livello mondiale tra i cifrari a blocchi.
AES è una SPN che lavora su blocchi lunghi 128 bit e presenta tre varianti a seconda della lunghezza k della chiave e del numero r di round: 128 bit e 10 round, 192 bit e 12 round, 256 bit e 14 round. Il blocco B di 128 bit (16 byte) viene organizzato in una matrice nel seguente modo: considerando la suddivisione in byte $B = b_1 \| b_2 \| \dots \| b_{16}$, lo stato viene memorizzato nella forma

$$\begin{pmatrix} b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \\ b_4 & b_8 & b_{12} & b_{16} \end{pmatrix}.$$

A ogni round viene calcolato lo XOR tra lo stato e la chiave di round, che quindi deve avere una lunghezza di 128 bit. Il key schedule di AES utilizza delle costanti di round, delle rotazioni circolari e la S-box S del cifrario per produrre le $r + 1$ chiavi di round necessarie.

L'algoritmo di cifratura di AES è schematizzato in Figura 3. Come descritto in generale per le SPN, l'algoritmo inizia con un round di whitening che prevede lo XOR tra il blocco in chiaro e la prima chiave di round.
Si prosegue quindi con i round standard, il primo dei quali è dettagliato a destra nella figura. La prima operazione prevede l'utilizzo di una singola S-box S con input e output di 8 bit. Questa utilizzata nella funzione di sostituzione $\mathrm{SubBytes}$, ottenuta applicando S byte per byte. Il mixing layer è composto da due fasi chiamate $\mathrm{ShiftRows}$ e $\mathrm{MixColumns}$: nella prima le righe della matrice che rappresenta lo stato vengono ruotate verso sinistra di un numero di posizioni che aumenta riga per riga ($\ll 0, \ll 1, \ll 2, \ll 3$), mentre nella seconda ogni colonna viene moltiplicata per una matrice fissata (funzione Mix in figura), così da ottenere un completo aggiornamento della matrice. Infine, si calcola lo XOR tra lo stato risultante e la chiave di round.
L'ultimo round si differenzia dai precedenti in quanto si applica un mixing layer parziale che non prevede l'utilizzo di $\mathrm{MixColumns}$.
La S-box di AES è una nota funzione booleana che compone una funzione affine con la funzione che calcola l'inverso nel campo finito di dimensione 2^8 definito dal polinomio irriducibile e non primitivo $x^8 + x^4 + x^3 + x + 1$. Questa funzione presenta una nonlinearità pari a 112, per cui si vanificano le speranze di successo degli attacchi lineari.

Inoltre, è 4-uniforme differenzialmente e quindi anche molto resistente agli attacchi differenziali.
Queste proprietà si combinano bene con l'utilizzo del mixing layer, che garantisce una rapida diffusione dei bit all'interno dello stato, incrementando ulteriormente la resistenza ad attacchi lineari e differenziali.

Il miglior attacco a AES è il **biclique attack** [34] che ha una complessità di pochi bit inferiore rispetto a quella del generico attacco a forza bruta. Quindi, questo attacco è puramente teorico e non risulta intaccare la sicurezza generale del cifrario, che è ancora considerato completamente sicuro.

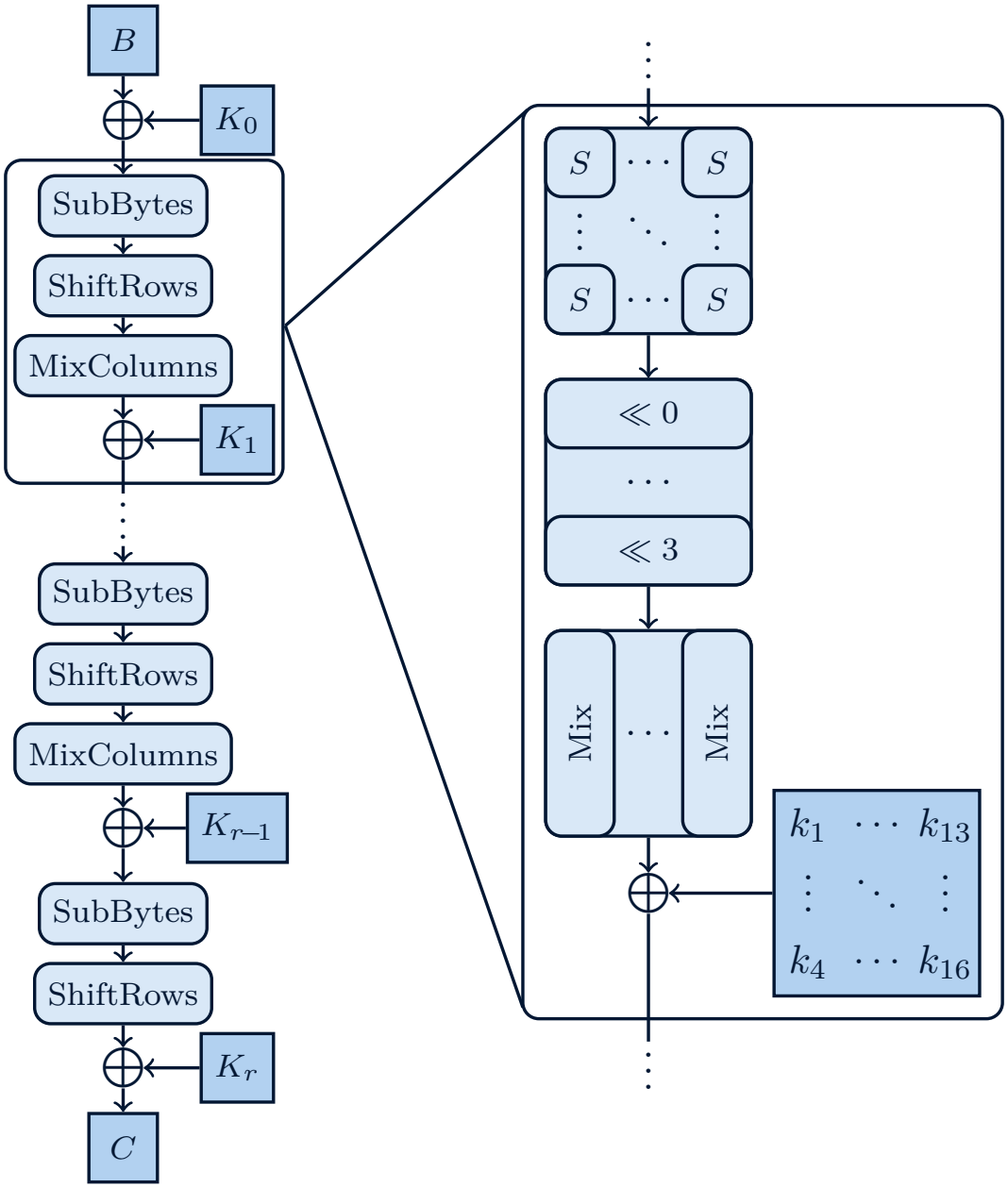

Figura 3 - Cifratura di AES, con dettaglio del primo round a destra

5 Modalità di funzionamento

In questo capitolo, si presentano le **modalità di funzionamento** per i cifrari a blocchi [35, 36], ossia i procedimenti utilizzati per cifrare un messaggio la cui lunghezza è maggiore di quella prevista dallo stato del cifrario a blocchi considerato. Le prime quattro modalità presentate furono sviluppate per DES, ma valgono per tutti i cifrari a blocchi con spazio dei messaggi che coincide con quello dei testi cifrati. Le due successive permettono di ottenere una cifratura autenticata. Infine, viene presentata una modalità adatta alla cifratura su dispositivi di memorizzazione.

Nel seguito, si indicheranno con $B_1, B_2, \ldots, B_n$ i blocchi del testo in chiaro M, cioè $M = B_1 \| B_2 \| \ldots \| B_n$, con $C_1, C_2, \ldots, C_n$ i blocchi del testo cifrato, con K la chiave, con E l'algoritmo di cifratura e con D l'algoritmo di decifratura. In tutte le modalità raccomandate si utilizza una stringa pseudocasuale di bit chiamata **vettore di inizializzazione**, che verrà indicata con IV. L'IV non deve necessariamente essere nascosto e può essere inviato in chiaro insieme al testo cifrato, eventualmente utilizzando uno schema crittografico per garantirne l'integrità.

Chiaramente può accadere che il messaggio M da cifrare abbia una lunghezza che non sia un multiplo di quella del blocco elaborato dal cifrario. In questo caso, vengono attuate delle strategie di padding, che vengono presentate nel dettaglio nel capitolo 6.

Si noti che, sebbene per completezza si presentano tutte le modalità di funzionamento più comuni, alcune di esse sono raccomandate solamente in situazioni particolari.

5.1 Electronic codebook (ECB)

La modalità **ECB** [35], come indica il nome, consiste semplicemente nel cifrare ogni blocco applicando direttamente la cifratura sempre con la stessa chiave. Più precisamente, i blocchi del messaggio cifrato saranno

$$\begin{aligned} C_1 &= E(B_1), \\ C_2 &= E(B_2), \\ &\ldots \\ C_n &= E(B_n). \end{aligned}$$

Il vettore di inizializzazione deve sempre essere utilizzato **una sola volta** per ogni chiave e deve essere generato **casualmente**, o comunque seguendo le specifiche della modalità di funzionamento adottata.

L'utilizzo della modalità ECB è **fortemente sconsigliato**.

Il messaggio in chiaro si ottiene applicando la funzione di decifratura ai singoli blocchi cifrati.
In questa modalità i blocchi vengono processati indipendentemente ed è quindi possibile effettuare le cifrature in parallelo, ottenendo un processo di cifratura molto efficiente.
Tuttavia, l'indipendenza tra i blocchi può anche essere sfruttata da un attore malevolo, in quanto la manomissione o sostituzione di un singolo blocco non influenza la corretta decifratura degli altri.
Un'altra debolezza critica di ECB risiede nella cifratura di blocchi uguali, in quanto i corrispondenti testi cifrati sarebbero identici. Questa problematica può rivelarsi cruciale soprattutto quando i messaggi da cifrare sono lunghi e strutturati, in quanto un'analisi statistica potrebbe sfruttare blocchi ricorrenti per cercare di decifrare il messaggio.

5.2 Cipher block chaining (CBC)

Nella modalità CBC [35, 36], al primo passo, si cifra lo XOR tra il primo blocco B_1 e un IV mentre, a ogni passo successivo i, quello tra il precedente blocco cifrato C_{i-1} e il blocco di testo in chiaro B_i, per cui

$$\begin{aligned} C_1 &= E(B_1 \oplus IV), \\ C_2 &= E(B_2 \oplus C_1), \\ &\cdots \\ C_n &= E(B_n \oplus C_{n-1}). \end{aligned}$$

La decifratura si svolge eseguendo le operazioni a ritroso, ovvero nel modo seguente:

$$\begin{aligned} B_1 &= D(C_1) \oplus IV, \\ B_2 &= D(C_2) \oplus C_1, \\ &\cdots \\ B_n &= D(C_n) \oplus C_{n-1}. \end{aligned}$$

I due algoritmi sono descritti graficamente in Figura 4. Si noti come, in questa modalità, la creazione di ogni blocco di testo cifrato dipende dal precedente blocco cifrato, per cui la cifratura non è parallelizzabile. Inoltre, questo significa che un errore nella cifratura di un blocco compromette tutti i blocchi successivi, il che rende CBC adatto alla costruzione di MAC, come descritto nel documento dedicato [37].

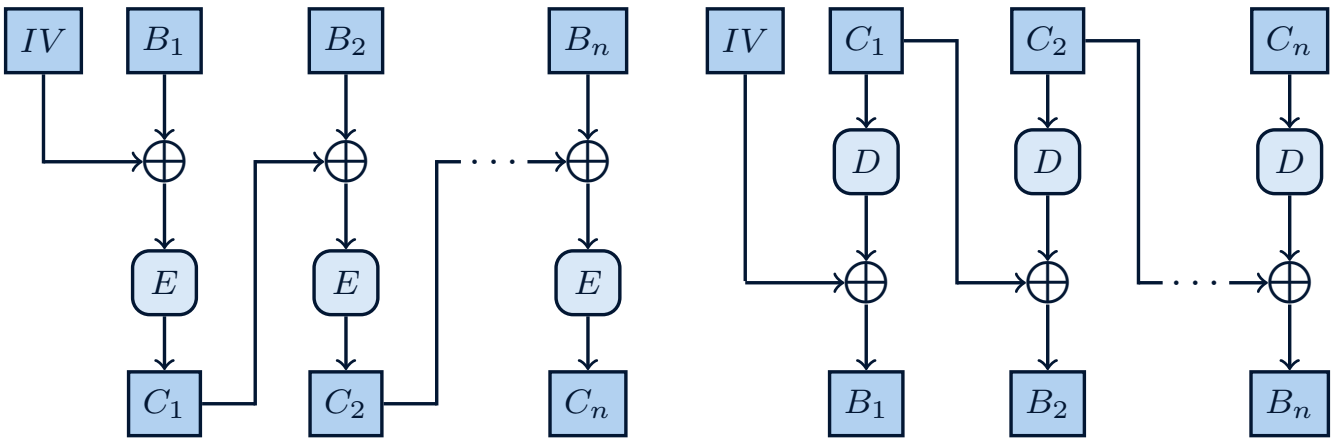

Figura 4 - Modalità CBC, cifratura a sinistra e decifratura a destra

La fase di decifratura è invece parallelizzabile e un errore in un singolo blocco cifrato o la sua compromissione malevola non impedisce la decifratura di tutti gli altri blocchi. In particolare, se il blocco C_i risulta modificato, mentre gli altri blocchi vengono ricevuti correttamente, allora B_i e B_{i+1} risultano compromessi ma i blocchi precedenti e successivi vengono decifrati correttamente.
Per quanto riguarda l'IV, esso deve essere generato con una funzione pseudocasuale e inviato in chiaro insieme al testo cifrato. È importante che questa stringa di bit non venga mai riutilizzata due volte con la stessa chiave, per evitare che da due messaggi che condividono il primo blocco si ottengano due testi cifrati anch'essi identici nel primo blocco. Inoltre, è necessario assicurare l'integrità dell'IV, in quanto un malintenzionato potrebbe tentare di modificarlo per compromettere a suo piacimento la decifratura del primo blocco del testo in chiaro.

5.2.1 Cipher block chaining ciphertext stealing (CBC-CS)

La modalità CBC può essere integrata incorporando il cosiddetto **ciphertext stealing**, ottenendo la modalità chiamata **CBC-CS** [35, 38]. La differenza sostanziale risiede nell'elaborazione degli ultimi due blocchi, come si può osservare in Figura 5. Per prima cosa, al messaggio M vengono aggiunti d bit uguali a 0 (fase di padding) per renderlo divisibile in n blocchi della lunghezza desiderata. Il padding non si applica quando la lunghezza di M si divide perfettamente, cioè quando l'ultimo blocco risulta completo.
Quindi, il cifrario viene eseguito in modalità CBC, ottenendo $C = C_1 \,\|\, C_2 \,\|\, \ldots \,\|\, C_{n-2} \,\|\, C_{n-1} \,\|\, C_n$. In una fase di finalizzazione, vengono scambiati gli ultimi due blocchi del cifrato e poi scartati gli ultimi d bit, ottenendo quindi il messaggio cifrato

$$C = C_1 \,\|\, C_2 \,\|\, \ldots \,\|\, C_{n-2} \,\|\, C_n \,\|\, C^*_{n-1},$$

in cui C^*_{n-1} è ottenuto da C_{n-1} scartando gli ultimi d bit. Esistono altre versioni di questa modalità che non prevedono lo scambio degli ultimi due blocchi di testo cifrato, o di effettuarlo solamente nel caso in cui il padding sia stato applicato.
La peculiarità di questa modalità è che la lunghezza del testo in chiaro M è la stessa del testo cifrato C, e quindi non viene fornita alcuna informazione sull'eventuale presenza di padding a un attaccante in possesso di testo in chiaro e corrispettivo testo cifrato (KPA o CPA).
La decifratura in modalità CBC-CS, rappresentata in Figura 6, lavora esattamente come in modalità CBC sui primi $n-2$ blocchi cifrati. Quando si arriva al penultimo blocco, è necessaria una fase preliminare per recuperare i bit scartati in fase di cifratura: si applica la funzione di decifratura a C_n e gli ultimi d bit della stringa ottenuta vengono concatenati al blocco C^*_{n-1}. Applicando decifratura e XOR con il cifrato C_{n-2} al blocco completo così ottenuto si ricostruisce B_{n-1}. Infine, si può concludere la decifratura di C_n in modalità CBC per ottenere l'ultimo blocco del messaggio in chiaro, inclusi i d bit nulli di padding che possono essere scartati.

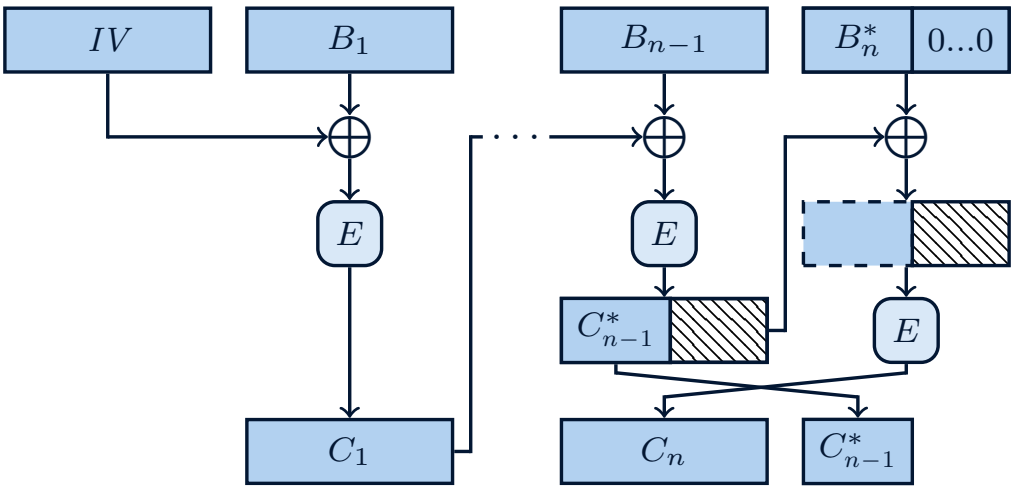

Figura 5 - Modalità CBC-CS, cifratura

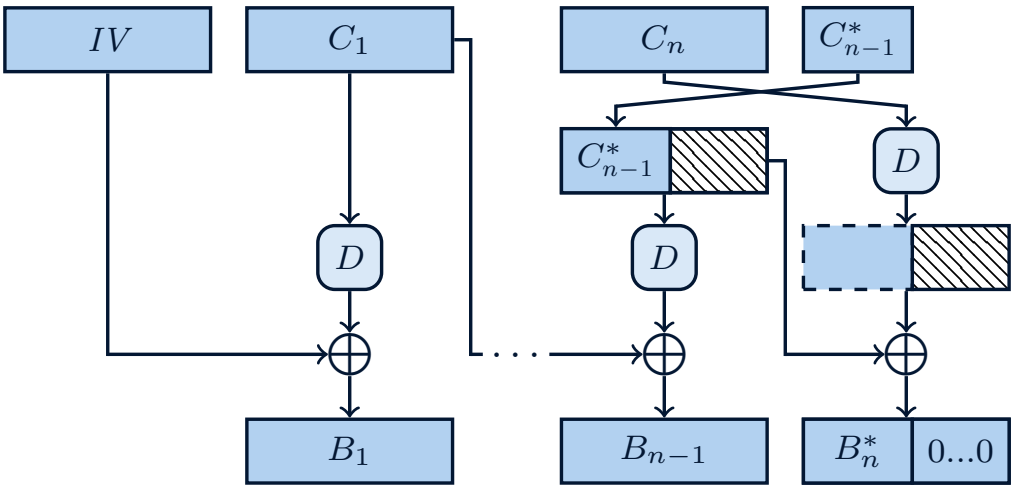

Figura 6 - Modalità CBC-CS, decifratura

5.3 Cipher feedback (CFB)

La modalità **CFB** [35, 36] permette di utilizzare un cifrario a blocchi come un cifrario a flusso, per cui si genera una stringa di bit di lunghezza arbitraria, chiamata **keystream**, e si ottiene il testo cifrato come XOR tra quest'ultima e il testo in chiaro. Per maggiori dettagli sui cifrari a flusso si rimanda al documento dedicato [2].
Con la modalità CFB, si ha

$$\begin{aligned} Z_1 &= E(IV), \quad C_1 = B_1 \oplus Z_1, \\ Z_2 &= E(C_1), \quad C_2 = B_2 \oplus Z_2, \\ &\cdots \\ Z_n &= E(C_{n-1}), \quad C_n = B_n \oplus Z_n, \end{aligned}$$

per cui il keystream si ottiene concatenando i blocchi $Z_1, Z_2, \ldots, Z_n$, i quali sono ottenuti cifrando un IV al primo passo e, nei passi successivi, il blocco di testo cifrato al passo precedente. Il testo cifrato è invece ottenuto con la concatenazione dei blocchi $C_1, C_2, \ldots, C_n$.
In fase di decifratura è sufficiente ottenere nuovamente lo stesso keystream e calcolare lo XOR con il testo cifrato ricevuto per ricostruire il messaggio in chiaro.
Si noti come l'algoritmo di cifratura, rappresentato a sinistra in Figura 7, non sia parallelizzabile mentre in quello di decifratura, raffigurato a destra, i blocchi del keystream si possono ottenere in parallelo cifrando i blocchi $C_1, C_2, \ldots, C_n$ ricevuti.

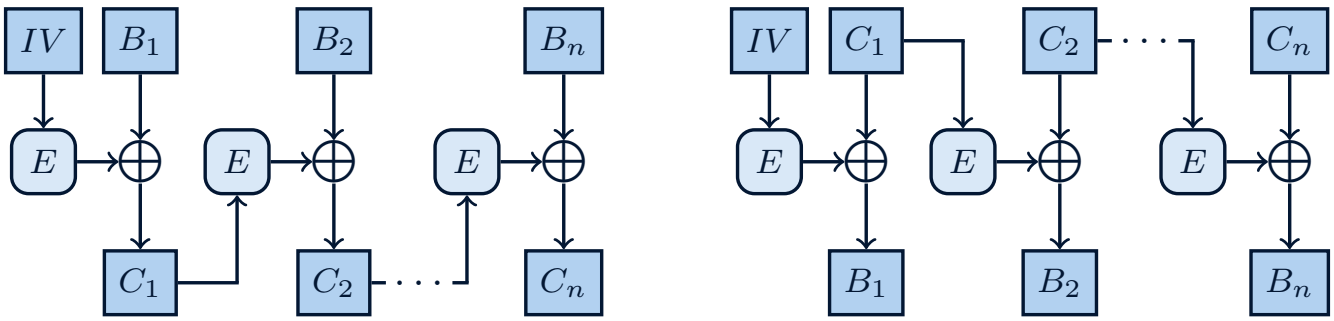

Figura 7 - Modalità CFB, cifratura a sinistra e decifratura a destra

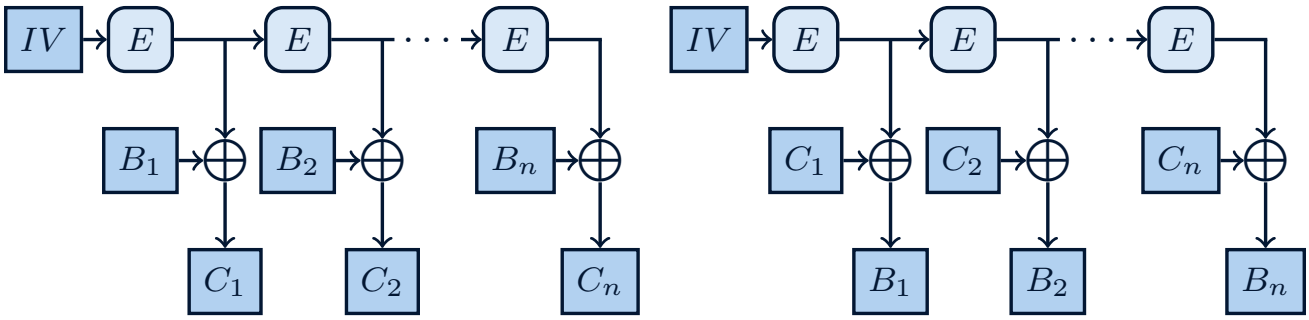

Figura 8 - Modalità OFB, cifratura a sinistra e decifratura a destra

5.4 Output feedback (OFB)

Similmente a CFB, anche la modalità **OFB** [35, 36] funziona come un cifrario a flusso. La differenza risiede nella generazione dei blocchi $Z_1, Z_2, \ldots, Z_n$ del keystream, i quali vengono ottenuti cifrando ripetutamente un IV, in formule

$$\begin{aligned} Z_1 &= E(IV), \quad C_1 = B_1 \oplus Z_1, \\ Z_2 &= E(Z_1), \quad C_2 = B_2 \oplus Z_2, \\ &\ldots \\ Z_n &= E(Z_{n-1}), \quad C_n = B_n \oplus Z_n. \end{aligned}$$

La cifratura e il suo inverso sono rappresentati in Figura 8. La decifratura avviene, come per CFB, generando il keystream e calcolando lo XOR blocco a blocco con il testo in chiaro. In questo caso, non è possibile parallelizzare la generazione del keystream, ma è possibile precalcolarlo, essendo esso indipendente dal testo in chiaro o dal cifrato. Una caratteristica peculiare di questa modalità è che un singolo errore di trasmissione di un bit del messaggio cifrato, non impedisce la corretta trasmissione del resto del messaggio.
È importante che l'IV sia diverso per ogni keystream da generare. Infatti, se così non fosse, si presenterebbe il problema in cui blocchi di testo in chiaro identici vengono cifrati nello stesso modo. Inoltre, se due messaggi M e M' vengono cifrati con la modalità OFB con stesso IV e stesso keystream, cioè $C = M \oplus Z$ e $C' = M' \oplus Z$, vale che $C \oplus C' = M \oplus M'$. L'attaccante sarebbe dunque in grado di ottenere lo XOR tra i due testi in chiaro, uno dei quali potrebbe essere ricavato tramite analisi statistica.

5.5 Counter (CTR)

La modalità **CTR** [35, 36] lavora in modo simile a OFB. In questo caso, la generazione dei blocchi $Z_1, Z_2, \ldots, Z_n$ del keystream avviene a partire da un **counter** (contatore), che viene cifrato e poi aggiornato per essere cifrato nuovamente. Viene quindi calcolato lo XOR tra i blocchi di testo in chiaro e i corrispondenti blocchi del keystream per ottenere il cifrato, cioè

$$Z_i = E(\text{counter}_i), \quad C_i = B_i \oplus Z_i.$$

Il primo counter corrisponde con un IV casuale e gli aggiornamenti dipendono dall'indice del blocco in fase di cifratura secondo una delle due seguenti modalità:

1. si fissa un numero intero m e a ogni iterazione si somma 1 agli ultimi m bit dell'IV, effettuando eventualmente un riporto. Per esempio, se $m = 4$ e $IV = b_1 \ldots b_\ell 1110$, allora

$$\begin{aligned} \text{counter}_1 &= b_1 \ldots b_\ell 1110, \\ \text{counter}_2 &= b_1 \ldots b_\ell 1111, \\ \text{counter}_3 &= b_1 \ldots b_\ell 0000, \end{aligned}$$

   così da ottenere 2^m counter diversi. Di conseguenza, al fine di evitare che due blocchi vengano cifrati con lo stesso counter, è necessario che questi siano al più 2^m. Inoltre, è richiesto che l'IV sia totalmente casuale per resistere ad attacchi nello scenario CPA;
2. si fissa un numero intero m che rappresenta il numero di bit da concatenare all'IV, i quali partono da 0 e raggiungono man mano 2^m. Per esempio, se $m = 4$,

allora i counter si ottengono come

$$\begin{aligned}\text{counter}_1 &= IV \,\|\, 0000,\\ \text{counter}_2 &= IV \,\|\, 0001,\\ \text{counter}_3 &= IV \,\|\, 0010,\end{aligned}$$

Anche in questo caso il numero di blocchi da cifrare dovrà essere inferiore a 2^m.

In generale, si raccomanda 128 come valore minimo per m.
La decifratura in modalità CTR avviene in modalità analoga alla cifratura: si genera il keystream a partire dai counter e viene effettuato lo XOR con il cifrato per recuperare il testo in chiaro. I due algoritmi sono rappresentati in Figura 9.
La modalità CTR è totalmente parallelizzabile sia in fase di cifratura che in fase di decifratura, e questo la rende ottima per l'implementazione su macchine che utilizzano più processori in parallelo.

5.6 Cifratura su dispositivi di memorizzazione (XTS)

Per alcuni casi specifici sono state ideate delle modalità di funzionamento dedicate.
Un caso esemplare è quello della cifratura su dispositivi di memorizzazione, come ad esempio hard disk o chiavette USB, i quali sono generalmente utilizzati per immagazzinare dati a riposo, che devono essere protetti correttamente dagli eventuali attacchi di attori malevoli.
In questo contesto, la modalità di funzionamento **XTS** per AES è stata ideata come alternativa più sicura alle modalità ECB e CBC, ed è diventata standard IEEE nel 2007 [39] e NIST nel 2010 [40].

La modalità XTS soddisfa le caratteristiche della cosiddetta **cifratura trasparente** che, nell'ambito dei dispositivi di memorizzazione, prevede i due requisiti seguenti:

- cifratura e decifratura devono preservare la dimensione dei dati, e quindi devono essere processi deterministici che non possono coinvolgere la computazione di un tag di autenticazione;
- la cifratura deve poter agire su ogni singola unità di dati indipendentemente e in ordine arbitrario così che, per accedere a un determinato settore della memoria, non sia necessario decifrarne numerosi altri.

Altre modalità di funzionamento, come CBC o CTR, falliscono nel soddisfare queste due proprietà in maniera sicura.
La modalità XTS si basa su una struttura **XEX** (XOR Encrypt XOR) e integra la modalità **tweaked-codebook**, dovuta a Phillip Rogaway [41], con la tecnica di ciphertext stealing già presentata per CBC.
Per prima cosa, XTS richiede una chiave K lunga il doppio di quelle da utilizzare per il cifrario E, in quanto questa viene suddivisa in due sottochiavi $K = K_1 \,\|\, K_2$. Nella struttura XEX, a input e output del cifrario a blocchi viene applicato lo XOR con una stessa stringa di bit chiamata **tweak**.
Questa modalità, a differenza di quelle precedenti, non può utilizzare un IV casuale, ma utilizza un IV generato deterministicamente a partire da alcune informazioni presenti sul disco (come la posizione di un blocco all'interno di un settore di memoria). In ogni caso, questo valore deve essere generato in modo che sia unico per ogni chiave, come raccomandato per le altre modalità di funzionamento.

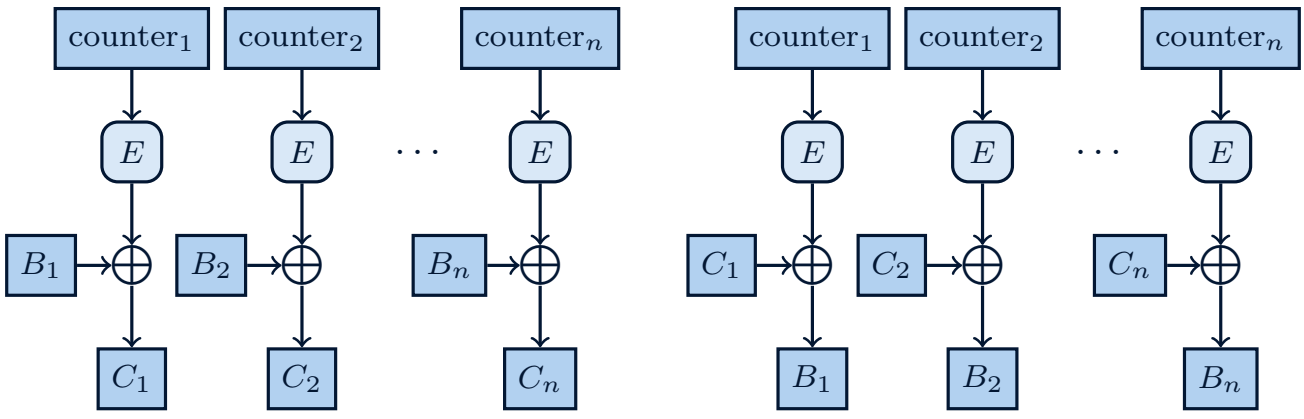

Figura 9 - Modalità CTR, cifratura a sinistra e decifratura a destra

Come rappresentato in Figura 10, nella cifratura in modalità XTS il tweak viene ottenuto al primo passo come cifratura di un IV con la chiave K_2 e ai passi successivi come il tweak del passo precedente moltiplicato in un campo finito ($\otimes$) per l'elemento primitivo α. In questa operazione, la modalità XTS identifica le stringhe di bit come elementi di un campo finito, analogamente a quanto accade nelle definizioni delle funzioni di AES. Nello specifico, il campo finito utilizzato nella modalità XTS ha 2^{128} elementi e viene generato dal polinomio $x^{128} + x^7 + x^2 + x + 1$, la cui radice α è l'elemento primitivo (i.e., ogni elemento del campo può essere ottenuto come potenza di α).
La cifratura dello XOR tra il blocco e il tweak viene effettuata utilizzando la chiave K_1. Agli ultimi due blocchi, si applica il ciphertext stealing in modo simile a quanto descritto nella sezione 5.2.1: supponendo che, in seguito alla divisione in blocchi, resti un'ultima parte di testo in chiaro B_n^* a cui mancano d bit per raggiungere la lunghezza di un blocco, si effettua un padding concatenando a B_n^* gli ultimi d bit di C_{n-1}, cioè il blocco cifrato al passo precedente.
Una volta ottenuti tutti i blocchi cifrati, avviene uno scambio di posizione tra l'ultimo blocco cifrato C_n e C_{n-1}^*, cioè la stringa di bit ottenuta al penultimo passo esclusi i d bit utilizzati per il padding. In questo modo, il testo cifrato ha la stessa lunghezza del messaggio in chiaro.
Nell'invertire il processo per la decifratura è importante fare attenzione alle operazioni da eseguire sugli ultimi due blocchi del cifrato. Come descritto in Figura 11, per i primi blocchi è sufficiente calcolare il tweak e ricostruire il blocco in chiaro tramite la struttura XEX utilizzando la decifratura D_{K_1} al posto di E_{K_1}.
Il penultimo blocco C_n deve essere decifrato utilizzando l'ultimo tweak e il risultato corrisponde alla concatenazione tra l'ultima parte B_n^* del testo in chiaro e il padding da utilizzare con l'ultima parte di testo cifrato C_{n-1}^*.
La concatenazione di questi due blocchi viene infine decifrata utilizzando il penultimo tweak per ottenere B_{n-1}, che si concatena agli altri blocchi prima di B_n^*.

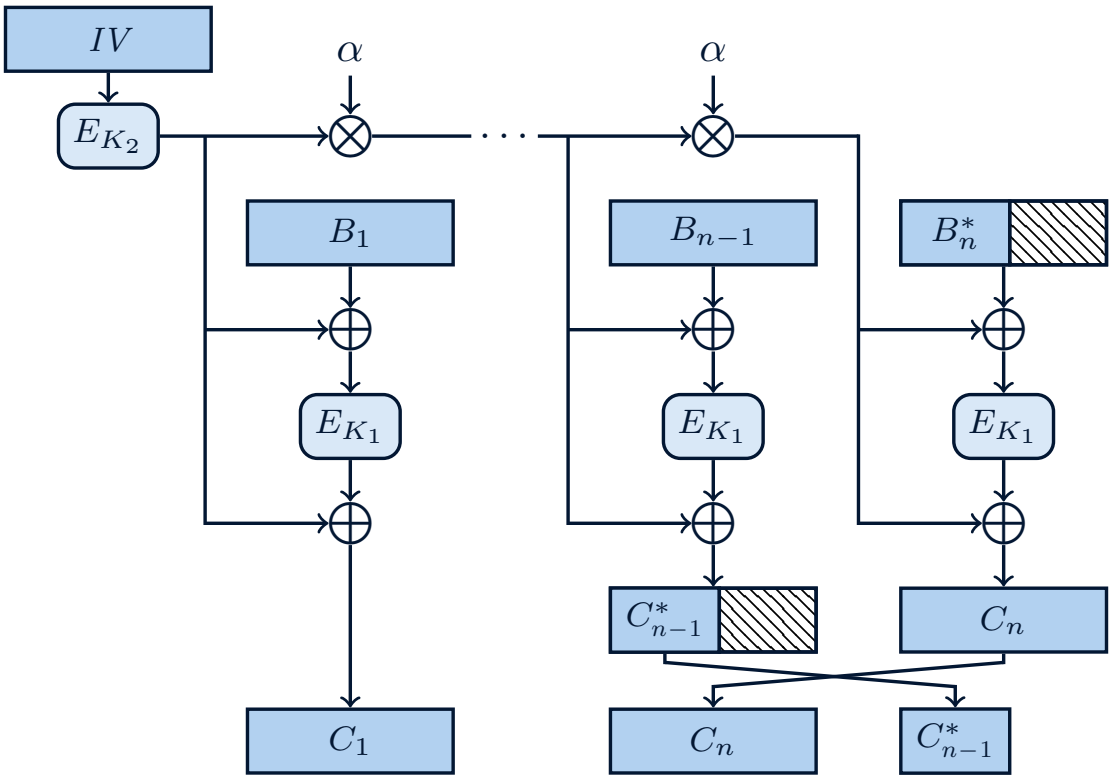

Figura 10 - Modalità XTS, cifratura

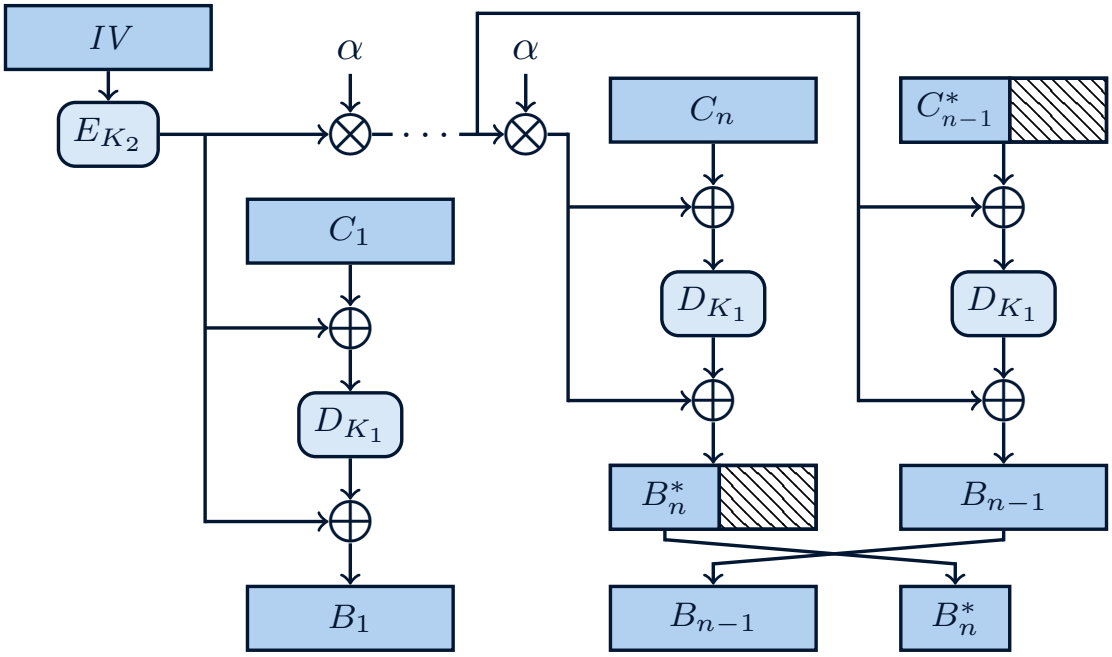

Figura 11 - Modalità XTS, decifratura

5.7 Modalità per la cifratura autenticata

Tutte le modalità presentate finora risultano essere **malleabili**. Questo significa che è possibile modificare un messaggio cifrato in modo che la sua decifratura produca un nuovo messaggio in chiaro, il quale presenta una relazione nota con il messaggio in chiaro di partenza.
Ad esempio, nella modalità CTR, se si considera il blocco B e il keystream Z, si ottiene il cifrato $C = B \oplus Z$. Se si fornisce il blocco C' ottenuto cambiando il primo bit di C, allora in fase di decifratura si otterrà il blocco M' che possiede il primo bit cambiato rispetto a quello di partenza.
Chiaramente, questa proprietà potrebbe essere sfruttata da malintenzionati per modificare il senso di un dato messaggio. Per questo motivo, si raccomanda sempre di utilizzare una soluzione per la **cifratura autenticata**, la quale prevede l'invio di un tag per garantire che il messaggio non sia stato alterato in alcun modo.
Questa proprietà viene garantita dal paradigma **encrypt-then-MAC** o da una delle modalità di funzionamento brevemente descritte in seguito. Per una trattazione più approfondita su questa tematica, si rimanda al documento dedicato [16].

5.7.1 Counter con cipher block chaining MAC (CCM)

La modalità **CCM** [42, 43] combina la modalità CBC con la CTR ed è un esempio di cifratura autenticata con dati associati. In particolare, la modalità CCM prevede l'utilizzo della modalità CBC per generare un tag, il quale viene poi utilizzato per cifrare il messaggio stesso utilizzando la modalità CTR.

5.7.2 Galois counter mode (GCM)

La modalità **GCM** [42, 44] è la più recente tra quelle presentate e, come CCM, fornisce una cifratura autenticata con dati associati. Questa modalità utilizza un contatore come nella modalità CTR, ma alcune delle operazioni vengono effettuate in un campo finito (o di Galois, da cui il nome). Può essere utilizzata anche per produrre solamente un MAC che assume il nome di GMAC, descritto nel documento dedicato [37].

6 Padding

Per utilizzare un cifrario a blocchi è necessario che il testo da cifrare abbia una lunghezza tale da poter essere suddiviso esattamente in blocchi della lunghezza prefissata. Quando questa richiesta non viene soddisfatta, è possibile concatenare dei bit al messaggio così da ottenere un messaggio in chiaro di lunghezza divisibile per quella dei blocchi. Questo processo viene detto **padding**. In tal caso, la lunghezza del testo cifrato risulta maggiore o uguale a quella del messaggio in chiaro e, per poter decifrare correttamente, il destinatario deve essere a conoscenza del metodo di padding utilizzato. Inoltre, per evitare ambiguità nella decifratura, è pratica comune applicare il padding anche quando la lunghezza del messaggio è già un multiplo della lunghezza dei blocchi, caso in cui si procede ad aggiungere un intero blocco di padding. Esistono anche alcuni metodi con i quali è possibile riportare il messaggio cifrato alla lunghezza del testo in chiaro senza perdere informazioni che ne impediscano la decifratura, come il ciphertext stealing descritto nella sezione 5.2.1.
Le strategie di padding raccomandate sono le seguenti:

- il **bit padding** [36, 45] consiste nell'ottenere una lunghezza multipla di quella del blocco concatenando al messaggio in chiaro una stringa della forma $10\ldots0$ o eventualmente il singolo bit 1. Se la lunghezza di partenza risulta già corretta, viene aggiunto un intero blocco di solo padding;
- il padding utilizzato nel protocollo **ESP** (Encapsulating Security Payload) [46], che consiste nell'aggiungere in maniera incrementale i byte mancanti, fino a un massimo di 255, concatenando una stringa del tipo "01 02 03 ...". Se la lunghezza di partenza risulta già corretta, il messaggio resta invariato;
- il metodo **PKCS #7** [47] aggiunge un padding di d byte costituito dal byte d ripetuto d volte, per cui il padding (scritto in esadecimale) sarà "01" se $d = 1$, "02 02" se $d = 2$, "03 03 03" se $d = 3$, e così via. Nel caso in cui la lunghezza del messaggio di partenza sia già multipla della lunghezza di un blocco, si aggiunge un intero blocco di padding della lunghezza del blocco. Dal momento che il valore massimo per d è 255 (o "FF" in esadecimale), questa tecnica di padding può essere utilizzata solo se la lunghezza del blocco del cifrario è inferiore a 256 byte.

6.1 Padding oracle attack

È importante evidenziare che è possibile sfruttare il padding per effettuare un attacco a un cifrario a blocchi utilizzato in modalità CBC attraverso il cosiddetto **padding oracle attack** [48]. L'attacco prevede l'utilizzo di un oracolo che, ricevuto un testo cifrato, è in grado di stabilire se al rispettivo testo in chiaro sia stato applicato un padding valido o meno. Sono un esempio di oracolo i servizi che rispondono alle richieste con messaggi di errore dettagliati, ma esistono anche alcuni attacchi side-channel in grado di fornire questa informazione.

Questo attacco è riservato alla modalità CBC per via della sua struttura in fase di decifratura (si veda Figura 4): se si riesce a ricavare un blocco decifrato allora è sufficiente calcolare lo XOR tra questo risultato e il blocco cifrato precedente per ottenere un blocco del testo in chiaro. L'attaccante sfrutta il fatto che, modificando l'IV, è possibile manipolare a proprio piacimento i bit del testo in chiaro, in quanto $B_1 = D(C_1) \oplus IV$. L'attacco funziona con ogni metodo di padding, adattando i passaggi alle diverse modalità.

Una soluzione per far fronte al padding oracle attack è autenticare il messaggio tramite un MAC che deve essere verificato prima della decifratura (**encrypt-then-MAC**). In questo modo, chiunque non sia in grado di autenticare correttamente il messaggio non riceverà alcuna informazione riguardo la validità o meno del testo in chiaro.

7 Conclusioni

7.1 Raccomandazioni sui cifrari a blocchi

Per quanto detto nei capitoli precedenti, l'unico algoritmo di cifratura a blocchi raccomandato è AES con i parametri riportati nella Tabella 3. Chiaramente, a chiavi più lunghe corrisponde una maggiore sicurezza, al costo di un leggero aumento dei tempi di calcolo richiesti. Questo algoritmo può essere utilizzato direttamente per cifratura e decifratura, tramite le modalità di funzionamento specificate nel paragrafo successivo. Ovviamente, la chiave segreta utilizzata deve prima essere condivisa in maniera sicura attraverso uno degli schemi per lo scambio o derivazione di chiave, come descritto nel documento dedicato*.

Algoritmo	Lunghezza della chiave
AES	128 bit
	192 bit
	256 bit

Tabella 3 - Cifrari a blocchi e parametri raccomandati

Si raccomanda sempre di utilizzare AES con lunghezza della chiave **maggiore possibile**, tenendo conto della **potenza computazionale** disponibile e delle esigenze implementative.

Minaccia quantistica

In casi che richiedono un elevato periodo di conservazione del dato cifrato, per evitare un attacco quantistico di tipo "harvest now, decrypt later", si raccomanda di **utilizzare una chiave da 192 o 256 bit**.

*In fase di pubblicazione.

7.2 Raccomandazioni sulle modalità di funzionamento

In Tabella 4 sono elencate le modalità di funzionamento raccomandate, con le seguenti **avvertenze**:

1. la scelta dell'IV deve sempre avvenire in maniera coerente con le specifiche tecniche delle singole modalità di funzionamento. In particolare, l'IV deve sempre essere utilizzato **una sola volta** per ogni chiave e deve essere generato **casualmente**, o comunque seguendo le specifiche della modalità di funzionamento adottata;
2. tutte le modalità presentate, a esclusione di XTS, risultano essere **malleabili**, cioè suscettibili a modifiche nel testo cifrato corrispondenti a variazioni prevedibili nel testo in chiaro. Per questo motivo, si raccomanda sempre l'utilizzo di una soluzione di cifratura autenticata;
3. le modalità CBC e CFB richiedono un **padding** adeguato per la formattazione dei dati in input. Le modalità di padding raccomandate sono quelle specificate nel capitolo 6. L'eventuale vulnerabilità al padding oracle attack può essere risolta adottando una soluzione di cifratura autenticata;
4. in caso di cifratura su **dispositivi di memorizzazione**, si raccomanda l'utilizzo della modalità XTS.

In generale, si raccomanda sempre di preferire soluzioni di **cifratura autenticata**, utilizzando le modalità in Tabella 5, oppure il paradigma encrypt-then-MAC con le modalità in Tabella 4, come indicato nel documento dedicato [16].

Modalità	Parallelizzabile		Avvertenze
	Cifratura	Decifratura	
CBC	✘	✔	1, 2, 3
CBC-CS	✘	✔	1, 2
CFB	✘	✔	1, 2, 3
OFB	✘	✘	1, 2
CTR	✔	✔	1, 2
XTS	✔	✔	1, 4

Tabella 4 - Modalità di funzionamento raccomandate con avvertenze

Modalità	Parallelizzabile	
	Cifratura	Decifratura
CCM	✔	✔
GCM	✔	✔

Tabella 5 - Modalità di funzionamento di cifratura autenticata raccomandate

Si raccomanda di adottare sempre una delle soluzioni che permettono la **cifratura autenticata** presenti in Tabella 5, oppure descritte nel documento dedicato [16]. L'uso di cifrari a blocchi per il solo scopo di confidenzialità è ammesso solo nei casi di assenza di trasmissione dei dati cifrati (come nel caso della cifratura dei dispositivi di memorizzazione).

Bibliografia

[1] ACN. *Introduzione alla Crittografia e alle Linee Guida*. Linee Guida Funzioni Crittografiche, 2026. URL: https://www.acn.gov.it/portale/crittografia.

[2] ACN. *Cifrari a Flusso*. Linee Guida Funzioni Crittografiche, 2026. URL: https://www.acn.gov.it/portale/crittografia.

[3] A. J. Menezes, P. C. V. Oorschot e S. A. Vanstone. *Handbook of applied cryptography (1st edition)*. CRC Press, 1997. DOI: 10.1201/9780429466335.

[4] D. R. Stinson e M. Paterson. *Cryptography: Theory and Practice (4th edition)*. Chapman e Hall/CRC, 2017. DOI: 10.1201/9781315282497.

[5] A. Bogdanov et al. «PRESENT: An Ultra-Lightweight Block Cipher». In: *Cryptographic Hardware and Embedded Systems (CHES)*. Lecture Notes in Computer Science. 2007, pp. 450–466. DOI: 10.1007/978-3-540-74735-2_31.

[6] C. Carlet. «Vectorial Boolean Functions for Cryptography». In: *Boolean Models and Methods in Math., Comput. Sci., and Eng.* Cambridge University Press, 2010, pp. 398–470. DOI: https://doi.org/10.1017/CBO9780511780448.012.

[7] NIST. *Data Encryption Standard (DES)*. FIPS 46-3. U.S. Department of Commerce, 1999. URL: https://csrc.nist.gov/pubs/fips/46-3/final.

[8] H. Feistel. «Cryptography and Computer Privacy». In: *Scientific American* 228.5 (1973), pp. 15–23. DOI: 10.1038/scientificamerican0573-15.

[9] ISO. *Information technology – Security techniques – Encryption Algortihms – Part 3: Block ciphers*. ISO/IEC 18033-3. 2010. URL: https://www.iso.org/standard/54531.html.

[10] K. Aoki et al. «Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis». In: *Selected Areas in Cryptography*. Lecture Notes in Computer Science. 2001, pp. 39–56. DOI: 10.1007/3-540-44983-3_4.

[11] B. Schneier et al. *Twofish: A 128-Bit Block Cipher*. NIST submission. 1998. URL: https://www.schneier.com/wp-content/uploads/2016/02/paper-twofish-paper.pdf.

[12] ETSI. *Universal Mobile Telecommunications System (UMTS); LTE; 3G Security; Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification, Version 17.0.0*. TS 135 202. 2022. URL: https://www.etsi.org/deliver/etsi_ts/135200_135299/135202/17.00.00_60/ts_135202v170000p.pdf.

[13] C. E. Shannon. «Communication theory of secrecy systems». In: *The Bell system technical journal* 28.4 (1949), pp. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x.

[14] NIST. *Advanced Encryption Standard (AES)*. FIPS 197. U.S. Department of Commerce, 2023. DOI: 10.6028/NIST.FIPS.197-upd1.

[15] E. Biham, R. Anderson e L. Knudsen. «Serpent: A New Block Cipher Proposal». In: *Fast Software Encryption (FSE)*. Lecture Notes in Computer Science. 1998, pp. 222–238. DOI: 10.1007/3-540-69710-1_15.

[16] ACN. *Cifratura Autenticata*. Linee Guida Funzioni Crittografiche, 2026. URL: https://www.acn.gov.it/portale/crittografia.

[17] NIST. *Lightweight Cryptography*. 2017. URL: https://csrc.nist.gov/projects/lightweight-cryptography.

[18] C. Dobraunig et al. «ASCON v1.2: Lightweight Authenticated Encryption and Hashing». In: *Journal of Cryptology* 34.33 (2021). DOI: 10.1007/s00145-021-09398-9.

[19] D. Hong et al. «HIGHT: A New Block Cipher Suitable for Low-Resource Device». In: *Cryptographic Hardware and Embedded Systems (CHES)*. Lecture Notes in Computer Science. 2006, pp. 46–59. DOI: 10.1007/11894063_4.

[20] M. Matsui e A. Yamagishi. «A New Method for Known Plaintext Attack of FEAL Cipher». In: *Advances in Cryptology - EUROCRYPT '92*. Lecture Notes in Computer Science. 1993, pp. 81–91. DOI: 10.1007/3-540-47555-9_7.

[21] M. Matsui. «Linear Cryptanalysis Method for DES Cipher». In: *Advances in Cryptology - EUROCRYPT '93*. Lecture Notes in Computer Science. 1994, pp. 386–397. DOI: 10.1007/3-540-48285-7_33.

[22] E. Biham e A. Shamir. «Differential Cryptanalysis of DES-like Cryptosystems». In: *Advances in Cryptology - CRYPTO '90*. Lecture Notes in Computer Science. 1991, pp. 2–21. DOI: 10.1007/3-540-38424-3_1.

[23] C. Carlet. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, 2021. DOI: 10.1017/9781108606806.

[24] M. Ouladj e S. Guilley. *Side-Channel Analysis of Embedded Systems*. Springer, 2021. DOI: 10.1007/978-3-030-77222-2.

[25] J. Daemen, L. Knudsen e V. Rijmen. «The block cipher Square». In: *Fast Software Encryption (FSE)*. Lecture Notes in Computer Science. 1997, pp. 149–165. DOI: 10.1007/BFb0052343.

[26] L. Knudsen e D. Wagner. «Integral Cryptanalysis». In: *Fast Software Encryption (FSE)*. Lecture Notes in Computer Science. 2002, pp. 112–127. DOI: 10.1007/3-540-45661-9_9.

[27] E. Biham, O. Dunkelman e N. Keller. «New Results on Boomerang and Rectangle Attacks». In: *Fast Software Encryption (FSE)*. Lecture Notes in Computer Science. 2002, pp. 1–16. DOI: 10.1007/3-540-45661-9_1.

[28] C. Boura e A. Canteaut. «On the Boomerang Uniformity of Cryptographic Sboxes». In: *IACR Transactions on Symmetric Cryptology* 2018.3 (2018), pp. 290–310. DOI: 10.13154/tosc.v2018.i3.290-310.

[29] C. Cid et al. «Boomerang Connectivity Table: A New Cryptanalysis Tool». In: *Advances in Cryptology - EUROCRYPT 2018*. Lecture Notes in Computer Science. 2018, pp. 683–714. DOI: 10.1007/978-3-319-78375-8_22.

[30] M. Blunden e A. Escott. «Related Key Attacks on Reduced Round KASUMI». In: *Fast Software Encryption (FSE)*. Lecture Notes in Computer Science. 2002, pp. 277–285. DOI: 10.1007/3-540-45473-X_23.

[31] A. Biryukov e D. Khovratovich. «Related-Key Cryptanalysis of the Full AES-192 and AES-256». In: *Advances in Cryptology - ASIACRYPT 2009*. Lecture Notes in Computer Science. 2009, pp. 1–18. DOI: 10.1007/978-3-642-10366-7_1.

[32] L. K. Grover. «A fast quantum mechanical algorithm for database search». In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*. 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[33] NIST. *Announcing Development of a FIPS for Advanced Encryption Standard*. 1997. URL: https://csrc.nist.gov/news/1997/announcing-development-of-fips-for-advanced-encryp.

[34] A. Bogdanov, D. Khovratovich e C. Rechberger. «Biclique Cryptanalysis of the Full AES». In: *Advances in Cryptology - ASIACRYPT 2011*. Lecture Notes in Computer Science. 2011, pp. 344–371. DOI: 10.1007/978-3-642-25385-0_19.

[35] ISO. *Information technology – Security techniques – Modes of operation for an n-bit block cipher*. ISO/IEC 10116. 2017. URL: https://www.iso.org/standard/64575.html.

[36] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. SP 800-38A. NIST, 2001. DOI: 10.6028/NIST.SP.800-38A.

[37] ACN. *Codici di Autenticazione di Messaggi (MAC)*. Linee Guida Funzioni Crittografiche, 2026. URL: https://www.acn.gov.it/portale/crittografia.

[38] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. SP 800-38A - Addendum. NIST, 2010. DOI: 10.6028/NIST.SP.800-38A-Add.

[39] IEEE. *IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices*. Std 1619-2007. 2008. DOI: 10.1109/IEEESTD.2008.4493450.

[40] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*. SP 800-38E. NIST, 2010. DOI: 10.6028/NIST.SP.800-38E.

[41] P. Rogaway. «Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC». In: *Advances in Cryptology - ASIACRYPT 2004*. Lecture Notes in Computer Science. 2004, pp. 16–31. DOI: 10.1007/978-3-540-30539-2_2.

[42] ISO. *Information security – Authenticated encryption*. ISO/IEC 19772. 2020. URL: https://www.iso.org/standard/81550.html.

[43] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*. SP 800-38C. NIST, 2007. DOI: 10.6028/NIST.SP.800-38C.

[44] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. SP 800-38D. NIST, 2007. DOI: 10.6028/NIST.SP.800-38D.

[45] ISO. *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*. ISO/IEC 9797-1. 2011. URL: https://www.iso.org/standard/50375.html.

[46] S. Kent. *IP Encapsulating Security Payload (ESP)*. RFC 4303. 2005. DOI: 10.17487/RFC4303.

[47] R. Housley. *Cryptographic Message Syntax (CMS)*. RFC 5652. 2009. DOI: 10.17487/RFC5652.

[48] X. Wang et al. «Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS...» In: *Advances in Cryptology - EUROCRYPT 2002*. Lecture Notes in Computer Science. 2002, pp. 534–545. DOI: 10.1007/3-540-46035-7_35.