Azzolini Riccardo 2021-03-15

Rete di Feistel

1 Reti di sostituzioni e permutazioni

Secondo Claude Shannon, il cifrario ideale dovrebbe oscurare le proprietà statistiche del messaggio originale. A tale scopo, Shannon suggerì di combinare sostituzioni e permutazioni/trasposizioni in modo da ottenere algoritmi di cifratura con due particolari proprietà:

- **diffusione**: la capacità di distribuire, "diffondere" nel testo cifrato le correlazioni statistiche presenti nel testo in chiaro (cioè nascondere le frequenze delle singole lettere e dei poligrammi, ecc.);[1]
- **confusione**: la relazione tra la chiave e il testo cifrato deve essere il più possibile complessa e scorrelata.

Shannon introdusse dunque l'idea delle **reti di sostituzioni e permutazioni** (*substitution-permutation network*, *S-P network*), molto usate nella crittografia moderna.

2 Rete di Feistel

Horst Feistel, un crittografo che lavorava come ricercatore presso IBM, fu uno dei primi a proporre una cifratura basata sulle reti S-P: la **rete di Feistel** (o *struttura di cifratura di Feistel*). Di per sé, questa non è un cifrario direttamente utilizzabile, ma piuttosto è una struttura generale sulla quale si possono basare dei cifrari concreti; un esempio di algoritmo basato su tale struttura è il DES.

La rete di Feistel è un **cifrario del prodotto**:[2] la cifratura è eseguita in più **fasi** o **round**, ciascuna delle quali consiste nell'eseguire le *stesse operazioni* (sostituzioni e permutazioni) con una *chiave diversa*, che prende il nome di **sottochiave di fase** o **di round**. Ciò non significa però che sia necessario fornire in input più chiavi: le sottochiavi di fase vengono generate da un'unica chiave data in input, tramite un apposito algoritmo

[1] La diffusione può essere misurata osservando quanto cambia il blocco di testo cifrato al variare di un singolo bit del blocco di testo in chiaro: il cifrario ha una buona capacità di diffusione se cambiano circa il 50 % dei bit del ciphertext.

[2] Il termine "cifrario del prodotto" si riferisce al fatto che, se si interpreta ciascuna fase come una funzione (in senso matematico), allora complessivamente il cifrario corrisponde a una composizione, un "prodotto" di funzioni.

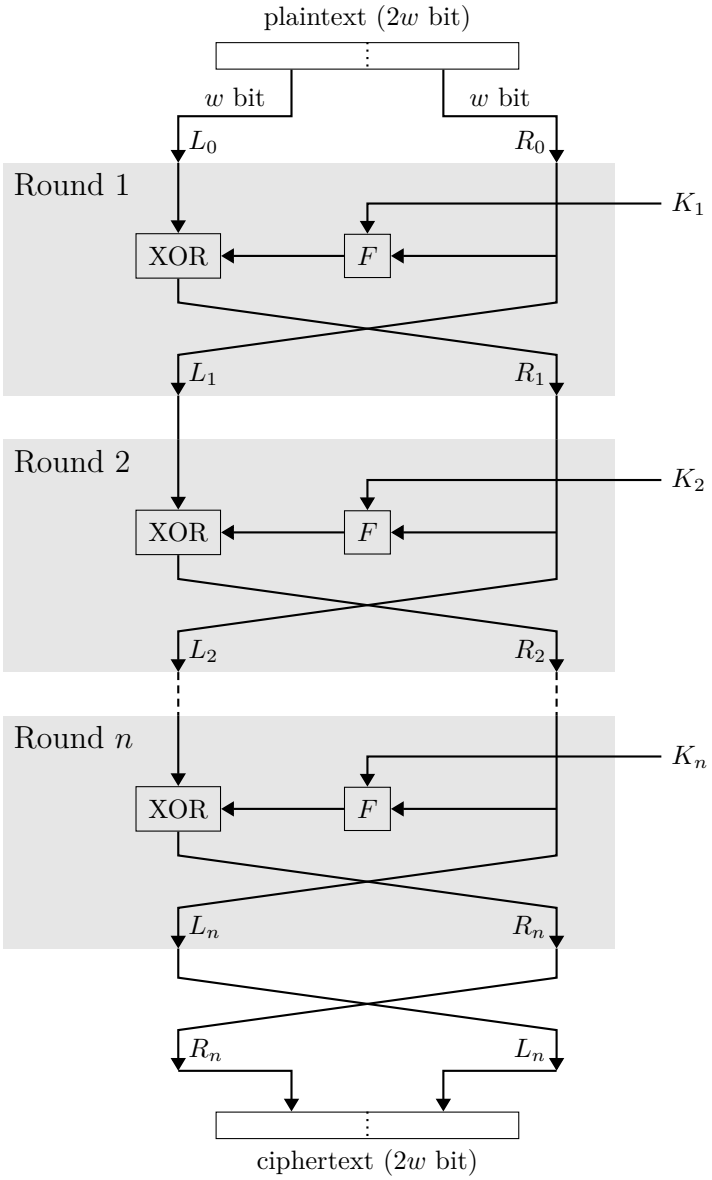
(che la struttura della rete di Feistel non fissa, e va invece specificato quando si definisce un cifrario concreto).

Una caratteristica particolare della rete di Feistel è che la cifratura e la decifratura sono praticamente identiche: come si vedrà in seguito, l'unica differenza tra queste due operazioni è l'inversione dell'ordine in cui vengono usate le sottochiavi. Ciò significa che la stessa implementazione può essere usata per cifrare e per decifrare, il che è un vantaggio soprattutto quando l'algoritmo viene realizzato a livello hardware tramite un circuito dedicato.

2.1 Funzionamento

Gli input del processo di cifratura sono il plaintext, che deve essere un blocco di dati formato da un numero pari di bit, e una chiave, dalla quale vengono in qualche modo generate tante sottochiavi diverse quante sono le fasi da eseguire.

La dimensione del plaintext deve essere pari, $2w$ bit, perché esso viene suddiviso in due metà, ciascuna di w bit, le quali attraversano n fasi di elaborazione, poi vengono scambiate (il perché sarà spiegato in seguito) e ricombinate per ottenere il blocco di testo cifrato, ancora di $2w$ bit.



Si consideri, in generale, l'i-esima fase della cifratura (per $i = 1, \ldots, n$). I dati che essa riceve in input sono la sottochiave K_i e le due metà del blocco generato dalla fase precedente — la metà sinistra L_{i-1} (i primi w bit del blocco) e la metà destra R_{i-1} (i restanti w bit del blocco). L'input destro R_{i-1} diventa direttamente l'output sinistro L_i, senza subire alcuna modifica. Tuttavia, R_{i-1} va anche in input, insieme alla sottochiave K_i, a una qualche **funzione di round** F, e il risultato di quest'ultima viene messo in OR esclusivo (XOR, $\oplus$) bit a bit con l'input sinistro L_{i-1} per calcolare l'output destro R_i. La rete di Feistel non indica una specifica funzione F da usare; gli unici vincoli sono che essa produca un risultato di w bit (in modo da poter calcolare lo XOR bit a bit

con metà dell'input) e che si usi la stessa funzione in tutte le fasi (perché ogni fase deve essere identica alle altre: deve cambiare solo la sottochiave usata).

Complessivamente, l'output dell'i-esima fase è:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \end{aligned}$$

In sostanza, ciascuna fase sostituisce solo la metà sinistra dell'input, lasciando la destra inalterata (non cifrata, se ci fosse una sola fase), ma poi scambia le due metà per far sì che la fase successiva sostituisca l'altra metà.

Se si considera una singola fase, l'output dello XOR (ovvero l'output destro della fase, R_i) dipende dalle due metà dell'input e dalla sottochiave. Tuttavia, considerando insieme più fasi, si osserva che la funzione F riceve ogni volta in input l'output dello XOR precedente, quindi il risultato dello XOR di una fase dipende anche dagli input e dalle sottochiavi di *tutte* le fasi precedenti.

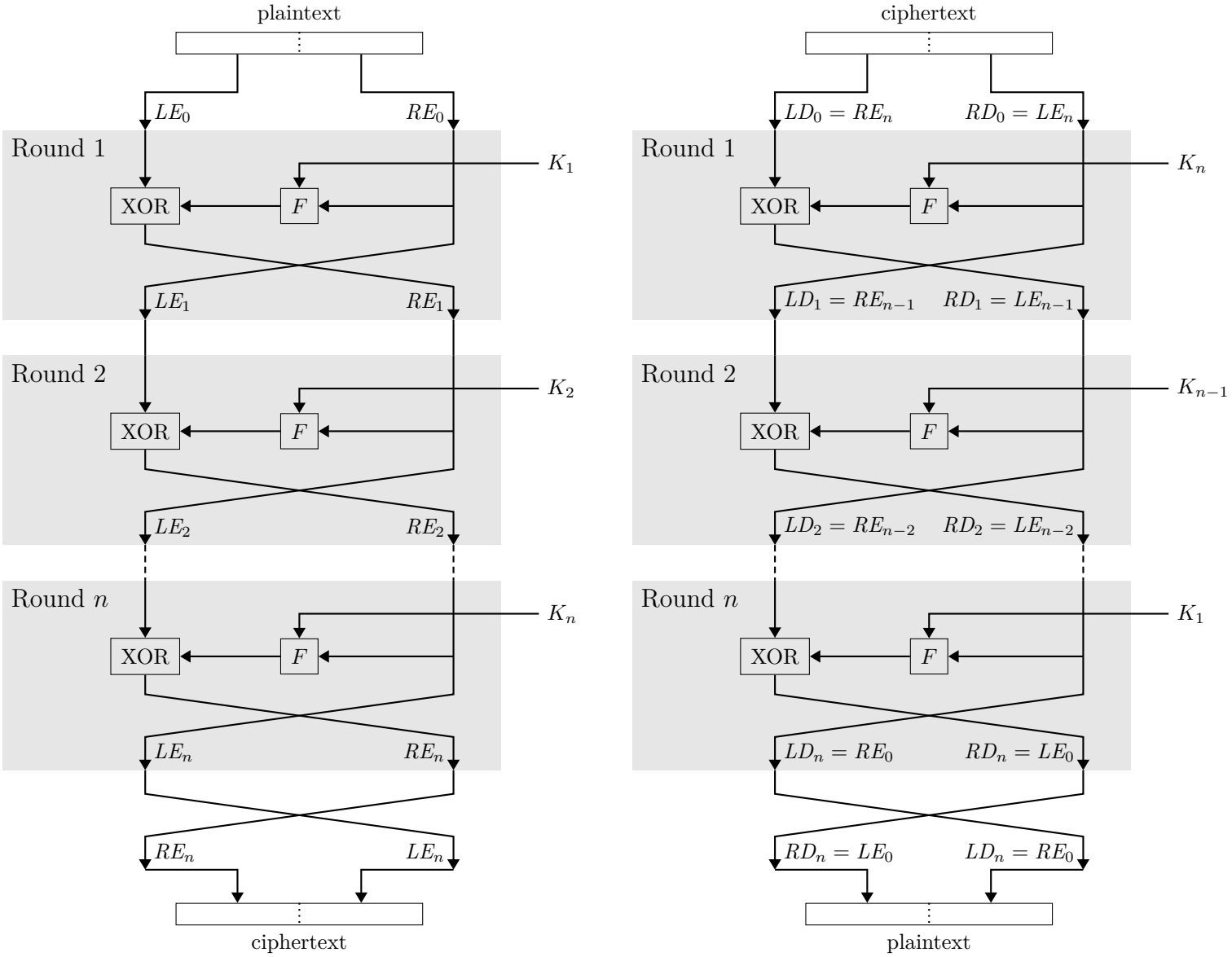
Come già accennato, le due metà generate in output dall'ultimo round vengono scambiate prima di ricombinarle per ottenere l'intero blocco di testo cifrato. Si potrebbe osservare che ciò equivale a eliminare lo scambio effettuato all'interno dell'ultimo round, ma aggiungere uno scambio finale è preferibile perché altrimenti l'ultimo round dovrebbe essere diverso dagli altri, il che complicherebbe soprattutto un'implementazione hardware (non sarebbe possibile utilizzare esattamente lo stesso circuito per eseguire tutti i round).

2.2 Decifratura

A prescindere da come sia fatta la funzione di round F, la decifratura avviene semplicemente dando il testo cifrato in input alla stessa identica rete usata per la cifratura, ma usando le sottochiavi di fase in ordine inverso (K_n per la fase 1, K_{n-1} per la fase 2, e così via, fino a K_1 per la fase n). Così, infatti, per come è strutturata la rete di Feistel, l'output (LD_i, RD_i) dell'i-esima fase di decifratura è uguale allo scambio delle metà dell'input (LE_{n-i}, RE_{n-i}) della $(n-i)$-esima fase di cifratura (dove n è il numero di fasi):

$$\forall i = 1, \ldots, n \qquad \begin{aligned} LD_i &= RE_{n-i} \\ RD_i &= LE_{n-i} \end{aligned}$$

Allora, in particolare, l'output (LD_n, RD_n) dell'n-esima e ultima fase di decifratura corrisponde allo scambio delle metà dell'input (LE_0, RE_0) della prima fase di cifratura, ovvero allo scambio delle metà del testo in chiaro. Allora, dopo lo scambio finale che segue le fasi della rete di Feistel si ottiene esattamente il testo in chiaro, con le due metà disposte nell'ordine originale.



Per capire come mai, a livello matematico, siano valide le relazioni $LD_i = RE_{n-i}$ e $RD_i = LE_{n-i}$, è comodo ragionare su un numero specifico di fasi, ad esempio $n = 3$.

Innanzitutto, siccome l'input della decifratura è il testo cifrato, ricordando lo scambio finale si ha che la metà sinistra dell'input è l'output destro dell'ultima fase di cifratura, e analogamente la metà destra dell'input è l'output sinistro dell'ultima fase di cifratura:

$$LD_0 = RE_3 \qquad RD_0 = LE_3$$

Ora, bisogna iniziare a dimostrare che l'output della prima fase di decifratura coincide con lo scambio dell'input dell'ultima fase di cifratura.

- Per la definizione della fase della rete di Feistel, l'output sinistro della fase di decifratura è uguale al suo input destro, $LD_1 = RD_0$, e quest'ultimo, come osservato prima, è l'output sinistro dell'ultima fase di cifratura, $RD_0 = LE_3$, che ancora, per definizione, è uguale all'input destro di tale fase, $LE_3 = RE_2$. Complessivamente:

$$\begin{aligned} LD_1 &= RD_0 && \text{[definizione della fase]} \\ &= LE_3 && \text{[input della decifratura]} \\ &= RE_2 && \text{[definizione della fase]} \end{aligned}$$

- Considerando invece l'output destro della fase di decifratura, per definizione esso è dato dalla formula

$$RD_1 = LD_0 \oplus F(RD_0, K_3)$$

(ricordando che le sottochiavi vengono usate in ordine inverso). Applicando le osservazioni precedenti sugli input della decifratura, $LD_0 = RE_3$ e $RD_0 = LE_3$, il lato destro della formula può essere riscritto come:

$$RD_1 = RE_3 \oplus F(LE_3, K_3)$$

Poi, si inseriscono nella formula le definizioni degli output della fase di cifratura, $LE_3 = RE_2$ e $RE_3 = LE_2 \oplus F(RE_2, K_3)$, ottenendo così:

$$RD_1 = (LE_2 \oplus F(RE_2, K_3)) \oplus F(RE_2, K_3)$$

Infine, per le proprietà dello XOR, $F(RE_2, K_3) \oplus F(RE_2, K_3)$ si annulla:

$$RD_1 = LE_2$$

Si è dunque dimostrato che le relazioni $LD_i = RE_{n-i}$ e $RD_i = LE_{n-i}$ valgono per $i = 1$, cioè per la prima fase di decifratura:

$$LD_1 = RE_2 \qquad RD_1 = LE_2$$

Da questa dimostrazione si può dedurre lo scopo dello scambio presente dopo l'ultima fase di cifratura: esso serve a fare in modo che il risultato dell'ultimo XOR sia rimandato direttamente nel primo XOR della decifratura, al fine di annullare la cifratura effettuata dallo XOR. Senza questo scambio, invece, il risultato dello XOR verrebbe mandato alla funzione F: invece di decifrare, si eseguirebbe di fatto un'altra fase di cifratura.

Adesso, la dimostrazione continua considerando la seconda fase di decifratura, $i = 2$, con un ragionamento sostanzialmente analogo.

$$\begin{aligned} LD_2 &= RD_1 && \text{[definizione della fase]} \\ &= LE_2 && \text{[output della prima fase di decifratura]} \\ &= RE_1 && \text{[definizione della fase]} \end{aligned}$$

$$
\begin{aligned}
RD_2 &= LD_1 \oplus F(RD_1, K_2) && \text{[definizione della fase]} \\
&= RE_2 \oplus F(LE_2, K_2) && \text{[output della prima fase di decifratura]} \\
&= (LE_1 \oplus F(RE_1, K_2)) \oplus F(RE_1, K_2) && \text{[definizione della fase]} \\
&= LE_1 && \text{[annullamento dello XOR]}
\end{aligned}
$$

Sempre con lo stesso ragionamento si dimostra che per l'ultima fase di decifratura ($i = 3$) valgono $LD_3 = RE_0$ e $RD_3 = LE_0$, da cui segue che il blocco di dati $(RD_3, LD_3) = (LE_0, RE_0)$ ottenuto dopo lo scambio finale è proprio il plaintext decifrato.

Tutto il procedimento appena svolto potrebbe essere generalizzato a un numero arbitrario di fasi, ripetendo più o meno volte il passo di dimostrazione per la singola fase.

2.3 Parametri

La rete di Feistel definisce solo la struttura di un algoritmo di cifratura, lasciando a chi definisce un algoritmo concreto il compito di fissare i seguenti parametri:

- la **dimensione del blocco** e la **dimensione della chiave** — aumentare queste dimensioni migliora la sicurezza ma peggiora le prestazioni;
- il **numero di fasi** — anche in questo caso, aumentando questo parametro si migliora la sicurezza a scapito delle prestazioni;
- l'algoritmo di **generazione delle sottochiavi** e la **funzione di round** — più questi sono complessi, più è difficile la crittoanalisi.