Azzolini Riccardo 2021-04-12

Modalità di funzionamento per la cifratura a blocchi

1 Output Feedback

Gli svantaggi della modalità di funzionamento CFB possono essere risolti usando come feedback non il ciphertext, bensì l'output dell'algoritmo di cifratura a blocchi. La modalità così ottenuta si chiama **OFB**, **Output Feedback**. Le formule che esprimono la cifratura e la decifratura in OFB sono uguali a quelle di CFB,

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

e lo stato iniziale del registro è ancora $R_1 = IV$, ma ciascuno stato successivo, R_j per $j > 1$, è calcolato inserendo alla destra del registro i bit $\text{HEAD}(E_K(R_{j-1}), s)$, la parte dell'output dell'algoritmo a blocchi che è stata messa in XOR con il plaintext precedente, invece dei bit di C_{j-1} risultanti da tale XOR.

Con questa piccola modifica si ottengono degli importanti vantaggi rispetto a CFB. Infatti, le caratteristiche della modalità OFB sono le seguenti:

- *Non è richiesto padding*, come per CFB.
- Siccome il feedback non dipende dal plaintext, dati IV e K è possibile *computare in anticipo* le cifrature $E_K(R_i)$ del registro, e successivamente *eseguire in parallelo la cifratura o decifratura* di più messaggi. Tuttavia, l'unica operazione che può essere fatta in parallelo è lo XOR, il quale in genere costa poco (è molto veloce) anche se eseguito in modo sequenziale. Allora, il vantaggio è più che altro una riduzione della latenza di cifratura/decifratura: se si computano le cifrature del registro prima ancora di ricevere i messaggi, quando poi questi arrivano bisogna eseguire solo uno XOR per cifrarli/decifrarli.
- *Non si ha propagazione dell'errore*: un errore di trasmissione in un ciphertext C_i impatta *solo* il corrispondente plaintext P_i, dato che C_i non viene usato per il feedback.

2 Counter

Le modalità CFB e OFB sono pensate per la cifratura di flussi di messaggi più piccoli del blocco dell'algoritmo di cifratura, ma possono essere utili anche per la cifratura a blocchi (cioè per messaggi più grandi, che vengono cifrati suddividendoli in blocchi), perché hanno alcuni vantaggi rispetto ad esempio alla modalità CBC: l'assenza di padding, la bassa latenza di cifratura/decifratura ottenibile con OFB (precomputando le cifrature del registro), ecc.

Per ottenere tali vantaggi nella cifratura a blocchi è stata definita una nuova modalità di funzionamento, chiamata **CTR**, **Counter**, che impiega un contatore come registro dato in input all'algoritmo di cifratura a blocchi, e poi mette la risultante cifratura del contatore in XOR con il plaintext per ottenere il ciphertext. La decifratura avviene esattamente allo stesso modo, eseguendo lo XOR del ciphertext con la stessa cifratura del contatore per ottenere il plaintext. In sostanza, questa modalità è simile a OFB, con la differenza che il registro/contatore viene aggiornato non tramite feedback, ma incrementandolo in qualche modo a ogni passo, a partire da un valore iniziale.

Ci sono diversi modi per generare i valori del contatore. In genere si suddivide il contatore in due parti: una fissa, chiamata *nonce*, e una che viene aggiornata a ogni passo per mezzo di una qualche funzione di incremento. In questo schema possono variare il modo di generare il nonce e la funzione di incremento.

I vantaggi della modalità CTR sono sostanzialmente gli stessi di OFB:

- *Non è richiesto padding*, cioè il messaggio non deve per forza essere un multiplo del blocco.
- Una volta stabiliti la chiave, il nonce e la funzione di incremento è possibile *precomputare le cifrature del contatore* per ridurre la latenza di cifratura/decifratura. Inoltre, al contrario di OFB, CTR permette di *applicare in parallelo l'algoritmo di cifratura a blocchi* per computare le cifrature del contatore: non essendoci feedback, ciascuno stato del contatore dipende solo dallo stato precedente, e non dall'output dell'algoritmo di cifratura, quindi si possono precomputare tali stati e poi cifrare ciascuno di essi in parallelo. Allora, complessivamente, *la cifratura/decifratura può essere eseguita in parallelo* a tutti gli effetti.
- *Non si ha propagazione dell'errore.*

La modalità CTR è adatta principalmente per la cifratura a blocchi, e non per quella a flussi, perché non può cifrare in modo sicuro flussi potenzialmente infiniti. Infatti, è importante che il valore del contatore sia diverso per ciascun blocco di plaintext da cifrare (altrimenti blocchi uguali cifrati con lo stesso valore del contatore produrrebbero blocchi di ciphertext uguali, dai quali un attaccante potrebbe ricavare informazioni sul plaintext), ma la dimensione del contatore è fissa, dunque su un flusso di lunghezza maggiore di un certo limite la funzione di incremento genererebbe valori ripetuti. Invece,

nell'ambito della cifratura a blocchi di messaggi di lunghezza finita e limitata si può cambiare il nonce (ed eventualmente anche la chiave) ogni volta che si cifra un nuovo messaggio, ovvero usare uno stesso nonce solo per i blocchi di uno stesso messaggio, eliminando così il problema della ripetizione dei valori.