Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI–CIVITA"
Corso di Laurea Magistrale in Matematica

Tesi di laurea magistrale

Analisi di sicurezza del cifrario GIFT-COFB attraverso tecniche di crittoanalisi lineare

Candidato:
Simone Pelizzola
Matricola 1207712

Relatore:
Prof. Andrea Lucchini,
Prof. Andrea Visconti (relatore esterno)

Anno Accademico 2021-22
21 Aprile 2022

Analisi di sicurezza del cifrario GIFT-COFB attraverso tecniche di crittoanalisi lineare

Simone Pelizzola

Indice

Introduzione

L'obiettivo principale di questo lavoro di tesi magistrale è quello di comprendere a fondo le idee che stanno alla base della crittoanalisi lineare e di tentare di stimare la robustezza di un cifrario moderno utilizzando questa tecnica.
La crittoanalisi lineare è una tecnica crittoanalitica di tipo chosen ciphertext introdotta nel 1994 da Matsui [1] che si basa su un'analisi statistica dell'evoluzione dei bit dello stato del cifrario considerato, round dopo round. Questo metodo e le sue varianti, ad oggi, sono tra le tecniche più utilizzate per analizzare la sicurezza dei cifrari a blocchi. Di conseguenza in questa tesi verrà preso in esame uno dei cifrari finalisti della gara del NIST bandita per standardizzare l'implementazione della sicurezza su dispositivi con basse capacità di calcolo. Questi dispositivi sono presenti sempre più frequentemente nella vita di tutti i giorni ed è quindi fondamentale effettuare analisi di sicurezza sui cifrari lightweight che verranno utilizzati per proteggerli.
Il cifrario scelto per questo lavoro è GIFT-128 [3,11], un cifrario a blocchi con lunghezza di input, output e chiave di 128 bit ciascuno.
L'analisi di sicurezza contro la crittoanalisi lineare operata dagli autori mostra che 40 round di cifratura sono più che sufficienti per rendere inefficace l'algoritmo di decodifica della chiave presentato da Matsui in [2]. In questo lavoro abbiamo confermato i risultati ottenuti dagli autori in termini di complessità.
Nel capitolo 1 introdurremo le idee principali alla base della crittoanalisi lineare e l'algoritmo di Matsui. Nel capitolo 2 applicheremo questa tecnica a un cifrario esempio di piccole dimensioni. Nel capitolo 3 descriveremo il cifrario DES e la crittoanalisi lineare dello stesso, con numero di round ridotto. Nel capitolo 4, infine, descriveremo l'applicazione della crittoanalisi lineare al cifrario GIFT-128.

Capitolo 1

La crittoanalisi Lineare

1.1 *Notazione*

Nel corso della tesi adotteremo la seguente notazione:

- I = input del cifrario
- S_i = stato del cifrario al round i
- O = output del cifrario
- K = chiave di cifratura
- $X[k]$ = k-esimo bit da sinistra della lista di bit X, la numerazione parte da 0
- $a \bigoplus b$ = XOR tra i bit a e b
- $X \bigoplus Y$ = XOR bit a bit tra le liste di bit X e Y di uguale lunghezza
- $X[k_1, \ldots, k_n] = \bigoplus_{i=1}^{n} X[k_i]$
- $a \cdot b$ = AND tra i bit a e b
- $X \cdot Y = \bigoplus_{i=0}^{n-1} X[i] \cdot Y[i]$, con X e Y liste di bit lunghe n

1.2 *Crittoanalisi lineare: la teoria*

La tecnica crittoanalitica studiata in questo lavoro è stata sviluppata a partire da una particolare debolezza dei cifrari a blocchi.
Questi sono chiamati così perché spezzano il testo da cifrare in blocchi di

lunghezza fissata, li cifrano uno alla volta e si accodano i risultati. In questi cifrari viene definita una funzione di round F e il risultato della cifratura si determina applicando la funzione F ai bit di input un numero fissato r di volte consecutivamente. La Figura 1.1 mostra un esempio di funzionamento di un semplice cifrario a blocchi.

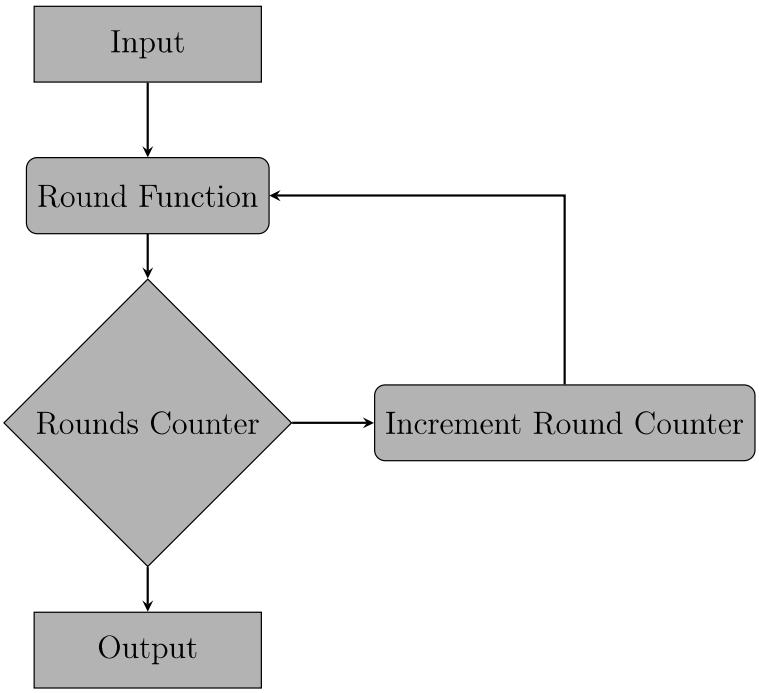

Figura 1.1: Modello di cifrario a blocchi

La funzione F, un cui possibile schema si può vedere nella Figura 1.2, solitamente è composta da alcune componenti. Si trova una componente di permutazione, in cui i bit dello stato vengono permutati attraverso una permutazione fissa. Inoltre c'è una componente di combinazione con la chiave, in cui i bit della chiave di round vengono XORati con lo stato allo step considerato. Infine c'è una componente di sostituzione in cui i bit dello stato, solitamente partizionato in sottoinsiemi disgiunti, vengono modificati utilizzando funzioni che agiscono sui pacchetti di bit.

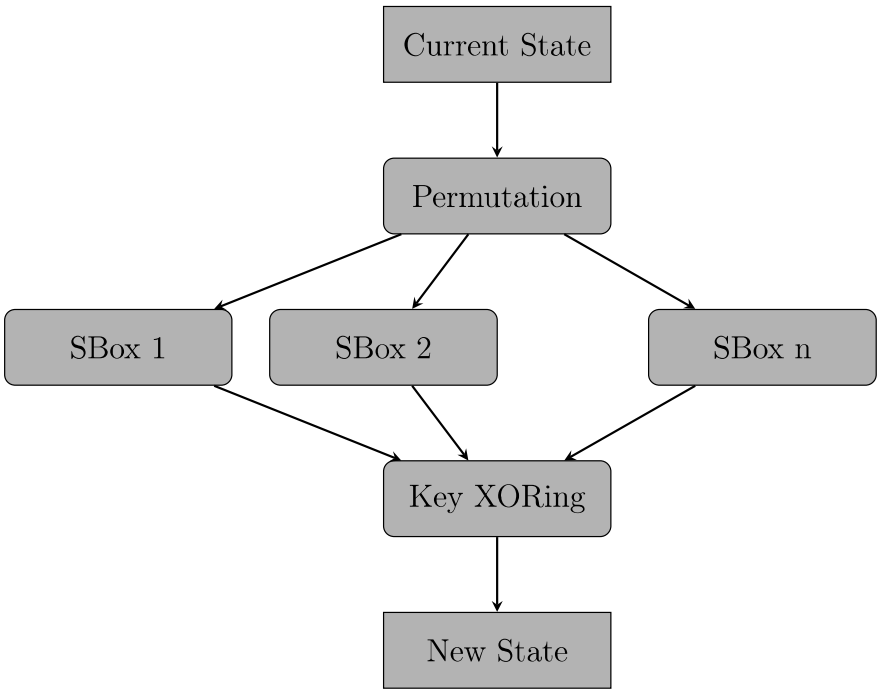

Figura 1.2: Modello di funzione di round

La combinazione di queste funzioni determina la sicurezza del processo. Il punto più controverso della cifratura è la funzione di sostituzione che da un lato garantisce la sicurezza del cifrario, dall'altro, tuttavia, presenta solitamente alcuni punti deboli, uno dei quali è alla base della crittoanalisi lineare. Senza questa fase, infatti, il risultato della cifratura non sarebbe altro che una lista di combinazioni lineari tra i bit della chiave e alcune costanti, il che consentirebbe di risalire alla chiave stessa con meno complessità computazionale rispetto a una banale ricerca esaustiva. Sarebbe sufficiente, in tal senso, operare una ricerca esaustiva su una porzione della chiave e risalire al resto a partire da queste combinazioni lineari.
D'altra parte questa funzione di sostituzione presenta delle debolezze. Quella utilizzata per la crittoanalisi lineare, in particolare, é dovuta al fatto che lo XOR tra i bit in determinate posizioni dell'input della funzione di sostituzione è uguale allo XOR dei bit in determinate posizioni (non necessariamente le stesse di prima) dell'output con probabilità diversa da $\frac{1}{2}$.
Sia SB la funzione di sostituzione (substitution box) e supponiamo che trasformi insiemi di n bit in insiemi di m bit, dunque

$$SB : \{0,1\}^n \rightarrow \{0,1\}^m$$

In presenza della debolezza descritta poco fa si può dedurre una relazione lineare tra chiave, input e output del round in esame come se non ci fosse la funzione di sotituzione, tenendo presente che questa sarà verificata con una probabilità diversa da 1. Siano infatti $x_1, \ldots, x_n$ le posizioni dei bit di output di SB. Questi verranno poi permutati dallo strato di permutazione diventando $o_1, \ldots, o_n$ sullo stato completo e infine sommati ai bit di chive in posizione $k_1, \ldots, k_n$. Siano inoltre $i_1, \ldots, i_n$ le posizioni dei bit di input di SB considerati, sullo stato completo.
Allora sapremo che

$$I[i_1, \ldots, i_n] \bigoplus O[o_1, \ldots, o_n] = K[k_1, \ldots K_n] \tag{1.1}$$

vale con probabilità $p \neq \frac{1}{2}$, su un singolo round.
Conoscendo un numero adeguato di copppie $\{I, O\}$ cifrate con la stessa chiave si può quindi calcolare il termine sinistro dell'equazione per ogni coppia. Poiché il termine destro è una costante ci si aspetta che il termine destro dia come risultato il valore di tale costante una porzione molto vicina a p del totale delle coppie considerate. Se il risultato è 0 per tale numero di volte allora anche il termine destro di (1.1) sarà 0, se no sarà 1.
Applicando questa operazione con più relazioni diverse si risale a un numero di bit della chiave sufficiente a poter provare i bit rimanenti con una ricerca esaustiva senza eccessivo tempo di calcolo. La parte più complessa di questo approccio da un punto di vista teorico è la determinazione delle relazioni tipo (1.1) sul cifrario completo. La'efficacia di questa tecnica è quindi determinata dalla complessità della ricerca delle equazioni lineari tra input, output e chiave e dall'analisi statistica successiva delle coppie di input e output.

1.3 *Crittoanalisi lineare: la pratica*

Per determinare le espressioni lineari tipo la (1.1) si parte dall'idea presentata nella sezione precedente.
Supponiamo di avere una relazione

$$S_{l-1}[i_1, \ldots, i_n] \bigoplus S_l[o_1, \ldots, o_m] = K[k_1, \ldots k_l]$$

su un round e

$$S_l[j_1, \ldots, j_r] \bigoplus S_{l+1}[q_1, \ldots, q_s] = K[x_1, \ldots x_t]$$

sul successivo. Se $[o_1, \ldots, o_m] = [j_1, \ldots, j_r]$ si possono XORare le due equazioni ottenendo

$$S_{l-1}[i_1, \ldots, i_n] \bigoplus S_{l+1}[q_1, \ldots, q_s] = K[k_1, \ldots k_l, x_1, \ldots, x_t]$$

su due round. Se la prima espressione era soddisfatta con probabilità p_1 e la seconda con probabilità p_2 allora il loro XOR sarà soddisfatto se e solo se sono entrambe soddisfatte o non lo è nessuna delle due, quindi con probabilità $p = p_1p_2 + (1 - p_1)(1 - p_2)$.

Proposizione 1.3.1. *$p = p_1p_2 + (1 - p_1)(1 - p_2) = \frac{1}{2}$ se e solo se $p_1 = \frac{1}{2}$ o $p_2 = \frac{1}{2}$.*

Dimostrazione. Sviluppando

$$p_1p_2 + (1 - p_1)(1 - p_2) = 2p_1p_2 - p_1 - p_2 + 1 = 2(p_1 - \frac{1}{2})(p_2 - \frac{1}{2}) + \frac{1}{2}$$

che vale $\frac{1}{2}$ se e solo se $2(p_1 - \frac{1}{2})(p_2 - \frac{1}{2}) = 0$, dunque se e solo se $p_1 = \frac{1}{2}$ o $p_2 = \frac{1}{2}$ □

Dunque se si riescono a determinare delle espressioni lineari tipo la (1.1) tutte con probabilità diversa da $\frac{1}{2}$, una per ogni round, tali che i bit di output a un round coincidano con quelli di input al round successivo, lo XOR di esse fornirà una relazione tra I, O, K sul cifrario completo, con I e O noti, dunque con la possibiltà di applicare la tecnica descritta.
Per il calcolo della probabilità complessiva ci serviamo di un'estensione del risultato appena dimostrato

Lemma 1.3.1 (Piling-up lemma)**.** *Siano $E_1, \ldots, E_n$, $n \geq 2$, espressioni lineari tipo la (1.1) soddisfatte con probabilità rispettivamente $p_1, \ldots, p_n$. Allora $\bigoplus_{i=1}^{n} E_i$ sarà soddisfatta con probabilità $p = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} (p_i - \frac{1}{2})$.*

Dimostrazione. Lavoriamo per induzione su n.
Il caso $n = 2$ l'abbiamo mostrato in precedenza.
Supponiamo ora che la proprietà sia valida fino a $n - 1$.

$$\bigoplus_{i=1}^{n} E_i = (\bigoplus_{i=1}^{n-1} E_i) \bigoplus E_n$$

Se la parte in parentesi vale con probabilità p_0 allora $p = \frac{1}{2} + 2(p_0 - \frac{1}{2})(p_n - \frac{1}{2})$

$$p_0 = \frac{1}{2} + 2^{n-2} \prod_{i=1}^{n} (p_i - \frac{1}{2})$$

Dunque $p = \frac{1}{2} + 2 \cdot 2^{n-2} \prod_{i=1}^{n} (p_i - \frac{1}{2}) \cdot (p_n - \frac{1}{2}) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} (p_i - \frac{1}{2})$ □

Una volta determinata l'espressione finale si procede con l'algoritmo di Matsui

1. Sia N il numero di coppie I, O considerate
2. Sia T il numero di coppie tali per cui il membro sinistro della (1.1) vale 0
3. Sia p la probabilità che la (1.1) sia valida
4. Se $p > \frac{1}{2}$ e $T > \frac{N}{2}$ si pone il membro destro uguale a 0
5. Se $p > \frac{1}{2}$ e $T < \frac{N}{2}$ si pone il membro destro uguale a 1
6. Se $p < \frac{1}{2}$ e $T > \frac{N}{2}$ si pone il membro destro uguale a 1
7. Se $p < \frac{1}{2}$ e $T < \frac{N}{2}$ si pone il membro destro uguale a 0

Vale la seguente

Proposizione 1.3.2. *L'efficacia dell'algoritmo appena esposto è pari a*

$$\int_{-2\sqrt{N}|p-\frac{1}{2}|}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

In particolare per $N = 2|p - \frac{1}{2}|^{-2}$ *si ottiene un'efficacia del* 99.8%.

Dimostrazione. Consideriamo N variabili aleatorie di Bernoulli equidistribuite $X_1, \ldots, X_N$, dove ogni X_i vale 1 se l'equazione (1.1) è verificata per la coppia i-esima di input-output e 0 altrimenti. Vale $P(X_i = 1) = p \ \forall 1 \leq i \leq N$. La variabile aleatoria $S_N = \sum_{i=1}^{N} X_i$ ha una distribuzione binomiale, dunque per N molto grande S_N tende a una normale di media pN e varianza $p(1-p)N$.
Supponiamo ora $p > \frac{1}{2}$.
L'algoritmo ha successo se $S_N > \frac{N}{2}$, dunque la probabilità di successo dell'algoritmo è

$$\int_{\frac{N}{2}}^{+\infty} \frac{1}{\sqrt{2\pi p(1-p)N}} e^{-\frac{(x-pN)^2}{2(p(1-p))^2}} dx$$

Standardizzando la variabile questo diventa

$$\int_{\frac{\frac{N}{2}-pN}{\sqrt{p(1-p)N}}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Essendo $p(1-p) < \frac{1}{4}$ per ogni $0 \leq p \leq 1$ e $p > \frac{1}{2}$ abbiamo

$$\frac{\frac{N}{2} - pN}{\sqrt{p(1-p)N}} < -2\sqrt{N}(p - \frac{1}{2})$$

Dunque la probabilità di successo sarà maggiore di

$$\int_{-2\sqrt{N}(p-\frac{1}{2})}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Nel caso $p < \frac{1}{2}$ la probabilità di successo è

$$\int_{-\infty}^{\frac{\frac{N}{2}-pN}{\sqrt{p(1-p)N}}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Poiché l'argomento è una funzione pari questo equivale a

$$\int_{-\frac{\frac{N}{2}-pN}{\sqrt{p(1-p)N}}}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Applicando un ragionamento analogo a quello fatto in precedenza (tenendo presente questa volta che $p < \frac{1}{2}$ e che c'è un segno $-$ davanti all'estremo inferiore) otteniamo la stima inferiore

$$\int_{-2\sqrt{N}(p-\frac{1}{2})}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

come in precedenza. In conclusione la probabilità di successo dell'algoritmo sarà sempre maggiore di

$$\int_{-2\sqrt{N}|p-\frac{1}{2}|}^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Inoltre questa stima viene trattata come valore esatto poiché il valore di p è solitamente molto vicino a $\frac{1}{2}$, di conseguenza la maggiorazione è molto bassa. Nel caso del cifrario DES, ad esempio, considerando solo 8 round di cifratura il valore di p migliore possibile si discosta da $\frac{1}{2}$ di circa 10^{-3}. □

Bisogna ora stabilire un metodo per determinare le espressioni lineari utili sui singoli round.
Sia X l'input della funzione SB e Y il relativo output. Siano inoltre α e β due liste di bit lunghe quanto X e Y, diciamo lunghe n e m. Queste liste, che chiameremo maschere, determinano quali bit di X e Y prendere (quelli in corrispondenza degli 1 delle maschere) per calcolarne lo XOR. Infatti un'equazione lineare generica tra alcuni bit di input e alcuni bit di output dell'SBox si può rappresentare come $\alpha \cdot X \bigoplus \beta \cdot Y = 0$. Questa espressione corrisponde a fare lo XOR tra i bit di input nelle posizioni corrispondenti

ai bit 1 di α e i bit di output in corrispondenza dei bit 1 di β per come è definito il prodotto tra sequenze di bit. Dunque cercheremo quete maschere tali che $X \cdot \alpha$ e $Y \cdot \beta$ siano uguali molto spesso o molto raramente.
Definiamo quindi

$$NS(\alpha, \beta) = |\{X \in \{0,1\}^n \ t.c. \ X \cdot \alpha = SB(X) \cdot \beta\}|$$

Se $NS(\alpha, \beta) \neq 2^{n-1}$ allora la probabilità della relazione sulla Substitution Box considerata (SBox) è diversa da $\frac{1}{2}$. Poiché quest SBox sono diverse tra loro e prendono porzioni diverse di stato del cifrario indicheremo con NS_i la funzione NS relativa all'i-esima SBox. A partire dalle relazioni trovate sulle SBox si risale alle posizioni nello stato completo dei bit coinvolti e si mettono insieme i bit per ottenere la relazione.
Resta ora da determinare un metodo per trovare le espressioni lineari migliori tra I e O a partire da quelle sulle singole SBox su un round. Dunque bisogna trovare delle espressioni lineari su ogni round in modo tale che l'input a un round coincida con l'output al precedente, determinando così un path di espressioni lineari che conducono a un'espressione finale che coinvolge solo I e O.
Per fare questo Matsui ha proposto una tecnica in [2] basata su una rappresentazione del sistema sotto forma di grafo.
Si consideri un grafo orientato i cui vertici sono le terne (B_I, B_O, r), dove B_I è un sottoinsieme di S_r, B_O un sottoinsieme di S_{r+1} e r è il round corrente. Due vertici (B_{I1}, B_{O1}, r_1) e (B_{I2}, B_{O2}, r_2) sono collegati se $r_1 = r_2 - 1$, $B_{O1} = B_{I2}$ e esistono espressioni lineare tipo (1.1) con i bit di input e output dati da (B_{I1}, B_{O1}) e (B_{I2}, B_{O2}) tali che la probabilità p dell'equazione risultante dalla loro somma sia diverse da $\frac{1}{2}$. Il peso dell'arco sarà dato dal massimo possibile di $|p - \frac{1}{2}|$.
Cercare l'espressione migliore equivale quindi a cercare il cammino di peso massimo che sia lungo quanto il numero di round. Per fare questo si utilizza un algoritmo tipo Djikstra.
Si parte da una stima per difetto dell'efficienza finale ad ogni numero di round tra 1 e il massimo, chiamiamola B_i su i round.

- Passo1: determino il punto che da l'equazione su un round che mi dia la migliore probabilità possibile. Controllo se l'efficienza ottenuta considerando il nuovo primo elemento e B_{r-1} è meglio di B_r. Se sì aggiorno B_r e procedo al Passo 2, se no mi fermo.

- Passo 2: determino il punto che da l'equazione su un round che mi dia la migliore probabilità possibile tra i punti collegati al primo nel percorso. Controllo se l'efficienza ottenuta considerando il nuovo secondo

elemento e B_{r-2} è meglio di B_r. Se sì aggiorno B_r eB_{r-1} e torno al Passo 1, se no procedo al passo j=3.

- Passo j: determino il punto che da l'equazione su un round che mi dia la migliore probabilità possibile tra i punti collegati al (j-1)-esimo nel percorso. Controllo se l'efficienza ottenuta considerando il nuovo j-esimo elemento e B_{r-j} è meglio di B_r. Se sì aggiorno B_r, B_{r-1},... , B_{r-j} e torno al Passo j-1, se no procedo al passo j+1.

Nel prossimo capitolo introdurremo il concetto di correlazione e la teoria matematica che giustifica il calcolo di queste espressioni ed enunceremo un teorema che fornisce un metodo algebrico per determinare l'efficacia della crittoanalisi lineare contro un cifrario.

1.4 *Crittoanalisi differenziale*

Concludiamo questo capitolo con una breve digressione su un'altra tecnica, la crittoanalisi differenziale, che è strettamente legato a quella lineare [14]. Questa tecnica è stata introdotta da Biham e Shamir nel 1993 [4,12] e studia come si evolvono le differenze tra due diversi input del cifrario round dopo round.
Siano I_1 e I_2 due input distinti del cifrario e sia $\Delta I = I_1 \bigoplus I_2$. Sia inoltre F la funzione di cifratura. Scelta una differenza di output ΔO si può determinare, al variare di I tra i possibili input, la probabilità che

$$F(I,K) \bigoplus F(I \bigoplus \Delta I, K) = \Delta O$$

Una volta determinata una coppia di differenze su un round si può proseguire al round successivo con una nuova coppia tale che la differenza di input della seconda sia uguale alla differenza di output della prima e moltiplicarne le probabilità.
Se queste probabilità sono particolarmente alte allora si possono ottenere informazioni sulla chiave guardando le differenze sul cifrario completo.
Un'analogia con la crittoanalisi lineare è molto evidente, infatti la ricerca della successione ottimale di differenziali può essere effettuata nello stesso modo in cui si cercavano le espressioni lineari migliori.
Queste tecniche si sono evolute in concomitanza a causa della loro affinità e ne sono state addirittura sviluppate altre che ne combinano le prestazioni [6].

Capitolo 2

La Teoria della Correlazione

In questo capitolo presentiamo la teoria della correlazione sviluppata e applicata alla crittoanalisi lineare in [7,10,13].
La teoria che andremo a spiegare culmina con la descrizione di un oggetto che fornisce un metodo per determinare l'efficacia della crittoanalisi lineare di un cifrario.

2.1 *La Correlazione*

Siano n e m due numeri interi positivo, una funzione booleana

$$f : \{0,1\}^n \longrightarrow \{0,1\}^m$$

Se $m = 1$ la funzione booleana si dice binaria

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

Definizione 2.1.1 (Correlazione)**.** *Date due funzioni booleane binarie*

$$f, g : \{0,1\}^n \longrightarrow \{0,1\}$$

si definisce correlazione tra f e g la funzione $C(f,g) = 2P(f = g) - 1$ della probabilità che $f(x) = g(x)$.

Chiaramente la correlazione è una funzione simmetrica nelle due variabili.
Se $C(f,g) \neq 0$ le due funzioni si dicono correlate. Inoltre $C(f,g) = 1 \iff f \equiv g$ e $C(f,g) = -1 \iff f \equiv 1 - g$.
Definiamo inoltre una particolare classe di funzioni, le parità.

Definizione 2.1.2 (Parità)**.** *Una funzione booleana binaria α si definisce una parità se $\alpha(x_1, x_2, \ldots, x_n) = x_{i_1} \bigoplus x_{i_2} \bigoplus \ldots \bigoplus x_{i_m}$.*

Quindi una funzione di parità calcolata in una sequenza di bit restituisce la parità del numero di bit valorizzati a 1 tra un sottoinsieme fissato della sequenza. Alternativamente si può vedere come il prodotto scalare tra due vettori di $(\mathbb{Z}/2\mathbb{Z})^n$, dove il vettore corrispondente alla funzione di parità ha valore 1 in tutte e sole le posizioni corrispondenti al sottoinsieme di bit che considera. Dunque detto α questo vettore si ha $\alpha(x) = \alpha^T \cdot x$.
Introduciamo ora la

Definizione 2.1.3 (Controparte Reale di una funzione booleana)**.** *Data una funzione booleana binaria f si definisce la sua controparte reale la funzione $\hat{f} : \{0,1\}^n \longrightarrow \mathbb{R}$, tale che $\hat{f}(x) = (-1)^{f(x)}$, dunque vale -1 quando f vale 1 e 1 quando f vale 0.*

Si osserva immediatamente che $f \hat{\bigoplus} g = \hat{f} \cdot \hat{g}$.
Questa proprietà ci consente di rappresentare l'insieme delle funzioni booleane binarie con dominio $\{0,1\}^n$ come un sottoinsieme di un $\mathbb{R}$-spazio vettoriale di dimensione 2^n. Il vettore dello spazio corrispondente alla funzione booleana f è $(\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_{2^n}))$, dove $x_1, x_2, \ldots, x_{2^n}$ sono gli elementi di $\{0,1\}^n$ in un qualche ordine fissato e avrà quindi solo -1 e 1 come elementi.
Il prodotto per scalari è il prodotto usuale, la somma vettoriale è la somma usuale, componente per componente e il prodotto scalare tra vettori è il prodotto scalare usuale, che denoteremo con $< \cdot, \cdot >$.
Consideriamo in questo insieme i vettori $\overline{f}, \overline{g}$ corrispondenti a funzioni booleane f, g. Il loro prodotto scalare $< \overline{f}, \overline{g} >$ è la somma di 2^n valori uguali a -1 o 1. In particolare, se $f(x) = g(x)$ per k valori di x allora $< \overline{f}, \overline{g} >= x - (2^n - x) = 2x - 2^n$. Si osservi, inoltre, che la correlazione $C(f,g) = 2\frac{x}{2^n} - 1 = \frac{2x-2^n}{2^n}$. Di conseguenza

$$C(f,g) = \frac{< \overline{f}, \overline{g} >}{||\overline{f}|| \cdot ||\overline{g}||}$$

infatti

$$||\overline{f}|| = \sqrt{< \overline{f}, \overline{f} >} = \sqrt{\sum_{i=1}^{2^n} 1} = 2^{\frac{n}{2}}$$

La correlazione tra due funzioni booleane binarie è quindi data dal coseno dell'angolo tra i due vettori corrispondenti alle funzioni nello spazio vettoriale introdotto poco fa.

Una base dello spazio vettoriale è data dai vettori corrispondenti alle funzioni di parità. Infatti date due diverse funzioni di parità α e β è immediato verificare che $< \overline{\alpha}, \overline{\beta} >= 0$ e che quindi le funzioni di parità formano una base ortogonale dello spazio vettoriale che chiameremo base di Walsh-Hadamard. Chiaramente il coefficienti di $\overline{f}$ relativo alla parità α nella base appena definita sarà dato da $\frac{<\overline{f},\overline{\alpha}>}{2^n}$, che per quanto mostrato in precedenza sono uguali a $C(f, \alpha)$.
Questa rappresentazione è detta spettro di Walsh-Hadamard e data una parità α_i definiamo la

Definizione 2.1.4 (Trasformata di Walsh-Hadamard)**.** *La trasformata di Walsh-Hadamard $A_i : \mathbb{R}^{2^n} \longrightarrow \mathbb{R}$ relativa alla funzione di parità α_i è la funzione che associa al vettore $\overline{f}$ la norma della sua proiezione sul sottospazio generato da $\overline{\alpha_i}$, cioè il coefficiente di $\overline{\alpha_i}$ nella decomposizione in base di Walsh-Hadamard.*

In particolare $A_i(f) = C(f, \alpha_i)$.
Di conseguenza $\forall x \in \{0,1\}^n$, $f : \{0,1\}^n \longrightarrow \{0,1\}$ booleana avremo

$$\hat{f}(x) = (-1)^{f(x)} = \sum_i A_i(f)\hat{\alpha_i}(x) = \sum_i A_i(f) \cdot (-1)^{\alpha^T \cdot x}$$

2.2 *La trasformata di Walsh-Hadamard*

Mostriamo ora alcune semplici proprietà della trasformata di Walsh-Hadamard.

Proposizione 2.2.1. *Per ogni funzione booleana binaria f vale*

$$\sum_i A_i^2(f) = 1$$

Dimostrazione.

$$2^n =< \hat{f}, \hat{f} >=< (\sum_i A_i(f)\hat{\alpha_i}), (\sum_j A_j(f)\hat{\alpha_j}) >= \sum_i \sum_j A_i(f)A_j(f) < \hat{\alpha_i}, \hat{\alpha_j} >$$

Come detto prima $< \hat{\alpha_i}, \hat{\alpha_j} >= 0$ se $i \neq j$, vale 2^n altrimenti, dunque

$$2^n = \sum_i \sum_j A_i(f)A_j(f) \cdot 2^n \delta_{i=j} = 2^n \sum_i A_i^2(f)$$

□

Proposizione 2.2.2. *Per ogni coppia di funzioni booleane binarie f, g vale*

$$A_i(f \bigoplus g) = A_i(f) \bigotimes A_i(g)$$

dove con $\bigotimes$ si intende il prodotto di convoluzione.

Dimostrazione.

$$f \hat{\bigoplus} g = \hat{f} \cdot \hat{g} = (\sum_i A_i(f)\hat{\alpha_i}) \cdot (\sum_j A_j(g)\hat{\alpha_j}) = \sum_i \sum_j A_i(f)A_j(g)\hat{\alpha_i}\hat{\alpha_j}$$

Abbiamo che $\hat{\alpha_i}\hat{\alpha_j} = \alpha_i \hat{\bigoplus} \alpha_j$ e i vari $\alpha_i \bigoplus \alpha_j$ danno come risultato gli stessi valori di α_k permutati e ripetuti 2^n volte. Quindi possiamo riscrivere quella somma isolando $\alpha_k = \alpha_i \bigoplus \alpha_j$ come

$$f \hat{\bigoplus} g = \sum_k (\sum_j A_i(f)A_j(g))\hat{\alpha_k}$$

dove i è definito dalla relazione precedente.
In conclusione $A_k(f \bigoplus g) = \sum_j A_i(f)A_j(g)$ dove, ponendo $F(\alpha_i) = A_i(f)$, $G(\alpha_j) = A_j(g)$ e $H(\alpha_k) = A_k(h)$, con $h = f \bigoplus g$, avremo $A_i(f) = F(\alpha_j \bigoplus \alpha_k)$, dunque $H(\alpha_k) = \sum_j F(\alpha_k \bigoplus \alpha_j)G(\alpha_j)$, il prodotto di convoluzione. □

Da questa proposizione si evince facilmente che $A_i(f \bigoplus 1) = -A_i(f)$ e $A_i(f \bigoplus \alpha_j) = A_k(f)$, con $\alpha_k = \alpha_i \bigoplus \alpha_j$. Usando la notazione della fine della dimostrazione, se $g = f \bigoplus \alpha_j$ abbiamo $G(\alpha_i) = F(\alpha_i \bigoplus \alpha_j) = F(\alpha_k)$.
Prima di enunciare un ultimo corollario della formula di convoluzione definiamo il supporto di una funzione booleana f come il sottospazio vettoriale $supp(f)$ generato dalle funzioni di parità α tali che $< \hat{f}, \hat{\alpha} > \neq 0 \iff A(f) = F(\alpha) \neq 0$. Siamo ora pronti per il seguente

Corollario 2.2.0.1. *Siano f e g funzioni booleane binarie con supporto disgiunto. Allora un vettore $v \in supp(f \bigoplus g)$ si scrive in modo unico come somma di due vettori $u \in supp(f)$ e $w \in supp(g)$. Inoltre posta $h = f \bigoplus g$ vale $H(v) = F(u)G(w)$* In generale $supp(f \bigoplus g) \in supp(f) + supp(g)$.

2.3 *Le Matrici di Correlazione*

I cifrari che considereremo e le loro componenti sono funzioni booleane non binarie, solitamente con $m = n$ ma non necessariamente, quindi per applicare la teoria sviluppata finora le decomporremo in m componenti binarie. Data

dunque una funzione booleana $h : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ in questa sezione la considereremo come un vettore $(h_1, h_2, \ldots, h_m)$, dove ogni h_i è una funzione booleana binaria $h_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$.
Ognuna di queste componenti avrà la sua funzione spettro H_i, dunque lo spettro di h può essere rappresentato come un vettore $(H_1, H_2, \ldots, H_m$. Inoltre scelta una funzione di parità β di dimensione m possiamo calcolare lo spettro di $\beta^T h$ che sarà la convoluzione degli H_i tali che la i-esima componente di β vale 1.
Essendoci 2^m possibili valori di β, ognuno dei quali genera uno spettro di dimensione 2^n, possiamo compilare una matrice $2^m \times 2^n$ le cui righe sono gli spettri appena definiti, dunque le entrate della matrice non saranno altro che le correlazioni tra funzioni di parità dell'input di h e funzioni di parità dell'output di h.
Questa matrice $C^{(h)}$ viene detta **Matrice di Correlazione** di h.
Si ha $C^{(h)}_{i,j} = C(\beta_i^T h, \alpha_j^T)$.

Proposizione 2.3.1. *L'insieme delle funzioni booleane $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ è in biiezione con l'insieme delle matrici di correlazione $M(2^m, 2^n)$.*

La dimostrazione della proposizione è immediata definendo l'operatore L_n che manda vettori binari di dimensione n in funzioni a valori in $\mathbb{R}^{2^n}$ tale che $L_n(\alpha) = \hat{\alpha}$ sia il vettore della base di Walsh-Hadamard corrispondente ad α visto come funzione di parità, dunque $L_n(\alpha)(x) = \hat{\alpha}(x) = (-1)^{\alpha^T x}$.
Allora posto $\beta = h(\alpha)$ avremo che la colonna della matriche di correlazione corrispondente ad α non è altro che β e dunque $C^{(h)}\hat{\alpha} = \hat{\beta} = L_m(\beta) = L_m(h(\alpha))$.
Quindi ad ogni h corrisponde una matrice di correlazione $C^{(h)}$ e viceversa.
Vediamo infine un'ultima proprietà delle matrici di correlazione fondamentale per il proseguimento dello studio della crittoanalisi lineare. Poiché l'approssimazione lineare del cifrario viene fatta round per round avremo bisogno delle correlazioni relative alla composizione di diverse funzioni booleane, infatti ad ogni round avremo una o più funzioni differenti. Procediamo quindi ad enunciare la seguente

Proposizione 2.3.2. *Date due funzioni booleane $h_1 : F_2^n \longrightarrow F_2^p, h_2 : F_2^p \longrightarrow F_2^m$ e definita la funzione booleana $h = h_2(h_1) : F_2^n \longrightarrow F_2^m$ vale la seguente relazione tra le loro matrici di correlazione*

$$C^{(h)} = C^{(h_2)} \times C^{(h_1)}$$

Dimostrazione. Per quanto detto prima

$$C^{(h)}\hat{\alpha} = \hat{h(\alpha)}$$

Dunque

$$C^{(h)}\hat{\alpha} = \hat{h(\alpha)} = h_2(\hat{h_1}(\alpha)) = C^{(h_2)}\hat{h_1(\alpha)} = C^{(h_2)}C^{(h_1)}\hat{\alpha}$$

□

2.4 *La Correlazione e la crittoanalisi Lineare*

Consideriamo ora un cifrario che opera su r round, la funzione di cifratura complessiva è quindi una funzione booleana $f = f_r \circ f_{r-1} \circ \ldots \circ f_1$, con $f_1, f_2, \ldots, f_r$ funzioni booleane. La matrice di correlazione di f è dunque $C^{(f)} = C^{(f_r)} \times \ldots \times C^{(f_1)}$.

Definizione 2.4.1 (Trail Lineare)**.** *Un trail lineare U su una funzione iterativa booleana a r round è una sequenza di $r+1$ funzioni di parità $(u_0, u_1, \ldots, u_r$ definita da r passi (u_{i-1}, u_i) con correlazione $C(u_i^T f_i, u_{i-1}^T)$.*

Il contributo di un trail lineare U alla correlazione complessiva è $C(U) = \prod_i C(u_i^T f_i, u_{i-1}^T)$. Infatti la rappresentazione della matrice di f come prodotto implica

Teorema 2.4.1. *La correlazione tra una parità di input α^T e una parità di output $\beta^T f$ è data dalla somma dei contributi alla correlazione di tutti i possibili trail lineari che iniziano da α e finiscono in β.*

$$C(\beta^T f, \alpha^T) = \sum_{U | u_0 = \alpha, u_r = \beta} C(U)$$

Introduciamo infine il concetto di potenziale di correlazione.

Definizione 2.4.2. *Il potenziale di correlazione medio tra un input α e un output β è dato da*

$$\sum_i C(U_i)^2$$

dove gli U_i sono tutti i trail che iniziano con l'input e finiscono con l'output dati.

Abbiamo mostrato in precedenza che la somma sugli input di tutti i potenziali medi di correlazione deve fare 1. Perciò il potenziale di correlazione medio di una coppia input/output dovrebbe valere in media 2^{-m}. Consideriamo quindi una coppia di input/output indicativa della robustezza del cifrario se ha un potenziale di correlazione superiore a 2^{-m}.
All'inizio del capitolo abbiamo mostrato anche come la correlazione sia $2P(\alpha^T x = \beta^T f(x)) - 1 = 2(P(\alpha^T x = \beta^T f(x)) - \frac{1}{2})$. Quindi il contributo potenziale di correlazione di un trail corrispondente a un'equazione lineare che ha probabilità $\frac{1}{2} + \epsilon$ è $4\epsilon^2$.
Nell'ultimo capitolo vedremo che questo ci darà indicazioni sulla robustezza del cifrario.

Capitolo 3

Un esempio giocattolo: DES ridotto

3.1 *Il cifrario*

Consideriamo il cifrario DES ridotto presentato in [8], che ha una costruzione simile a DES ma lavora su blocchi di 12 bit con una chiave lunga 9 bit.
Definiamo la funzione di round.
Al round i si considerano le metà sinistra e destra del testo, L_i e R_i. La funzione di round f agisce su R_i per ottenere R_{i+1}, mentre L_{i+1} si ricava dallo stato precedente:
$L_{i+1} = R_i$.
$R_{i+1} = f(R_i, K_i) \bigoplus L_i$
dove K_i è l'i-esima chiave di round, ottenuta considerando 8 bit consecutivi della chiave originaria partendo dall'i-esimo.
$K_i = (K[i \pmod 9], K[i+1 \pmod 9], \ldots, K[i+7 \pmod 9]$
f è la composizione di un'espansione, una somma con la chiave e una sostituzione di bit:

1. L'espansione è una funzione $E : \{0,1\}^6 \longrightarrow \{0,1\}^8$ tale che $E(R_i) = E(b_1b_2b_3b_4b_5b_6) = b_1b_2b_4b_3b_4b_3b_5b_6$.

2. La somma con la chiave è il semplice XOR dello stato con la chiave di round K_i.

3. La funzione di sostituzione viene descritta attraverso la classica rappresentazione tramite S-Box.

Le S-Box sono due tabelle costituite da 2 righe, tali che ogni riga è una permutazione di $\{0, \ldots, 7\}$, dunque sono tabelle da 2 righe e 8 colonne. La funzione

di sostituzione è una funzione $SB : \{0,1\}^4 \times \{0,1\}^4 \longrightarrow \{0,1\}^3 \times \{0,1\}^3$, $SB = (SB_1, SB_2)$, dove le S-Box rappresentano funzioni $SB_j : \{0,1\}^4 \longrightarrow \{0,1\}^3$. L'input della funzione SB, una sequenza di 8 bit, viene divisa nelle due metà destra e sinistra, ognuna delle quali verrà trasformata da una S-Box. I due risultati, quindi, vengono accodati, ottenendo un output di 6 bit.
Descriviamo ora il funzionamento delle S-Box.
L'input di una S-Box è del tipo $b_1b_2b_3b_4$.
Il bit b_1 indica la riga della tabella da considerare.
$b_2b_3b_4$, convertito in decimale, indica la colonna.
Si tenga presente che la numerazione di righe e colonne inizia sempre da 0.
Dunque, se l'input è 1010, l'output sarà l'elemento nella seconda riga (riga 1) e terza colonna (colonna 010=2).
Nel caso del nostro cifrario le S-Box sono:

$$SB_1 = \begin{bmatrix} 101 & 010 & 001 & 110 & 011 & 100 & 111 & 000 \\ 001 & 100 & 110 & 010 & 000 & 111 & 101 & 011 \end{bmatrix}$$

$$SB_2 = \begin{bmatrix} 100 & 000 & 110 & 101 & 111 & 001 & 011 & 010 \\ 101 & 011 & 000 & 111 & 110 & 001 & 010 & 100 \end{bmatrix}$$

Si possono vedere rappresentazioni grafiche del cifrario e della funzione di round rispettivamente in Figura 3.1 e in Figura 3.2.

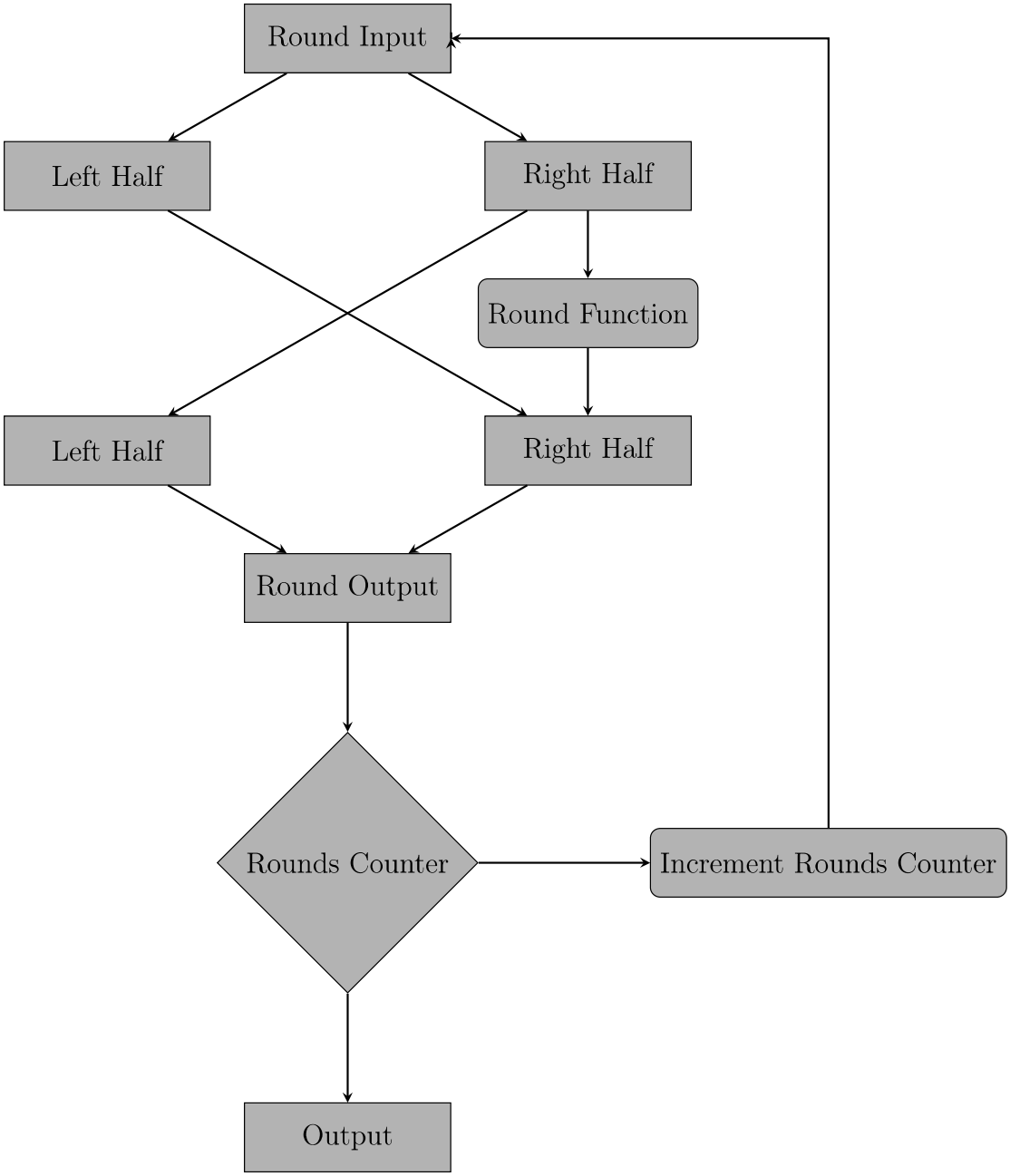

Figura 3.1: Schema di funzionamento di DES ridotto

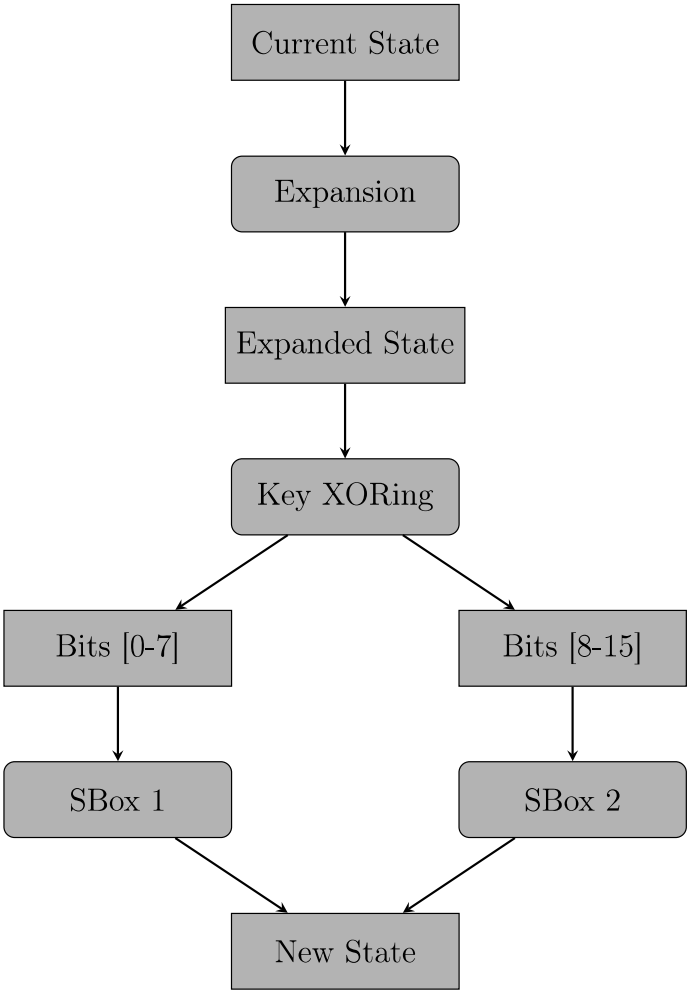

Figura 3.2: Funzione di round di DES ridotto

Cifriamo ad esempio su un round 010011101001 con la chiave 111000101.
$L_1 = 010011$.
$R_1 = 101001$.
$K_1 = 11100010$.
$L_2 = R_1 = 101001$.
L'espansione di R_1 è $E(R_1) = 10010101$.
$E(R_1) \bigoplus K_1 = 01110111$.
Per calcolare la S-Box consideriamo la prima metà del risultato, 0111. Dovremo prendere quindi dalla prima S-Box l'elemento nella prima riga e ultima colonna, cioè 000. Per la seconda metà, essendo uguale alla prima, è lo stesso, ma con la seconda tabella, quindi 010. L'output delle S-Box sarà quindi

000010
Lo XOR con L_1 è 010001.
Il risultato di un round di cifratura è quindi 101001010001.

3.2 *La crittoanalisi lineare del cifrario*

Per crittanalizzarlo dobbiamo calcolare i valori di NS_1 e NS_2 per ogni $\alpha \neq 0$ e $\beta \neq 0$ (si ricordi che NS_i è il valore di NS relativo alla S-Box i, quindi $NS_i(\alpha, \beta) = |\{X \in \{0,1\}^n \ t.c. \ X \cdot \alpha = SB(X) \cdot \beta\}|$.). Per calcolare $NS_1(0011, 011)$ consideriamo tutti i possibili input di SB_1, dunque tutti i possibili numeri binari di 4 bit, e calcoliamo i corrispondenti output.

$SB_1(0000) = 101$
$SB_1(0001) = 010$
$SB_1(0010) = 001$
$SB_1(0011) = 110$
$SB_1(0100) = 011$
$SB_1(0101) = 100$
$SB_1(0110) = 111$
$SB_1(0111) = 000$
$SB_1(1000) = 001$
$SB_1(1001) = 100$
$SB_1(1010) = 110$
$SB_1(1011) = 010$
$SB_1(1100) = 000$
$SB_1(1101) = 111$
$SB_1(1110) = 101$
$SB_1(1111) = 011$

A questo punto prendiamo due valori di α e β, ad esempio $\alpha = 0011$ (quindi considereremo lo XOR degli ultimi due bit di input) e $\beta = 011$ (quindi considereremo lo XOR degli ultimi due bit di output). I valori così ottenuti sono uguali in 8 casi su 16, dunque questa relazione non è utile.
Determinando tutti gli NS (sono 105 valori non banali, infatti se $\alpha = 0$ o $\beta = 0$ otterremo sempre 8 tranne nel caso in cui sono entrambi 0 in cui otterremo 16, ognuno dei quali è il risultato di 16 computazioni, non è un'operazione lunga), si osserva che il caso migliore è dato da $NS_2(0011, 100) = 2$. Questa relazione considera gli ultimi due bit dell'input della seconda SBox, che sono i bit in posizione 10 e 11 dell'input iniziale, e il primo bit di output della seconda SBox, che è il bit in posizione 9 dell'output finale. Inoltre

durante l'operazione di XOR con L_1 al bit di output viene sommato il bit di input in posizione 3. Infine i bit di chiave sommati ai bit di input sono in posizione 6 e 7.
Di conseguenza si determina l'espressione lineare

$$I[3, 10, 11] \bigoplus O[9] = K[6, 7]$$

con probabilità $p = \frac{NS_2(0011,100)}{16} = \frac{2}{16} = \frac{1}{8}$.
Dalla proposizione esposta in precedenza segue che l'algoritmo descritto ha successo nel $99,8\%$ dei casi considerando $N = 2|p - \frac{1}{2}|^{-2} = \frac{128}{9} = 14,\overline{2}$ coppie di testo in chiaro e rispettivo cifrato, quindi con l'analisi di soli 15 testi possiamo determinare lo XOR tra due bit della chiave, riducendo di uno le incognite del problema.
Chiaramente i valori scelti di α e β erano quelli ottimali, ma non gli unici con $NS \neq 8$, quindi compiendo altre scelte otterremo altre equazioni che consentiranno di ridurre ulteriormente le incognite.
Proviamo ora ad analizzare un cifrario che è stato utilizzato nella pratica a livello mondiale, il DES.

Capitolo 4

Il cifrario DES

4.1 *Il cifrario*

Proviamo ad applicare la tecnica esposta al cifrario DES (Data Encryption Standard).
DES è stato autorizzato e divulgato nel 1975 dal NIST (National Institute of Standards and Technology) ed è stato utilizzato intensivamente per la sicurezza informatica per molti anni, finché negli anni '90 sono stati presentati alcuni attacchi particolarmente efficaci contro di esso. Ad oggi il cifrario è molto datato e meno utilizzato a causa delle debolezze che ne sono state trovate, una delle quali ha dato appunto origine alla crittoanalisi lineare.
Andiamo a descrivere il cifrario.
La versione di DES che consideriamo è un cifrario a blocchi che prende in input insiemi di 64 bit e restituisce output di 64 bit, utilizzando una chiave di 64 bit. Ci sono versioni che lavorano su sequenze di dimensioni diverse, ad esempio 128 bit, ma il metodo operativo è lo stesso, cambiano le funzioni di round che devono essere adattate alla dimensioni di input e output.
In questo lavoro approfondiremo, appunto, la versione a 64 bit. Dei bit di chiave quelli in posizione $\equiv 7 \pmod 8$ sono bit di controllo ed equivalgono allo XOR dei precedenti 7.
La funziona di cifratura è la composizione di una permutazione iniziale, una funzione di round applicata il numero di volte previsto e una permutazione finale, che è l'inversa di quella iniziale. Nel caso considerato il numero di round previsto è 16.
Ad ogni round lo stato S_i viene diviso tra metà destra R_i e metà sinistra L_i che verranno trattate separatamente. La funzione di round F agisce su R_i per ottenere R_{i+1}, mentre L_{i+1} si ricava dallo stato precedente:
$L_{i+1} = R_i$.

$R_{i+1} = F(R_i, K_i) \bigoplus L_i$
dove K_i è l'i-esima chiave di round.
La funzione di round è una funzione $F : \{0,1\}^{32} \longrightarrow \{0,1\}^{32}$ ed è la composizione di un'espansione, una somma con la chiave, una sostituzione di bit e una permutazione. La funzione F si può quindi esprimere come $F(R,K) = P(S(E(R) \bigoplus K))$. P è una permutazione, E è una funzione di espansione e S di sostituzione.

- L'espansione è una funzione $E : \{0,1\}^{32} \longrightarrow \{0,1\}^{48}$ tale che $E(0,1,\ldots,31) = (31,0,1,2,3,4,3,4,5,6,7,8,7,8,\ldots,29,30,31,0)$. Dunque ogni 4 bit (a partire dal numero 1, quindi il secondo), gli ultimi due vengono replicati.

- La somma con la chiave è il semplice XOR dello stato con la chiave di round.

- La funzione di sostituzione S viene descritta attraverso la rappresentazione tramite S-Box.

- la permutazione è un elemento del gruppo simmetrico S_{32}.

Scendiamo ora nel dettaglio delle ultime tre funzioni.
Iniziamo dalla definzione della chiave di round K_i. Questa viene calcolata a ogni round a partire dai 56 bit significativi della chiave di partenza K, cioè $K_0 \ldots K6, K_8 \ldots K_{14}, \ldots, K_{56} \ldots K_{62}$ poiché i bit in posizioni multiple di 8 sono i bit di controllo. A essi viene applicata un permutazione iniziale P_K. Poi ad ogni round le due metà del risultato del round precedente vengono shiftate verso sinistra, rispettivamente di 1 posto ai round 1, 2, 9 e 16 e di 2 posti agli altri round. Quindi, definite K_L e K_R le metà sinistra e destra della chiave, avremo che ai round 1, 2, 9 e 16

$$(K_L, K_R) \longleftarrow (K_L <<< 1, K_R <<< 1)$$

mentre agli altri round

$$(K_L, K_R) \longleftarrow (K_L <<< 1, K_R <<< 1)$$

dove $(b_0, b_1, b_2, \ldots, b_n) <<< k = (b_k, b_{k+1}, \ldots, b_n, b_0, b_1, \ldots, b_{k-1}$.
Una volta ottenuto il risultato dello shift la chiave di round si deduce da questo dai 48 bit nelle posizioni (13, 16, 10, 23, 0, 4, 2, 27, 14, 5, 20, 9, 22, 18, 11, 3, 25, 7, 15, 6, 26, 19, 12, 1, 40, 51, 30, 36, 46, 54, 29, 39, 50, 44, 32, 47, 43, 48, 38, 55, 33, 52, 45, 41, 49, 35, 28, 31).
La funzione S è determinata da 8 SBox, $SB_1, \ldots, SB_8$. Queste sono funzioni

$$SB_i : \{0,1\}^6 \to \{0,1\}^4$$

e sono rappresentate come tabelle 4×16. Lo stato viene quindi diviso in 8 sottoinsiemi di 6 bit consecutivi, ognuno dei quali entra in una SBox.
Il funzionamento delle S-Box è simile a quelle del cifrario esempio. In questo caso il primo e l'utimo bit di input indicano la riga del valore di output, mentre i quattro bit centrali la colonna.
Le SBox sono:

$$SB_1 = \begin{bmatrix} 14 & 4 & 13 & 1 & 2 & 15 & 11 & 8 & 3 & 10 \\ 6 & 12 & 5 & 9 & 0 & 7 & & & & \\ 0 & 15 & 7 & 4 & 14 & 2 & 13 & 1 & 10 & 6 \\ 12 & 11 & 9 & 5 & 3 & 8 & & & & \\ 4 & 1 & 14 & 8 & 13 & 6 & 2 & 11 & 15 & 12 \\ 9 & 7 & 3 & 10 & 5 & 0 & & & & \\ 15 & 12 & 8 & 2 & 4 & 9 & 1 & 7 & 5 & 11 \\ 3 & 14 & 10 & 0 & 6 & 13 & & & & \end{bmatrix}$$

$$SB_2 = \begin{bmatrix} 15 & 1 & 8 & 14 & 6 & 11 & 3 & 4 & 9 & 7 \\ 2 & 13 & 12 & 0 & 5 & 10 & & & & \\ 3 & 13 & 4 & 7 & 15 & 2 & 8 & 14 & 12 & 0 \\ 1 & 10 & 6 & 9 & 11 & 5 & & & & \\ 0 & 14 & 7 & 11 & 10 & 4 & 13 & 1 & 5 & 8 \\ 12 & 6 & 9 & 3 & 2 & 15 & & & & \\ 13 & 8 & 10 & 1 & 3 & 15 & 4 & 2 & 11 & 6 \\ 7 & 12 & 0 & 5 & 14 & 9 & & & & \end{bmatrix}$$

$$SB_3 = \begin{bmatrix} 10 & 0 & 9 & 14 & 6 & 3 & 15 & 5 & 1 & 13 \\ 12 & 7 & 11 & 4 & 2 & 8 & & & & \\ 13 & 7 & 0 & 9 & 3 & 4 & 6 & 10 & 2 & 8 \\ 5 & 14 & 12 & 11 & 15 & 1 & & & & \\ 13 & 6 & 4 & 9 & 8 & 15 & 3 & 0 & 11 & 1 \\ 2 & 12 & 5 & 10 & 14 & 7 & & & & \\ 1 & 10 & 13 & 0 & 6 & 9 & 8 & 7 & 4 & 15 \\ 14 & 3 & 11 & 5 & 2 & 12 & & & & \end{bmatrix}$$

$$SB_4 = \begin{bmatrix} 7 & 13 & 14 & 3 & 0 & 6 & 9 & 10 & 1 & 2 \\ 8 & 5 & 11 & 12 & 4 & 15 & & & & \\ 13 & 8 & 11 & 5 & 6 & 15 & 0 & 3 & 4 & 7 \\ 2 & 12 & 1 & 10 & 14 & 9 & & & & \\ 10 & 6 & 9 & 0 & 12 & 11 & 7 & 13 & 15 & 1 \\ 3 & 14 & 5 & 2 & 8 & 4 & & & & \\ 3 & 15 & 0 & 6 & 10 & 1 & 13 & 8 & 9 & 4 \\ 5 & 11 & 12 & 7 & 2 & 14 & & & & \end{bmatrix}$$

$$SB_5 = \begin{bmatrix} 2 & 12 & 4 & 1 & 7 & 10 & 11 & 6 & 8 & 5 \\ 3 & 15 & 13 & 0 & 14 & 9 & & & & \\ 14 & 11 & 2 & 12 & 4 & 7 & 13 & 1 & 5 & 0 \\ 15 & 10 & 3 & 9 & 8 & 6 & & & & \\ 4 & 2 & 1 & 11 & 10 & 13 & 7 & 8 & 15 & 9 \\ 12 & 5 & 6 & 3 & 0 & 14 & & & & \\ 11 & 8 & 12 & 7 & 1 & 14 & 2 & 13 & 6 & 15 \\ 0 & 9 & 10 & 4 & 5 & 3 & & & & \end{bmatrix}$$

$$SB_6 = \begin{bmatrix} 12 & 1 & 10 & 15 & 9 & 2 & 6 & 8 & 0 & 13 \\ 3 & 4 & 14 & 7 & 5 & 11 & & & & \\ 10 & 15 & 4 & 2 & 7 & 12 & 9 & 5 & 6 & 1 \\ 13 & 14 & 0 & 11 & 3 & 8 & & & & \\ 9 & 14 & 15 & 5 & 2 & 8 & 12 & 3 & 7 & 0 \\ 4 & 10 & 1 & 13 & 11 & 6 & & & & \\ 4 & 3 & 2 & 12 & 9 & 5 & 15 & 10 & 11 & 14 \\ 1 & 7 & 6 & 0 & 8 & 13 & & & & \end{bmatrix}$$

$$SB_7 = \begin{bmatrix} 4 & 11 & 2 & 14 & 15 & 0 & 8 & 13 & 3 & 12 \\ 9 & 7 & 5 & 10 & 6 & 1 & & & & \\ 13 & 0 & 11 & 7 & 4 & 9 & 1 & 10 & 14 & 3 \\ 5 & 12 & 2 & 15 & 8 & 6 & & & & \\ 1 & 4 & 11 & 13 & 12 & 3 & 7 & 14 & 10 & 15 \\ 6 & 8 & 0 & 5 & 9 & 2 & & & & \\ 6 & 11 & 13 & 8 & 1 & 4 & 10 & 7 & 9 & 5 \\ 0 & 15 & 14 & 2 & 3 & 12 & & & & \end{bmatrix}$$

$$SB_8 = \begin{bmatrix} 13 & 2 & 8 & 4 & 6 & 15 & 11 & 1 & 10 & 9 \\ 3 & 14 & 5 & 0 & 12 & 7 & & & & \\ 1 & 15 & 13 & 8 & 10 & 3 & 7 & 4 & 12 & 5 \\ 6 & 11 & 0 & 14 & 9 & 2 & & & & \\ 7 & 11 & 4 & 1 & 9 & 12 & 14 & 2 & 0 & 6 \\ 10 & 13 & 15 & 3 & 5 & 8 & & & & \\ 2 & 1 & 14 & 7 & 4 & 10 & 8 & 13 & 15 & 12 \\ 9 & 0 & 3 & 5 & 6 & 11 & & & & \end{bmatrix}$$

Per esempio l'output della prima S-Box di 101000 è $SB_1(101000) = 13_{10} = 1101_2$, infatti 13 è il quinto elemento ($0100_2 = 4_{10}$) della terza riga ($10_2 = 2_{10}$).
La funzione di permutazione P vale

$$P = (0, 15, 9, 14, 30, 3, 20, 31, 24, 18, 23, 8)$$

$$(1, 6, 27, 5, 11, 25, 12, 4, 28, 21, 26, 29, 10, 22, 2, 19, 13, 17, 7, 16)$$

Una rappresentazione grafica dello schema di cifratura e una della funzione di round si possono vedere rispettivamente in Figura 4.1 e in Figura 4.2

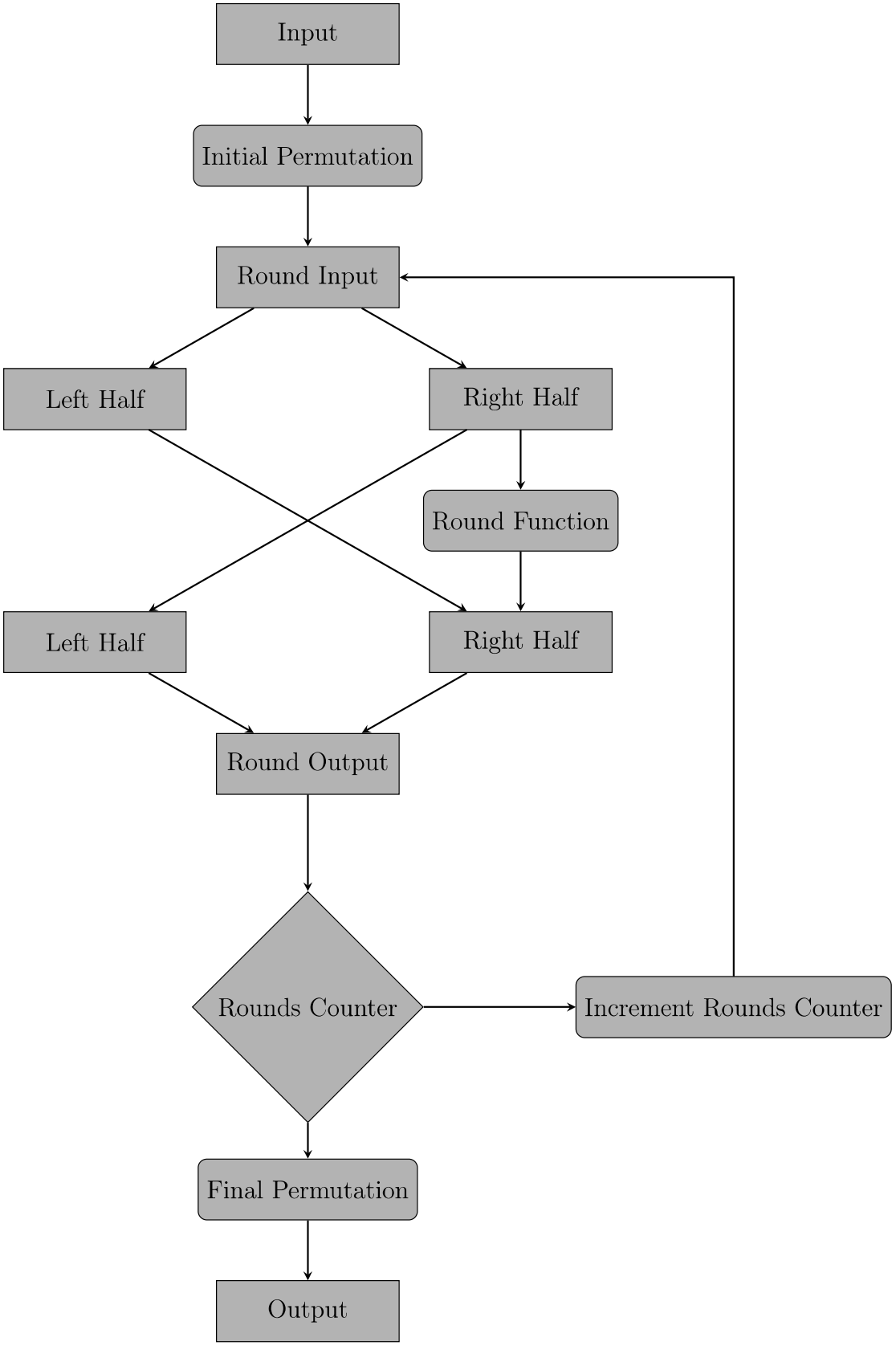

Figura 4.1: Schema di funzionamento di DES

e della funzione di round

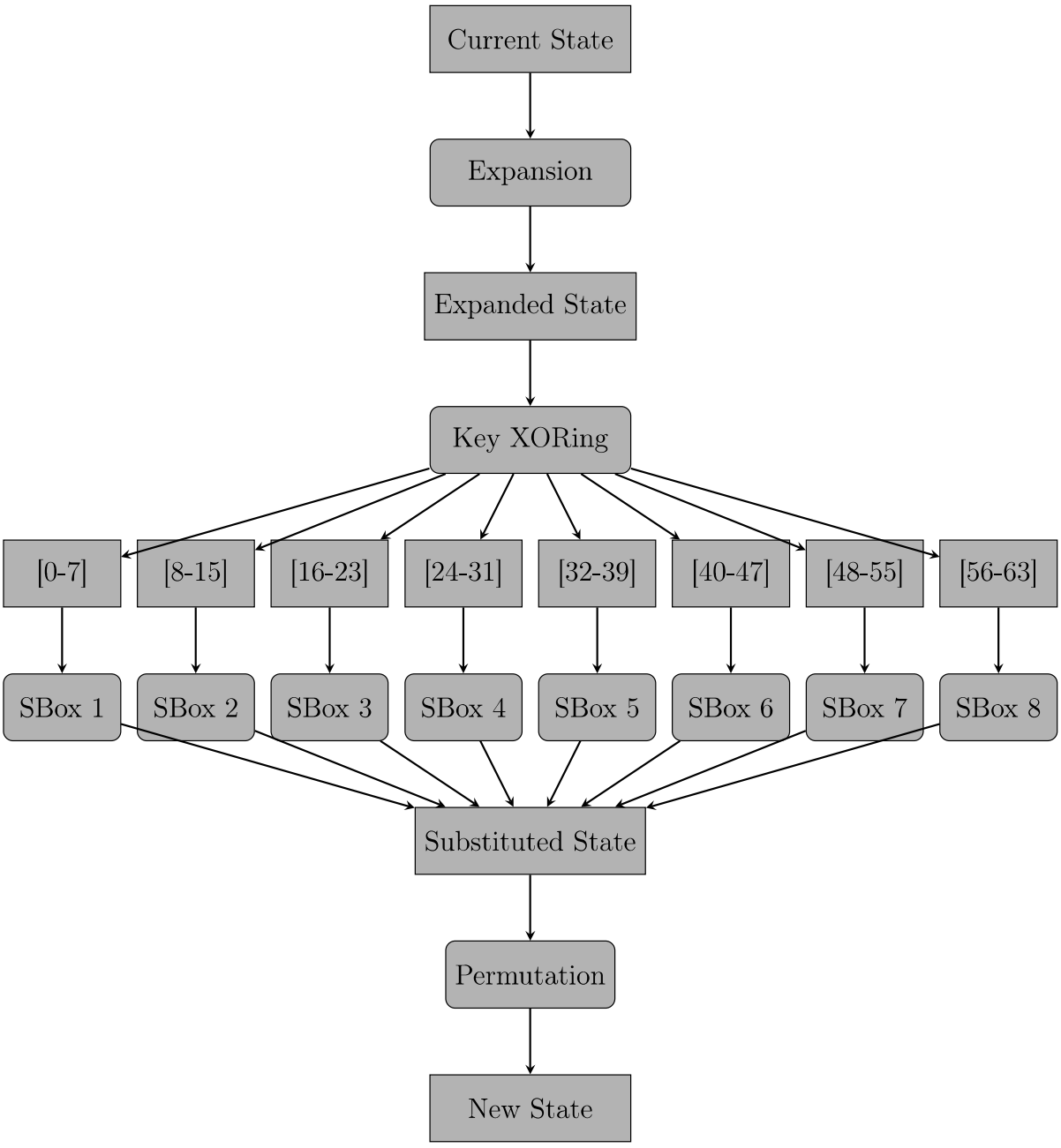

Figura 4.2: Funzione di round di DES

4.2 La crittoanalisi lineare di DES a 3 round

Iniziamo con la crittoanalisi di DES su 3 round. Poi descriveremo l'analisi di sicurezza su 5 round. Infine presenteremo i risultati ottenuti in letteratura su 8 round e sul cifrario completo.
La struttura di DES consente di sfruttare un'equazione su un round ed estenderla a tre round senza dover cercare equazioni sugli ultimi round che si adattino alla scelta di input e output.
Consideriamo infatti un'espressione valida con probabilità p al primo round, che derivi dall'analisi statistica di una SBox, quindi del tipo

$$R_1[r_1, \ldots, r_n] \bigoplus F(R_1, K_1)[l_1, \ldots, l_m] = K_1[k_1, \ldots, k_l]$$

dove R_1 è la metà destra dello stato iniziale.
Consideriamo poi la stessa equazione al terzo round, dunque

$$R_3[r_1, \ldots, r_n] \bigoplus F(R_3, K_3)[l_1, \ldots, l_m] = K_3[k_1, \ldots, k_l]$$

Valgono le seguenti relazioni:

- $R_1 = L_I$, metà destra dell'output iniziale
- $F(R_1, K_1) = R_2 + L_1 = R_2 + L_I$
- $R_3 = L_4 = R_O$, metà destra dell'output finale
- $F(R_3, K_3) = R_4 + L_3 = L_O + L_3$
- $L_3 = R_2$

Sostituendo nelle equazioni lineari otteniamo

$$R_I[r_1, \ldots, r_n] \bigoplus R_2[l_1, \ldots, l_m] \bigoplus L_I[l_1, \ldots, l_m] = K_1[k_1, \ldots, k_l]$$

e

$$R_O[r_1, \ldots, r_n] \bigoplus L_O[l_1, \ldots, l_m] \bigoplus L_3[l_1, \ldots, l_m] = K_3[k_1, \ldots, k_l]$$

$K_1[k_1, \ldots, k_l] = K[i_1, \ldots, i_l]$ e $K_3[k_1, \ldots, k_l] = K[j_1, \ldots, j_l]$, noti. Sommando le equazioni, tenendo presente che $R_2 = L_3$, si ottiene quindi

$$(R_I \bigoplus R_O)[r_1, \ldots, r_n] \bigoplus (L_I \bigoplus L_O)[l_1, \ldots, l_m] = K[i_1, \ldots, i_l, j_1, \ldots, j_l]$$

Applicando poi le permutazioni iniziale e finale si ottiene un'espressione classica del tipo

$$I[a_1, \ldots, a_{n+m}] \bigoplus O[b_1, \ldots, b_{n+m}] = K[c_1, \ldots, c_{2l}]$$

Questa equazione sarà valida con probabilità $p^2 + (1-p)^2$.
Dal calcolo degli NS_i risulta che la migliore espressione possibile su un round deriva da $NS_5(010000, 1111) = 12$, che dunque fornisce un'espressione lineare valida con probabilità $\frac{12}{64} = \frac{3}{16}$.
In particolare la migliore equazione determinabile sul primo round è

$$R_1[16] + F(R_1, K_1)[2, 7, 13, 24] = K_1[27]$$

Sommata con l'equazione al terzo round corrispondente ed effettuate le dovute operazioni si ottiene, prima delle permutazioni,

$$I[2, 7, 13, 24, 48] \bigoplus O[2, 7, 13, 24, 48] = K_1[27] \bigoplus K_3[27]$$

che vale con probabilità

$$p = (\frac{3}{16})^2 + (\frac{13}{16})^2 = \frac{89}{128} = 69,53125\%$$

Applicando l'algoritmo precedentemente esposto è quindi sufficiente controllare circa 50 input per trovare l'informazione sui bit di chiave.

4.3 *La crittoanalisi lineare di DES a più di 3 round*

L'estensione della crittoanalisi al cifrario a 5 round comporta un passaggio ulteriore, sebbene sia molto simile al caso precedente. In questo caso ripetiamo il procedimento fatto in precedenza ma considerando il secondo e il quarto round invece che il primo e l'ultimo, ottenendo un'espressione lineare che lega l'input al secondo round all'output al quarto.
Una volta determinata l'equazione si estende verso il primo e l'ultimo round sommandoci le equazioni migliori possibili che consentano di semplificare i termini dell'equazione relativi al secondo e al quarto round.
Nel caso specifico dell'equazione trovata in precedenza per i 3 round le equazioni che estendono l'espressione sono derivate da $NS_1(011011, 0100) = 22$ applicata su primo e quinto round.
Di conseguenza si ottiene l'espressione finale

$$(L_I \bigoplus L_O)[16] \bigoplus (R_I \bigoplus R_O)[0, 1, 2, 3, 4, 7, 13, 24] =$$

$$= (K_1 \bigoplus K_5)[1, 2, 4, 5] \bigoplus (K_2 \bigoplus K_4)[27]$$

che vale con probabilità

$$p = \frac{1}{2} + 2^3(\frac{12}{64} - \frac{1}{2})^2(\frac{22}{64} - \frac{1}{2})^2$$

per il Piling-up lemma.

$$p = \frac{1}{2} + \frac{5^4}{2^{15}}$$

che è circa $0,519$. Dunque occorrono circa 2800 testi affinché l'algoritmo abbia successo nel $99,8\%$ dei casi.
Su 8 e 16 round l'algoritmo è stato applicato con successo utilizzando rispettivamente 2^{21} e 2^{47} testi, dunque un numero di computazioni dell'algoritmo di cifratura minori di 2^{56} che corrispondono a una ricerca esaustiva sulla chiave.

Capitolo 5

Il cifrario GIFT-128

5.1 *Il cifrario*

Il cifrario su cui abbiamo applicato le tecniche precedentemente viste è il cifrario GIFT-128 [3,11], un cifrario a blocchi finalista nella gara del NIST per la standardizzazione della sicurezza dei cifrari lightweight.
Il cifrario prende in input testi di 128 bit e restituisce in output testi della stessa lunghezza usando chiavi di lunghezza 128 bit.
Rappresentiamo lo stato S del cifrario come una tabella 4×32, dunque 4 righe da 32 bit, $S = (S^0, S^1, S^2, S^3)$.
La funzione di cifratura è composta dalla sola funzione di round iterata il numero previsto di volte. La funzione di round è una funzione $F : \{0,1\}^{128} \longrightarrow \{0,1\}^{128}$ ed è la composizione di tre operazioni, una sostituzione, una permutazione e la somma con la chiave.

- La funzione di sostituzione è una funzione $SB : \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32} \longrightarrow \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32}$.

- La funzione di permutazione è un elemento del gruppo simmetrico S_{128}, sebbene agisca indipendentemente sui 4 S^i, dunque può anche essere rappresentata come quattro elementi del gruppo S_{32}.

- La somma con la chiave è lo XOR di S^1 e S^2 con la chiave di round.

Approfondiamo ora le tre operazioni. Gli autori del cifrario danno una rappresentazione in forma di combinazione di operatori logici della funzione di sostituzione [3], ma questa può essere descritta anche tramite l'uso delle S-Box.

Nella formulazione adottata dagli autori si operano in ordine le seguenti sostituzioni:

1. $S^1 = S^1 \bigoplus (S^0 \& S^2)$
2. $S^0 = S^0 \bigoplus (S^1 \& S^3)$
3. $S^2 = S^2 \bigoplus (S^0 | S^1)$
4. $S^3 = S^3 \bigoplus S^2$
5. $S^1 = S^1 \bigoplus S^3$
6. $S^3 = S^3 \bigoplus 1 (= not S^3)$
7. $S^2 = S^2 \bigoplus (S^0 \& S^1)$
8. $(S^0, S^1, S^2, S^3) = (S^3, S^1, S^2, S^0)$

Queste operazioni agiscono indipendentemente sulle colonne dello stato e lo fanno nello stesso modo su ogni colonna, quindi si possono calcolare gli output su tutte le possibili colonne e definire la funzione come una S-Box di una sola riga dove la conversione in decimale dell'input indica direttamente la posizione all'interno della riga del valore da scegliere.

$SB = [1000, 0100, 0110, 1010, 0010, 1101, 1100, 0001, 0101, 1011, 1111, 0000, 0011, 1110, 1001, 0$

Se ad esempio una colonna avesse il valore 0101 allora $SB(0101) = 1101$, infatti $0101_2 = 5_{10}$ e il sesto valore della S-Box è 1101. D'altra parte operando le sostituzioni della rappresentazione precedente si ottiene

$$0101 \to 0101 \to 1101 \to 1111 \to 1110 \to 1110 \to 1111 \to 1101 \to 1101$$

Il secondo strato è dato dalla permutazione delle righe. Ogni riga S^i è sottoposta a una permutazione dei suoi 32 bit P_i, elemento. Le permutazioni sono

$$P_1 = (1, 24, 6, 17, 28, 7, 9, 26, 22, 21, 29, 31, 15, 11,$$
$$10, 18, 20, 5, 25, 30, 23, 13, 27, 14, 19, 12, 3, 8, 2, 16, 4)$$
$$P_2 = (0, 8, 10, 26, 30, 31, 23, 21, 5, 1)(2, 24, 14, 27, 22, 29, 7, 17, 4, 9)$$
$$(3, 16, 12, 11, 18, 28, 15, 19, 20, 13)(6, 25)$$
$$P_3 = (0, 16, 20, 21, 13, 11, 26, 6, 1, 8, 18, 4, 17, 12,$$
$$19, 28, 23, 29, 15, 27, 30, 7, 25, 14, 3, 24, 22, 5, 9, 10, 2)$$

$$P_4 = (0, 24, 30, 15, 3)(1, 16, 28, 31, 7)(2, 8, 26, 14, 11)$$
$$(4, 25, 22, 13, 19)(5, 17, 20, 29, 23)(6, 9, 18, 12, 27)$$

Infine allo stato viene sommata la chiave di round.
La chiave complessiva viene rappresentata come una tabella 4×32 come lo stato, $K = (K^0, K^1, K^2, K^3)$. Indicheremo inoltre con K_L^i e K_R^i le due metà di K^i, rispettivamente sinistra e destra. A partire dalla chiave complessiva la chiave di round viene calcolata come segue

- $K \longleftarrow K >>> 32$, dunque K viene ruotata verso destra di 32 bit e diventa (K^3, K^0, K^1, K^2).
- $K_L^3 \longleftarrow K_L^3 >>> 2$, dunque K^3 viene ruotata verso destra di 2 bit.
- $K_R^3 \longleftarrow K_L^3 >>> 12$, dunque K_R^3 viene ruotata verso destra di 12 bit.

Quindi a ogni round avremo che la chiave complessiva diventa

$$K' = (K^3, K^0, K^1, (K_L^2[14:15], K_L^2[0:13], K_R^2[4:15], K_R^2[0:3]$$

La funzione C di somma con la chiave è tale che

$$C(S^0, S^1, S^2, S^3 K^0, K^1, K^2, K^3) = (S^0, S^1 \bigoplus K^2, S^2 \bigoplus K^1, S^3 \bigoplus c)$$

dove c è una costante che dipende dal round in corso.
La costante di round è una sequenza di 32 bit e viene rappresentata come $1000 \ldots 0 c_5 c_4 \ldots c_0$. Questi bit sono definiti ricorsivamente. Prima di iniziare la cifratura sono impostati a 0. Ad ogni round vengono aggiornati come segue

- $c_5 \longleftarrow c_4$
- $c_4 \longleftarrow c_3$
- $c_3 \longleftarrow c_2$
- $c_2 \longleftarrow c_1$
- $c_1 \longleftarrow c_0$
- $c_0 \longleftarrow c_5 \bigoplus c_4 \bigoplus 1$

Si tenga presente che l'aggiornamento della chiave viene effettuato dopo l'applicazione della funzione C di somma con la chiave.
Il numero di round proposto per la cifratura è 40.
Scendiamo ora in profondità nella crittoanalisi lineare.

5.2 *La crittoanalisi lineare di GIFT-128*

Chiaramente in questo caso le strategie utilizzate per la crittoanalisi lineare del DES derivanti dalla struttura del cifrario non sono utilizzabili, infatti nessuna porzione dell'input a un round viene trasportata come porzione dell'input al round successivo ma tutto lo stato passa attraverso la funzione di cifratura. Tuttavia la piccola dimensione delle SBox consente di effettuare un'analisi delle equazioni su un singolo round molto efficiente.
Gli autori del cifrario hanno effettuato uno studio di crittoanalisi lineare su GIFT-128 e ne riportano i risultati in [3]. Attraverso un'analisi standard a 9 round del cifrario l'espressione lineare migliore che trovano ha un potenziale di correlazione (si veda il capitolo 2) medio di circa 2^{-46}. Quindi con 27 round, supponendo di continuare a ottenere la complessità migliore possibile ogni 9 round, questo numero scenderebbe sotto 2^{-128} e 40 round sarebbero dunque sufficienti per rendere il cifrario robusto contro questo tipo di analisi.
Come detto in precedenza la complessità della crittoanalisi lineare deriva da due parti dell'algoritmo. Da una parte c'è la ricerca delle migliori equazioni lineari, dall'altra l'analisi delle coppie di input/output. In questo lavoro di tesi abbiamo tentato 3 approcci principali per la ricerca della migliore equazione:

1. Un approccio basilare in cui come input iniziale si prende una singola colonna non nulla in tutto lo stato e successivamente si procede cercando la continuazione migliore in termini di probabilità (e in caso di equivalenza si opera la scelta con il minor impatto possibile sulle colonne non nulle) ad ogni round.

2. Un approccio del tipo meet in the middle, dunque la ricerca di un'espressione ottimale sulla metà dei round considerati congiunta a un'espressione ottimale in senso inverso.

3. Una ricerca di espressioni lineari non singolarmente ottimali ma che riducessero la propagazione dei bit di stato considerati nelle equazioni di round.

Successivamente abbiamo applicato l'algoritmo 1 di Matsui con le equazioni trovate per recuperare alcuni bit della chiave del cifrario su un numero di round ridotto.
Poiché i risultati esposti dagli autori per giustificare la scelta dei 40 round sono relativi a un'analisi su 9 round ci siamo concentrati sul cifrario ridotto a 9 round.
Per poter calcolare le migliori equazioni lineari e la relativa probabilità è fondamentale il calcolo dei valori di $NS(\alpha, \beta)$. Di seguito riportiamo una tabella

con i corrispondenti valori. Si tenga presente che i valori di input possibili sono 16, dunque più un valore è lontano da 8 più è efficace l'equazione a esso corrispondente. A partire da questa tabella si può ricostruire facilmente la

α/β	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	8	8	8	8	8	8	8	8	4	4	8	8	4	12
2	8	8	8	8	8	8	4	12	8	8	8	8	8	8	4	4
3	8	8	8	8	12	12	8	8	12	4	8	8	8	8	8	8
4	8	10	8	10	8	10	8	10	8	10	12	6	4	6	8	10
5	8	6	8	6	4	10	12	10	8	6	8	6	8	6	8	6
6	8	6	12	10	8	6	8	6	8	6	8	6	4	10	8	6
7	8	10	12	6	8	10	8	10	4	6	8	10	8	10	8	10
8	8	12	10	6	8	8	6	6	8	8	6	6	8	4	10	6
9	8	12	6	10	8	8	10	10	8	8	6	6	8	12	10	6
10	8	8	10	10	8	12	10	6	8	12	6	10	8	8	6	6
11	8	8	6	6	12	8	10	6	4	8	10	6	8	8	6	6
12	8	10	6	8	8	6	10	8	8	6	6	12	4	6	6	8
13	8	6	10	8	12	6	10	12	8	10	6	8	8	6	10	8
14	8	10	10	12	8	6	10	8	8	6	10	8	12	6	6	8
15	8	6	6	12	8	10	6	8	4	6	6	8	8	6	10	8

Tabella 5.1: $NS(\alpha, \beta)$

matrice di correlazione di una S-Box, semplicemente sostituendo al valore x il valore $x' = 2(\frac{x}{16}) - 1$ Si possono trovare una rappresentazione del cifrario e una della funzione di round rispettivamente in Figura 5.1 e in Figura 5.2

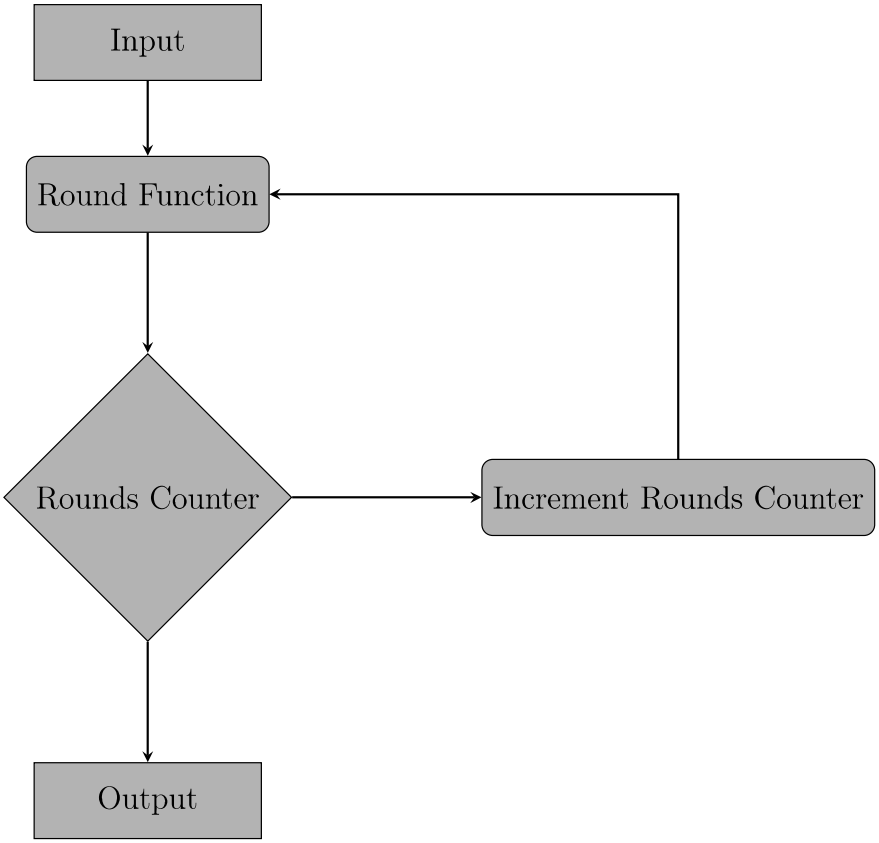

Figura 5.1: Schema di funzionamento di GIFT-128

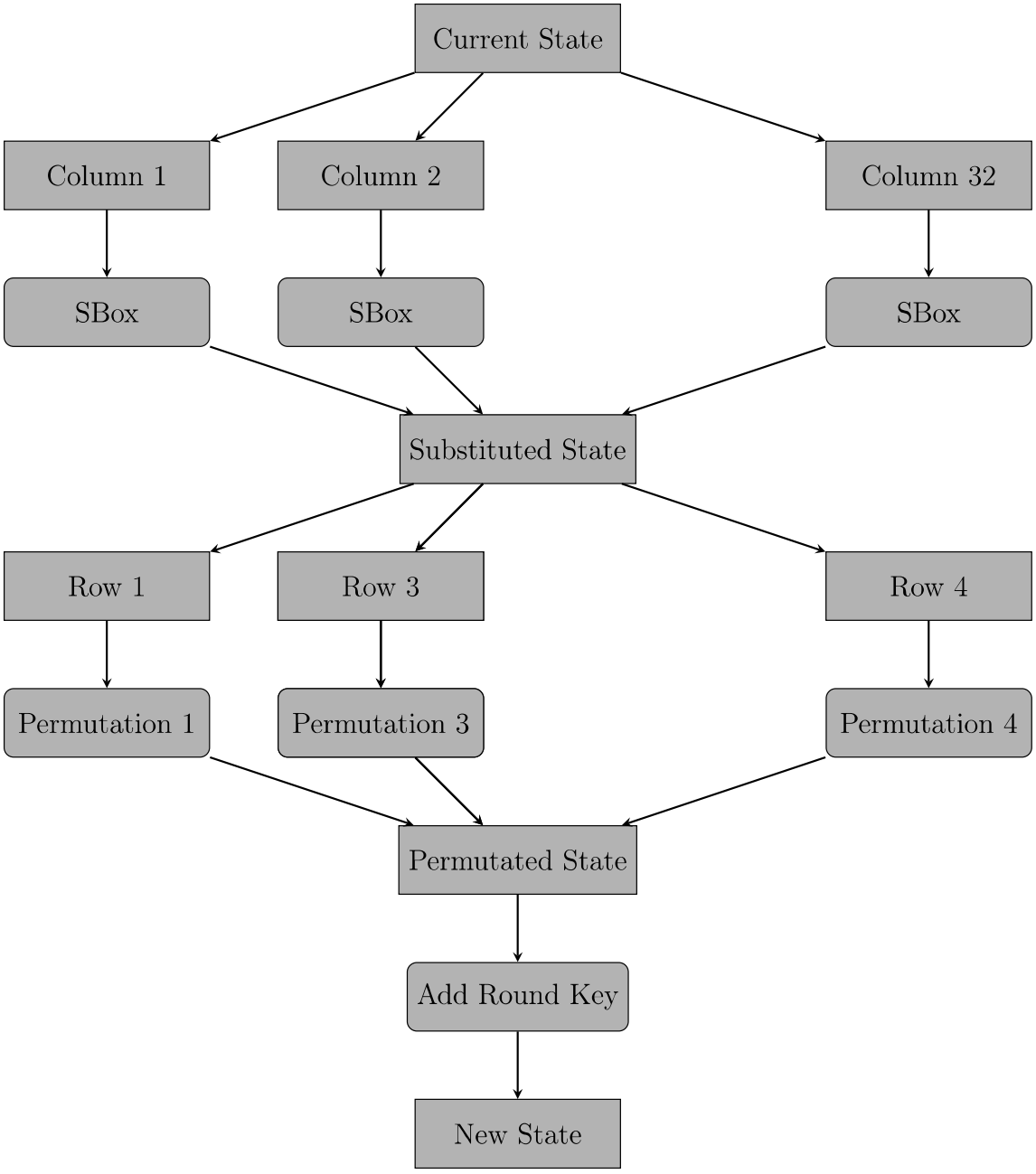

Figura 5.2: Funzione di round di GIFT-128

5.3 *L'approccio basilare*

Si tenga presente che le 32 colonne dello stato sono trattate indipendentemente nella ricerca dei trail, poiché ognuna fa riferimento a una S-Box indipendente. Di conseguenza se le colonne in input a un round sono c allora l'equazione di quel round sarà il risultato della combinazione di c equazioni differenti, ognuna che da il proprio contributo al calcolo complessivo della probabilità.
In generale, quindi, un buon modo per migliorare l'efficienza dell'equazione finale è quello di ridurre il numero di equazioni lineari da prendere in considerazione ad ogni round, dunque il numero di colonne "attive" ad ogni round.

Definizione 5.3.1 (Colonna Attiva)**.** *Diciamo che la colonna c dello stato è attiva al round r se nel calcolo dell'equazione lineare totale consideriamo un'equazione lineare derivata dalla colonna c al round r. Alternativamente la colonna c è attiva al round r se nell'input al round r dell'equazione lineare almeno un bit della colonna c viene considerato.*

Per ridurre il numero di colonne attive in questo approccio, quindi, abbiamo scelto come stato input uno stato con esattamente una colonna attiva, il che da un insieme di 480 input iniziali possibili (32 colonne di cui attivarne una e 15 valori possibili di α con cui renderla attiva). Per ognuno di questi 480 stati iniziali possibili abbiamo proceduto con la scelta del valore di β corrispondente alla migliore probabilità possibile. In caso di parità di probabilità si sceglie il β con il minor numero di bit valorizzati a 1, in modo tale che venendo poi permutati si attivino meno colonne possibili. Questo approccio l'abbiamo utilizzato per analizzare il cifrario su 5 round al massimo, ottenendo equazioni lineari valide con probabilità di 0,484375. Ci siamo fermati al quinto round perché approcci più efficaci di questo richiedevano l'utilizzo di quello basilare su 5 round.
In Figura 5.3 si può un esempio pratico di questo metodo

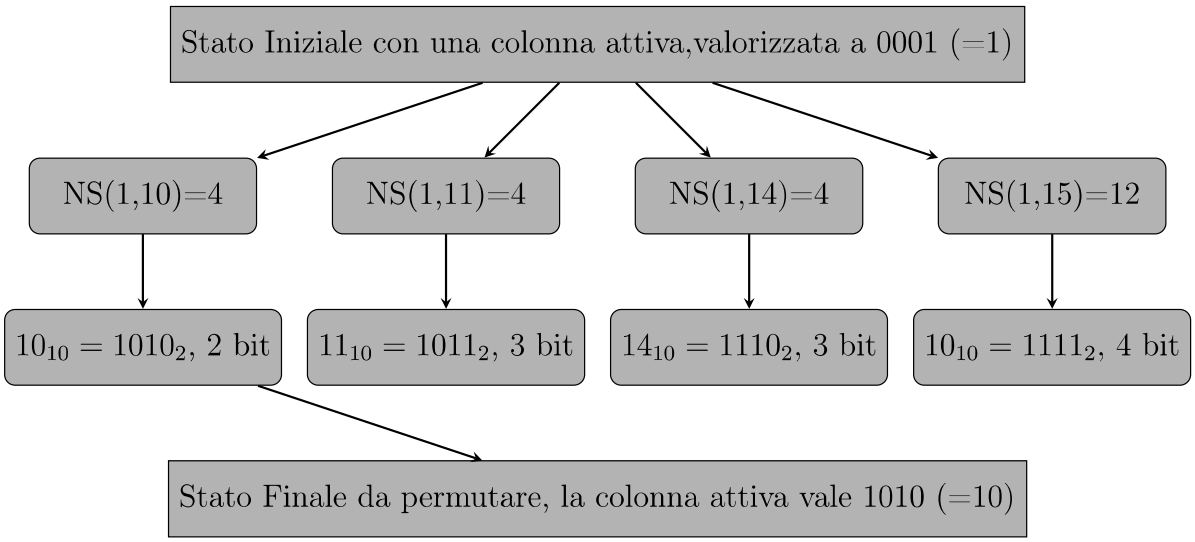

Figura 5.3: Esempio di scelta della prosecuzione di un trail

Una delle equazioni migliori su 5 round (ce ne sono molteplici con la stessa efficienza) coinvolge in input il solo bit in prima riga e prima colonna nello stato iniziale. In output 8 colonne sono attive e l'equazione risultante ha una probabilità di $\frac{1}{2} - 2^{-14}$.
Su 3 round invece la probabilità è $\frac{1}{2} - 2^{-8}$ e una delle possibili equazioni è

$$I[7, 33, 34, 66, 68, 97, 100, 103] \bigoplus O[4] = K[16, 32, 96]$$

5.4 *L'approccio meet in the middle*

Per migliorare la ricerca dell'equazione migliore su 9 round abbiamo tentato un approccio del tipo meet in the middle. L'idea è che la propagazione dei bit a causa della permutazione tende, chiaramente, ad attivare sempre più colonne man mano che i round aumentano. Di conseguenza componendo due ricerche indipendenti che abbiano in comune un estremo della ricerca si riduce la propagazione dei bit attivi.
In particolare se stiamo lavorando su r round sceglieremo uno stato centrale $S_{\lfloor \frac{r}{2} \rfloor}$ con una sola colonna attiva. Successivamente determineremo la migliore equazione lineare che inizia con quello stato su $\frac{r}{2}$ round e la migliore equazione che finisce con quello stato su $\frac{r}{2}$ round. Per determinare la prima metà, dunque l'equazione che ha come output $S_{\lfloor \frac{r}{2} \rfloor}$, è sufficiente considerare il cifrario inverso di GIFT-128 e determinare la migliore equazione che inizia con

quello stato su quel cifrario. Con cifrario inverso si intende il cifrario composto dalla operazioni inverse di quelle standard applicate in ordine inverso. Questa tecnica, su 9 round, prevede dunque tre fasi

- Cercare le equazioni lineari migliori possibili sul cifrario a 5 round (che rappresentano i round 5, 6, 7, 8 e 9).
- Cercare le equazioni lineari migliori possibili sull'inverso del cifrario a 4 round (che rappresenteranno i round 1, 2, 3 e 4).
- "Saldare" le equazioni, dunque accodare le equazioni trovate e calcolare la probabilità complessiva con il Piling-Up Lemma.

Questo approccio ha consentito di trovare equazioni lineari su 3 round con probabilità $\frac{1}{2} - 2^{-6}$, su 5 round con probabilità $\frac{1}{2} - 2^{-11}$ e su 9 round con probabilità $\frac{1}{2} - 2^{-28}$.

Capitolo 6

Conclusioni

L'analisi di sicurezza del cifrario contro la crittoanalisi lineare effettuata dagli autori verte su due aspetti principali.
Innanzitutto la propagazione delle colonne attive dello stato nei trail, infatti come detto in precedenza mantenere questo valore basso garantisce una migliore efficienza dell'algoritmo di Matsui, in quanto poche colonne coinvolte equivale a poche S-Box coinvolte, che a sua volta equivale a poche equazioni lineari combinate per ottenere l'equazione lineare finale.
Sui 9 round il lower bound fornito dagli autori riguardo il numero totale di S-Box attive in almeno un round del trail è 18, risultato per il quale abbiamo trovato un'equazione esplicita con la tecnica del meet in the middle.
Un altro approccio tentato per controllare il numero di S-Box attive sfrutta il fatto che la funzione di permutazione contiene, su alcune righe, dei cicli di lunghezza non eccessivamente lunga, il che consente di confinare le colonne considerate nei trail di funzioni di parità a un sottoinsieme invariante. Il problema di questa tecnica è che non consente di restare in questi sottoinsiemi per troppi round poiché il vincolo di avere equazioni con probabilità diversa da $\frac{1}{2}$ fa sì che non si trovino trail ciclici. Comunque un approccio di questo tipo consente di limitare il numero di colonne attive su un numero di round ridotto e di trovare altre equazioni che eguagliano il lower bound degli autori sul numero di S-Box considerate.
Inoltre l'analisi di sicurezza degli autori ha determinato, su 9 round, un potenziale di correlazione medio pari a $2^{-45.99}$. L'equazione lineare migliore che abbiamo trovato su 9 round è verificata con probabilità $\frac{1}{2} - 2^{-28}$, il che corrisponde a un contributo al potenziale medio di correlazione di 2^{-54}.
Tramite lo stesso metodo siamo riusciti a trovare altre 3 equazioni lineari con lo stesso potenziale di correlazione, inoltre la -esima equazione migliore trovata ha un contributo pari a , possiamo quindi concludere che i nostri risultati sono in linea con quelli trovati dagli autori.

Riportiamo infine uno dei trail che raggiungono la massima correlazione trovata con questo metodo, usando la rappresentazione del capitolo 5, dunque come sequenza di funzioni di parità corrispondenti a vettori di lunghezza 128 (li rappresenteremo come lo stato del cifrario per chiarezza, dunque tabelle di 32 colonne da 4 elementi, al posto delle colonne inseriremo la rispettiva conversione in decimale in modo tale da poter effettuare facilmente il calcolo della correlazione facendo riferimento alla tabella 5.1)

$$\alpha_0 = (0,6,9,0,0,0,0,0,0,0,0,0,0,0,0,9,3,0,0,0,0,0,5,0,9,0,5,6,0,0,0,0,0,0)$$

$$\alpha_1 = (0,0,0,0,0,5,6,0,3,0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_2 = (0,6,9,0,0,0,0,0,0,0,0,0,0,0,0,0.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_3 = (0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_4 = (0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_5 = (0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_6 = (0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0.0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_7 = (0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0)$$

$$\alpha_8 = (0,0,8,0,0,0,0,8,0,4,0,0,0,0,0,4,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

$$\alpha_9 = (0,0,0,9,5,5,0,3,0,5,14,0,0,0,0,0,0,0,0,0,0,9,0,5,0,0,9,3,0,14,10,0)$$

Bibliografia

[1] Matsui M.: Linear Cryptanalysis Method of DES Cipher. Advances in Cryptology – Eurocrypt '93, Lecture Notes in Computer Science 765 386-397, 1993.

[2] Matsui M.: On correlation between the order of S-boxes and the strength of DES. Advances in Cryptology — EUROCRYPT'94. EUROCRYPT 1994, De Santis A., Ed., LNCS 950. Springer, 1995.

[3] Banik S., Pandey S., Peyrin T., Sasaki Y., Sim S. M., Todo Y.: GIFT: A Small Present Towards Reaching the Limit of Lightweight Encryption. Conference on Cryptographic Hardware and Embedded Systems (CHES), 10.1007/978-3-319-66787-4-16, 2017.

[4] Biham E., Shamir A.: Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag (1993).

[5] Wagner D.: The Boomerang Attack. Fast Software Encryption, L. R. Knudsen, Ed., LNCS 1636, pp. 156–170. Springer, 1999.

[6] Langford S.K., Hellman M.E.: Differential-Linear Cryptanalysis. Advances in Cryptology — CRYPTO '94. CRYPTO 1994, Y.G. Desmedt, Ed., LNCS 839. Springer, 1994.

[7] Daemen, J., Rijmen, V.: The Design of Rijndael. Springer-Verlag New York, Inc., 2002.

[8] Washington, L. C.: Introduction to Cryptography with Coding Theory. Pearson Education, Inc., 2002.

[9] Stinson D. R.: Cryptography Theory and Practice. Chapman & Hall, 2006.

[10] Nyberg: Linear Approximation of Block Ciphers. Advances in Cryptology, Proc. Eurocrypt '94, LNCS 950, A. De Santis, Ed., Springer-Verlag, 1995.

[11] Banik S., Chakraborti A., Iwata T., Minematsu K., Nandi M., Peyrin T., Sasaki Y., Sim S. M., Todo Y.: GIFT-COFB V 1.1. Submission to the Lightweigth Cryptography Standardization Process, 2021.

[12] Biham E., Shamir A.: Differential Cryptanalysis of DES-like Cryptosystems, Journal of Cryptology,Vol.4,pp.3-72, 1991.

[13] O'Connor L.: On the distribution of characteristics in bijective mappings, Journal of Cryptology, Vol. 8, No. 2, 1995.

[14] Biham, E.: On Matsui's Linear Cryptanalysis. Pre-proceedings of Eurocrypt'94, 1994.