Azzolini Riccardo 2021-04-08

# Modalità di funzionamento per la cifratura a blocchi

## 1 Modalità per la cifratura a flussi

Le modalità di funzionamento viste finora, ECB e CBC, risolvono il problema di come cifrare messaggi più grandi dei blocchi su cui opera l'algoritmo di cifratura impiegato. Ci sono invece situazioni in cui si ha il problema opposto: ciascun messaggio ha una dimensione (anche molto) più piccola di quella del blocco (ad esempio 8 bit), e si deve cifrare un **flusso** (**stream**) di tali messaggi.

Una soluzione sarebbe utilizzare un cifrario a flussi invece che a blocchi, ma gli standard di cifratura più diffusi sono quelli a blocchi, dunque spesso si preferirebbe usare questi. Allora, si potrebbe pensare di raccogliere i messaggi in un buffer fino a riempire un blocco, che verrebbe poi cifrato e inviato tutto insieme. Così, però, si introduce un ritardo nell'invio, soprattutto se i messaggi arrivano con una frequenza bassa e/o irregolare (ad esempio, si potrebbe immaginare che il buffer sia quasi pieno, ma l'ultimo messaggio necessario per riempirlo impieghi molto tempo ad arrivare, ritardando l'invio di tutti i messaggi nel buffer). In certi scenari, questa latenza non è accettabile: si vogliono cifrare e inviare i messaggi uno a uno, come se fossero appunto un flusso continuo di bit.

Anche un'altra soluzione intuitiva, aggiungere del padding a ciascun messaggio per fare in modo che riempia da solo un intero blocco, non è in realtà accettabile, perché aumenterebbe inutilmente di molto la quantità di dati da trasmettere: ad esempio, se si cifrassero in questo modo messaggi di 8 bit con AES, bisognerebbe inviare 128 bit cifrati per ogni 8 bit di dati in chiaro.

Poiché le soluzioni semplici non sono adeguate, per risolvere il problema è stato necessario definire delle modalità di funzionamento per gli algoritmo di cifratura a blocchi che consentono di utilizzarli per eseguire la cifratura a flussi. Le principali modalità di questo tipo verranno presentate in seguito.

## 2 Cipher Feedback

Il modo migliore per cifrare un flusso di dati indipendentemente dalla dimensione dell'*unità di flusso* (o *di trasmissione*), cioè dei singoli messaggi, è cifrarlo bit a bit. Un'operazione che permette di fare ciò è lo XOR bit a bit tra le unità di flusso e una sequenza

di bit che dipende da una chiave. Il destinatario, che conosce la chiave, può usarla per generare la stessa sequenza di bit e fare ancora lo XOR, recuperando così i messaggi in chiaro.

I bit da mettere in XOR con un'unità di flusso possono essere generati tramite un algoritmo di cifratura a blocchi, dando a esso in input un blocco con un valore sostanzialmente casuale, che costituisce il vettore di inizializzazione, ed estraendo dal blocco cifrato risultante tanti bit quanti servono per lo XOR (cioè un numero di bit pari a quello dell'unità di flusso). Ad esempio, il primo messaggio verrebbe cifrato secondo la formula

$$C_1 = P_1 \oplus \mathrm{HEAD}(E_K(IV), s)$$

dove:

- $P_1$ è il plaintext del primo messaggio, e $C_1$ è il corrispondente ciphertext;
- $IV$ è il vettore di inizializzazione;
- $E_K$ rappresenta la cifratura con un qualche algoritmo di cifratura a blocchi, applicato con una chiave $K$;
- $s$ è la dimensione in bit dell'unità di flusso, ovvero di $P_i$ e $C_i$, e di conseguenza è il numero di bit che servono per lo XOR;
- la funzione $\mathrm{HEAD}(X, s)$ estrae i primi $s$ bit (più significativi, cioè più a sinistra) da un blocco $X$.

Per decifrare il messaggio ricevuto, il destinatario deve conoscere i valori della chiave (segreta) e del vettore di inizializzazione (non necessariamente segreto), ai quali applica ancora l'algoritmo di cifratura a blocchi, come fatto dal mittente, ricavando in tal modo la sequenza di bit con cui annullare lo XOR:

$$P_1 = C_1 \oplus \mathrm{HEAD}(E_K(IV), s)$$

I vantaggi di uno schema di questo tipo sono che:

- è un modo facile di generare una sequenza di bit dipendente da una chiave;
- sfrutta tutte le caratteristiche (di robustezza, ecc.) dell'algoritmo di cifratura a blocchi impiegato;
- è adattabile a dimensioni dell'unità di flusso da un singolo bit fino alla dimensione del blocco dell'algoritmo di cifratura.

Rimane però un problema: se la stessa identica operazione di cifratura $P_i = C_i \oplus \mathrm{HEAD}(E_K(IV), s)$ venisse ripetuta anche per i messaggi successivi al primo, la sequenza di bit usata nello XOR sarebbe la stessa, dunque per messaggi in chiaro uguali si otterrebbero ciphertext uguali. La soluzione è analoga a quella adottata per risolvere lo stesso problema nella modalità CBC: organizzare le operazioni di cifratura in una "catena", nella quale ciascuna sequenza di bit usata nello XOR a ciascun passo (generata

dal cifrario a blocchi), e di conseguenza ciascun ciphertext, dipende da tutti i plaintext precedenti, così anche per plaintext uguali si hanno ciphertext diversi. Si noti che per cambiare la sequenza di bit usata nello XOR è necessario modificare l'input dell'algoritmo di cifratura a blocchi, come in CBC, ma qui tale input non è il plaintext (che non passa dalla cifratura a blocchi), bensì il vettore di inizializzazione.

In base al principio appena descritto si definisce la modalità di funzionamento **CFB**, **Cipher Feedback**. Essa usa come input dell'algoritmo di cifratura a blocchi il contenuto di un **registro a scorrimento** (*shift register*) sinistro:

1. inizialmente, il registro contiene il valore del vettore di inizializzazione;
2. dopo ogni passo di cifratura, il contenuto del registro viene aggiornato facendo scorrere verso sinistra di $s$ posizioni i bit precedentemente contenuti nel registro (il che comporta l'eliminazione degli $s$ bit più a sinistra) e inserendo a destra gli $s$ bit risultanti dall'operazione di XOR.

Formalmente, la cifratura in modalità CFB è espressa dalla formula

$$\boxed{C_i = P_i \oplus \mathrm{HEAD}(E_K(R_i), s) \quad \text{per } i \geq 1}$$

dove $R_i$ indica lo *stato* (cioè il contenuto) del registro all'$i$-esimo passo: si fissa $R_1 = IV$, dopodiché ciascun valore $R_j$ per $j > 1$ viene ricavato a partire dal precedente valore $R_{j-1}$, secondo la procedura di scorrimento dei bit e inserimento del ciphertext $C_{j-1}$ appena descritta. La decifratura è analoga, perché consiste nell'annullare lo XOR facendo un secondo XOR con la stessa quantità:

$$\boxed{P_i = C_i \oplus \mathrm{HEAD}(E_K(R_i), s) \quad \text{per } i \geq 1}$$

Si osservi in particolare che il cifrario a blocchi viene impiegato *sempre in modalità di cifratura* ($E_K$), e mai in modalità di decifratura ($D_K$).

Il nome di questa modalità, "Cipher Feedback", si riferisce al fatto che:

- "Feedback": si usa un qualche risultato di ciascun passo per generare lo stato del registro al passo successivo;
- "Cipher": come feedback si usa in particolare il ciphertext del messaggio da inviare, ottenuto dall'operazione di XOR.

## 2.1 Caratteristiche

- La modalità CFB *non richiede padding*, perché dall'output dell'algoritmo di cifratura a blocchi vengono estratti solo i bit necessari a fare lo XOR bit a bit con il messaggio in chiaro, ottenendo così un ciphertext lungo esattamente quanto il plaintext.

- La cifratura di più messaggi *non può* essere eseguita in parallelo,[1] perché lo stato $R_i$ del registro all'$i$-esimo passo dipende dal ciphertext ottenuto alla fine dell'$(i-1)$-esimo passo. Invece, la decifratura *può* essere eseguita in parallelo: un destinatario che ha ricevuto una sequenza di ciphertext può precomputare tutti i corrispondenti stati del registro, e poi usare questi per decifrare in parallelo i singoli messaggi.
- Un errore di trasmissione in un ciphertext $C_i$ comporta, come sempre, la decifratura errata del corrispondente plaintext, ma in più influisce anche sulla generazione degli stati successivi del registro, quindi l'errore viene propagato ai passi $i+1$, $i+2$, ecc., finché i bit corrotti di $C_i$ non escono dal registro. Il numero esatto di passi a cui l'errore viene propagato dipende dalle dimensioni dell'unità di flusso e del registro (ovvero del blocco del cifrario impiegato), ma tipicamente è piuttosto elevato (ad esempio, se l'unità di flusso è di 8 bit e si usa l'algoritmo AES, che ha blocchi di 128 bit, un errore di trasmissione in un ciphertext determina stati del registro corrotti nei 16 passi successivi, dunque si ha la decifratura errata di 17 messaggi).

[1] Nel caso della cifratura a blocchi, quando si parla di eseguire la cifratura o decifratura in parallelo si può supporre di operare appunto in parallelo su tutti i blocchi del messaggio. Invece, nel caso della cifratura o decifratura di un flusso di messaggi è spesso impossibile operare in parallelo su un intero flusso (dato che i messaggi potrebbero arrivare in momenti anche molto diversi, e il numero di messaggi potrebbe essere teoricamente infinito), ma rimane comunque utile gestire in parallelo un gruppo di messaggi ricevuti insieme.