Azzolini Riccardo 2021-03-18

Data Encryption Standard

1 Storia

Alla fine degli anni '60, IBM iniziò un progetto di ricerca sulla crittografia simmetrica, diretto da Feistel, che portò allo sviluppo di Lucifer, un algoritmo basato sulla rete di Feistel che operava su blocchi di 64 bit con una chiave di 128 bit.

Nel 1972, il National Bureau of Standards (NBS) emise una richiesta di proposte per un algoritmo di cifratura da adottare come standard per tutte le agenzie governative americane. Tale algoritmo doveva soddisfare dei requisiti rigorosi, e in questa prima "gara" non fu proposto nessun algoritmo soddisfacente. Nel 1975 fu allora indetta una seconda gara, alla quale partecipò anche IBM, inviando una versione modificata di Lucifer, che vinse e fu adottata come **Data Encryption Standard** (**DES**).

Quando il DES fu pubblicato per ricevere commenti dalla comunità, ci furono due critiche, legate alle differenze del DES rispetto a Lucifer:

- la lunghezza della chiave era stata ridotta da 128 a 56 bit;
- alcuni criteri progettuali erano stati tenuti segreti.

Per fare queste modifiche, IBM aveva lavorato con dei crittografi che collaboravano con l'NSA, quindi la comunità sospettava che l'NSA avesse intenzionalmente indebolito l'algoritmo, introducendo delle backdoor per facilitare la decifratura dei messaggi senza la conoscenza della chiave.

A causa di questi sospetti, il DES è stato molto analizzato dalla comunità di ricerca crittografica, e tali studi hanno dimostrato che l'algoritmo è molto robusto. In particolare, negli anni '90 furono pubblicati i primi attacchi di *crittoanalisi differenziale*, un nuovo tipo di attacco chosen plaintext, e si scoprì che il DES era molto resistente a questi attacchi (servivano 2^{47} chosen plaintext, un numero che in pratica è impossibile ottenere), mentre la versione originale di Lucifer non lo era (bastavano 256 chosen plaintext, dunque l'attacco risultava concretamente fattibile). Perciò, si pensò che già al tempo della progettazione del DES, quasi 20 anni prima, l'NSA conoscesse la crittoanalisi differenziale, e avesse quindi modificato Lucifer proprio per renderlo resistente a essa; siccome molti altri algoritmi di cifratura del tempo erano vulnerabili a questi attacchi, per evitare di renderli noti al pubblico sarebbe stato necessario tenere segreti alcuni criteri progettuali del DES. Tale ipotesi fu sostanzialmente confermata pochi anni dopo, quando uno dei progettisti del DES pubblicò alcuni di questi criteri progettuali.

Nel 1994, l'NBS — ora chiamato National Institute of Standards and Technology (NIST) — riaffermò il DES per l'utilizzo a livello federale per altri 5 anni (ma solo per la cifratura di informazioni *non* confidenziali). Poi, nel 1999, il NIST richiese l'adozione del *Triple DES*, che rende più difficili gli attacchi a forza bruta triplicando la lunghezza della chiave, poiché lo spazio delle chiavi di 56 bit cominciava a essere troppo piccolo. Infine, nel 2002 DES fu rimpiazzato da un nuovo standard, AES (Advanced Encryption Standard), ma principalmente per motivi di efficienza dell'implementazione, perché il Triple DES è ancora considerato sicuro, tanto è vero che nel 2005 il NIST ne approvò l'utilizzo fino all'anno 2030.

2 Struttura dell'algoritmo

La struttura dell'algoritmo DES può essere suddivisa in due parti: la rete di Feistel e l'algoritmo di **key scheduling**, cioè di generazione delle sottochiavi.

La rete di Feistel opera su un *blocco di 64 bit*, suddivisi in due metà di 32 bit, e prevede *16 round*, seguiti, come al solito, da uno scambio finale delle due metà del blocco. In più, DES applica ai bit del blocco una **permutazione iniziale** (*IP*, *Initial Permutation*) prima di mandarli in input alla rete di Feistel. L'output della rete di Feistel (dopo lo scambio finale) è chiamato blocco di pre-output; per ottenere il blocco di output vero e proprio, si applica a esso una permutazione che è l'inversa di quella iniziale.

La chiave data in input all'algoritmo è di 64 bit, ma 8 bit sono usati come bit di parità (per il controllo degli errori nella trasmissione della chiave), quindi la dimensione effettiva della chiave è *56 bit*. Da questa, l'algoritmo di key scheduling genera le 16 *sottochiavi di 48 bit* usate per i 16 round.

3 Funzione di round

Uno dei parametri da specificare quando si implementa una rete di Feistel è la funzione di round F. Nel caso del DES, all'i-esimo round essa riceve in input i 48 bit della sottochiave K_i e i 32 bit della metà destra R_{i-1} del blocco prodotto al round precedente, e deve produrre un output $F(R_{i-1}, K_i)$ di 32 bit (in modo che l'output possa essere messo in XOR bit a bit con la metà sinistra del blocco, L_{i-1}, anch'essa di 32 bit). Tale output viene calcolato nel seguente modo:

1. R_{i-1} viene espanso da 32 a 48 bit, tramite un'operazione di permutazione ed espansione;
2. il risultato dell'espansione viene combinato in XOR con la sottochiave K_i;
3. il risultato dello XOR viene passato in input a una funzione di sostituzione, che dà un output di 32 bit;

4. questi 32 bit vengono infine permutati.

3.1 Permutazione ed espansione E

La permutazione ed espansione di R_{i-1} da 32 a 48 bit avviene, in sintesi, suddividendo i 32 bit in gruppi di 4 e duplicando i "bordi" dei gruppi. Più nel dettaglio, all'inizio di ogni gruppo di 4 bit viene aggiunto l'ultimo bit del gruppo precedente, e alla fine di ogni gruppo viene aggiunto il primo bit del gruppo successivo. In particolare, all'inizio del primo gruppo, che non ha gruppi precedenti, si aggiunge l'ultimo bit dell'ultimo gruppo, e analogamente alla fine dell'ultimo gruppo si aggiunge il primo bit del primo gruppo.

Quest'operazione può essere descritta graficamente tramite una tabella. Innanzitutto, si rappresentano i 32 bit di input, suddivisi in gruppi di 4, come una tabella con 8 righe di 4 celle ciascuna, dove ogni cella indica un bit, numerato da 1 (il bit più a sinistra) a 32 (quello più a destra):

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

I 48 bit di output vengono poi rappresentati aggiungendo una nuova cella all'inizio e una alla fine di ciascuna riga; in queste celle sono scritti gli indici dei bit duplicati che vengono usati per riempire tali celle (qui evidenziati in corsivo):

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

Questa tabella prende il nome di *tabella E*.

3.2 Sostituzione

Dopo l'espansione e lo XOR con la sottochiave, i 48 bit elaborati vengono ritrasformati in 32 bit tramite una funzione di sostituzione, definita da 8 tabelle chiamate **S-box**.

In generale, negli algoritmi crittografici, una S-box è una tabella che definisce una sostituzione di gruppi di m bit con gruppi di n bit (dove m e n possono essere uguali o diversi). Il gruppo di m bit dato in input identifica in qualche modo una cella della tabella, e tale cella contiene un valore di n bit che costituisce l'output della sostituzione. Di conseguenza, il numero m di bit in input determina la dimensione della tabella — essa deve avere 2^m celle — e il numero n di bit in output fissa il range dei valori nelle celle — i 2^n valori da $000\ldots0$ a $111\ldots1$ (cioè, in decimale, da 0 a 2^{n-1}).

Nel caso del DES, ciascuna delle 8 S-box riceve in input uno dei gruppi di 6 bit generati dalla fase di espansione (e modificati dallo XOR con la sottochiave), quindi ha $2^6 = 64$ celle, e produce in output un gruppo di 4 bit, quindi le celle devono contenere i valori da 0 a 15. Le 64 celle sono disposte in 4 righe per 16 colonne, e ciascuna riga contiene una permutazione dei valori da 0 (0000) a 15 (1111). Ad esempio, la S-box S_1, usata per la sostituzione del primo gruppo di 6 bit, è fatta in questo modo:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

(le altre S-box, $S_2, \ldots S_8$, hanno una struttura analoga, ma cambia la permutazione di valori presente in ogni riga).

La regola per la selezione della cella contenente il valore da usare come sostituto è la seguente: dato il gruppo di 6 bit da sostituire, $B = b_1b_2b_3b_4b_5b_6$, il numero della riga è determinato dal valore decimale del primo e dell'ultimo bit del gruppo, $(b_1b_6)_{10}$, mentre il numero della colonna è dato dal valore decimale dei 4 bit centrali, $(b_2b_3b_4b_5)_{10}$. Il valore decimale situato all'intersezione della riga e della colonna considerate viene poi interpretato come un numero binario a 4 bit per ottenere l'output della sostituzione.

Ad esempio, dato in input il gruppo $B = 110010$, si considera l'elemento all'intersezione della riga $(10)_{10} = 2$ e della colonna $(1001)_{10} = 9$, che è 12, ovvero $(12)_2 = 1100$ in binario.

Ricordando che i 6 bit in input provengono da uno dei gruppi prodotti nella fase di espansione E, si osserva che la colonna selezionata dipende sostanzialmente dal corrispondente gruppo di 4 bit originali del blocco R_{i-1}, mentre la riga dipende dai bit duplicati nell'espansione, ovvero dai "bordi" dei gruppi di 4 bit adiacenti in R_{i-1}. Inoltre, la sostituzione dipende anche dalla sottochiave K_i, visto che viene eseguita dopo

che i gruppi di 6 bit sono stati modificati facendo appunto l'operazione di XOR con K_i. Tutti questi fattori contribuiscono a rendere più efficace la sostituzione.

I criteri progettuali delle 8 S-box sono proprio i dettagli del DES che sono stati mantenuti segreti: quando l'algoritmo fu pubblicato, furono elencati i contenuti delle S-box (altrimenti non sarebbe stato possibile implementarlo), ma non fu spiegato come essi fossero stati scelti.

3.3 Permutazione finale

L'ultimo passo del calcolo della funzione F è la permutazione dei 32 bit che costituiscono il risultato. Essa è molto importante: se non ci fosse, a ogni round ciascun gruppo di 4 bit del blocco di dati da cifrare "passerebbe" sempre dalla stessa S-box. Invece, grazie a questa permutazione, complessivamente ogni S-box coinvolge quanti più valori possibili, il che serve a garantire una buona *diffusione* (una delle proprietà definite da Shannon).

4 Key scheduling

Per la generazione delle sottochiavi, i 56 bit della chiave data in input (64 meno gli 8 bit di parità, che vengono scartati) subiscono innanzitutto una permutazione iniziale, dopodiché vengono suddivisi in due metà di 28 bit. Poi, le 16 sottochiavi vengono generate ripetendo iterativamente i seguenti passi:

1. Si fa uno shift circolare a sinistra (ovvero una semplice permutazione) di ciascuna metà della chiave: a seconda del numero del round per cui verrà utilizzata la sottochiave, gli uno o due bit iniziali della metà vengono spostati alla fine.
2. Si usa una tabella per selezionare, dai 56 bit della chiave, i 48 che costituiscono una sottochiave.

La tabella di selezione dei 48 bit è sempre la stessa, ma grazie agli shift i bit selezionati cambiano per ogni round.

5 Decifratura

Siccome DES è basato sulla rete di Feistel, e le permutazioni effettuate sull'input e sull'output della rete sono l'una l'inversa dell'altra, l'algoritmo di decifratura è uguale a quello di cifratura, con la sola differenza che le sottochiavi vengono utilizzate in ordine inverso.

6 Sicurezza

Il DES è tutt'oggi molto resistente agli attacchi di crittoanalisi noti. Infatti, nel corso degli anni sono stati proposti vari attacchi, principalmente di crittoanalisi differenziale e lineare, che sono di tipo known/chosen plaintext; essi non sono facilmente implementabili perché richiedono un numero elevato di testi (ad esempio, come detto prima, esiste un attacco di crittoanalisi differenziale che richiede 2^{47} testi in chiaro scelti; un altro attacco, basato invece sulla crittoanalisi lineare, richiede 2^{43} testi in chiaro conosciuti).

Il punto debole del DES è invece la lunghezza della chiave: con chiavi a 56 bit, ci sono $2^{56} = 7.2 \cdot 10^{16}$ possibili chiavi. Dato che lo spazio delle chiavi è piuttosto piccolo, già alla fine degli anni '90 si riuscì a dimostrare più volte l'esecuzione di un attacco a forza bruta; il più veloce di questi attacchi richiese solo 22 ore di tempo. Perciò, è opportuno utilizzare algoritmi alternativi a DES che hanno chiavi di dimensioni maggiori.

7 Triple DES

Una delle alternative al DES è il **Triple DES** (**TDES**), che utilizza 3 esecuzioni di DES con 3 chiavi diverse, che insieme costituiscono di fatto una singola chiave di $3 \cdot 56 = 168$ bit, ampliando così lo spazio delle chiavi da 2^{56} a 2^{168}.

Esistono varie forme di TDES, che si distinguono per la combinazione di cifrature e decifrature DES applicata. La forma più comunemente impiegata, chiamata **EDE**, esegue la cifratura del testo in chiaro con una chiave, seguita dalla decifratura con una seconda chiave, e infine dalla cifratura con una terza chiave:

$$C = E_{K_1}(D_{K_2}(E_{K_3}(M)))$$

Questa forma non è più sicura della forma *EEE*, che usa semplicemente tre operazioni di cifratura; il vantaggio è invece che un'implementazione di questa forma di TDES può essere usata anche per realizzare il DES "normale": ponendo $K_1 = K_2$ (oppure $K_2 = K_3$), l'operazione di decifratura con K_2 annulla la cifratura con K_1 (o con K_3), lasciando solo l'effetto della singola cifratura con K_3 (o con K_1). Ciò è un vantaggio soprattutto per implementazioni hardware di TDES: così, lo stesso identico circuito può essere usato anche per le applicazioni legacy che usano ancora il DES.

Il principale svantaggio del TDES è che le sue implementazioni software non sono particolarmente efficienti. Ciò ha portato alla definizione di un nuovo standard di cifratura: AES.