UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Progettazione e Implementazione di un Sistema Bayesiano per la crittoanalisi differenziale del DES

Tesi di Laurea Magistrale in Ingegneria Informatica

L. Gagliano

Relatore: Prof. Giuseppe Lo Re
Correlatore: Ing. A. De Paola

Università degli Studi di Palermo

FACOLTÀ DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi Di Laurea

Progettazione e Implementazione di un Sistema Bayesiano per la crittoanalisi differenziale del DES.

Candidato:
Luca Gagliano

Relatore:
Prof.re Giuseppe Lo Re

Controrelatore:
Prof.re Salvatore Gaglio

Correlatore:
Ing. Alessandra De Paola

Anno Accademico 2011/2012

Sommario

Il lavoro di questa tesi nasce dall'idea di riproporre il metodo di crittoanalisi differenziale mediante l'uso di reti bayesiane.
Inizialmente è stata effettuata una ricerca sullo stato dell'arte per studiare diversi approcci in letteratura, con il fine principale di constatare se l'uso delle Reti Bayesiane era già stato pensato come metodo per poter perpretare un attacco al DES.
L'esito della ricerca è stato negativo, il che ha spinto a studiare inizialmente le proprietà statistiche delle S-Box del DES, in particolare riformulando la vulnerabilità trovata da Biham e Shamir mediante il formalismo delle Reti Bayesiane, il ragionamento, dopo opportuni accorgimenti, è stato poi esteso all'intera funzione di Feistel, e infine all'intero DES.
A seguito di questo studio teorico vi è la progettazione dei tools del sistema utilizzati per gli esperimenti, sia per attaccare le S-Box, sia per attaccare alcuni round iterati del DES.
I tools utilizzati per attaccare le S-Box implementano l'inferenza diagnostica, mentre l'attacco al DES, dopo uno studio di fattibilità, è stato implementato mediante campionamento di ipotesi dalla distribuzione indotta dalla Rete Bayesiana implementata.
Successivamente sono riportati i risultati sperimentali, finalizzati a individuare il numero di dati e il tempo necessario a portare a termine un'attacco alle S-Box e al DES, in particolare sull'attacco al DES è stato fatto anche uno studio sul numero di campioni ottimali, e sui valori di innesto all'algoritmo per poter portare ad una convergenza più veloce.
Infine vengono tratte le conclusioni sul percorso affrontato proponendo possibili sviluppi futuri.
Nelle appendici vengono riportati le tabelle che definiscono le S-Box del DES, le distribuzioni delle differenze input/output ricavate da Biham e Shamir, un metodo di risoluzione di sistemi lineari sotto opportune ipotesi e infine alcuni approfondimenti sulla funzione di espansione del DES.

Indice

Capitolo 1

Introduzione

1.1 Il DES

Il DES è un'algoritmo di cifratura simmetrico adottato come standard di cifratura nel 1977 dal National Bureau of Standards, divenuto poi National Institute of Standards and Technology (NIST) [15]. La cifratura avviene su blocchi di 64 bit usando una chiave a 56 bit, tale blocco viene trasformato mediante step consecutivi in un altro blocco di 64 bit, e l'output dipende dalla particolare chiave utilizzata. Le trasformazioni adottate si basano sul concetto di *Permutazione* e *Sostituzione* introdotto da Shannon in [16], la permutazione ha lo scopo di mutare le relazioni statistiche presenti nel testo in chiaro, distribuendo in modo più uniforme l'istogramma delle occorrenze nel testo cifrato, la sostituzione deve rendere difficile l'individuare la relazione presente tra il testo cifrato e la chiave. La figura 1.2 mostra come è strutturata la permutazione, dall'alto verso il basso si ha la composizione dei vari bit, dal più significativo al meno significativo, ogni valore nelle varie entry indica quale particolare bit dell'input si mappa in una particolare posizione. La figura 1.1, tratta da [19] mostra il funzionamento generale dell'algoritmo mediante uno schema a blocchi, che in realtà è una particolare istanza del cifrario di Feistel [9]. Come si vede un blocco di testo in chiaro di 64 bit viene sottoposto ad uno step iniziale di permutazione dei bit che compongono tale blocco, la figura 1.2 mostra la particolare permutazione applicata inizialmente. L'output di questa permutazione viene poi sottoposto a diverse funzioni di trasformazioni successive, applicate in serie, ovvero l'output della trasformazione i-esima è l'input della $i+1$-esima. Queste trasformazioni sono comunemente chiamate *round* e sono parametrizzate in base all'output dell'algoritmo di schedulazione della chiave. Dopo l'applicazione delle funzioni *round* si passerà ad uno swap dei 32 bit dell'output dell'ultimo round, ovvero se l'ultimo round produce come output il blocco di 64 bit (L, R), L e R sono blocchi a 32 bit, lo swap produrrà (R, L), al quale sarà applicata la permutazione inversa di quella rappresentata in figura 1.2.

1.2 Schedulazione della chiave

La chiave che parametrizza le trasformazioni del DES è formalmente di 64 bit, da essa però vengono eliminati i bit in posizione multipla di 8 che vengono però

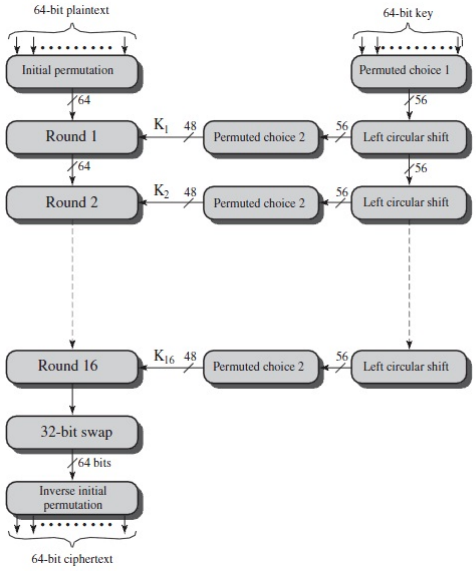


Figura 1.1: Struttura generale del DES

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Figura 1.2: Tabella di permutazione iniziale del DES [19]

utilizzati come bit di parità su blocchi di 7 bit. Detta $K \in \mathbb{Z}_2^{64}$ la chiave formale del DES, la prima trasformazione, lineare, farà ottenere una chiave S_K tale che questa sia ottenuta eliminando i bit in posizione $i * 8$, $i = 1, ..., 8$. Quindi la chiave reale del DES può spaziare all'interno di $\mathbb{Z}_2^{56}$. La schedulazione della chiave è un'algoritmo che permette di produrre, a partire dalla chiave di 56 bit, 16 sottochiavi di 48 bit i quali verranno poi utilizzati all'interno dei singoli round come verrà spiegato in seguito. La schedulazione della sottochiave può essere descritta mediante lo pseudocodice 1.1.

Nelle righe 2 e 3 si ha la separazione del blocco di 56 bit in ingresso in due blocchi di 28 bit, a ciascuno dei quali è applicato uno shift circolare di uno o due bit a sinistra, dipendentemente dal particolare round, i risultati di questi shift sono indicati in riga 4 e 5 come c_i e d_i. Concatenando c_i e d_i si ottiene l'input del successivo round di schedulazione, che verrà poi ritornato dall'algoritmo, mentre in riga 7 si ha la creazione di una sottochiave di 48 bit che verrà dato come input al round i-esimo. I 48 bit si sottochiave sono ottenuti mediante una fun-

Algoritmo 1.1 Fase generica di Schedulazione della chiave

function KEY_SCHEDULE(k_{i-1}, i)
    $c_{i-1} \leftarrow$ 28 bit più significativi di k_{i-1}
    $d_{i-1} \leftarrow$ 28 bit meno significativi di k_{i-1}
    $c_i \leftarrow Left\_Shift(c_{i-1}, i)$
    $d_i \leftarrow Left\_Shift(d_{i-1}, i)$
    $k_i \leftarrow c_i || d_i$
    $sk_i \leftarrow$ PC-2(k_i)
    return (k_i, sk_i)
end function

Algoritmo 1.2 Generico Round del DES

function ROUND_FUNCTION(p_i, sk_i)
    $l_{input} \leftarrow$ 32 bit più significativi di p_i
    $r_{input} \leftarrow$ 32 bit meno significativi di p_i
    $r_{output} \leftarrow l_{input} \oplus$ FEISTEL(r_{input}, sk_i)
    $l_{output} \leftarrow r_{input}$
    return $l_{output} || r_{output}$
end function

zione detta di *permutazione/compressione* sostanzialmente alcuni bit vengono permutati e altri eliminati, per un dettaglio maggiore di questa trasformazione si veda [19].

1.3 Operazione del Singolo Round

Si supponga di avere un blocco $P_i \in \mathbb{Z}_2^{64}$ e una sottochiave $S_K \in \mathbb{Z}_2^{48}$, il singolo Round del DES produce C_i secondo l'algoritmo 1.2. L'input P viene diviso in due parti di 32 bit, la parte meno significativa sono i 32 bit più significativi di C_i, mentre i 32 bit meno significativi sono ottenuti mediante una trasformazione $F(\cdot, \cdot)$ (funzione di $Feistel$ all'interno dell'algoritmo) che trasforma la parte più significativa di P_i in base alla sottochiave S_K.

La funzione di Feistel, che sarà studiata in dettaglio nei capitoli successivi, è una funzione non lineare rispetto alla chiave e non lineare rispetto all'input. Lo pseudocodice 1.3 riassume il suo funzionamento. Si tratta essenzialmente dell'applicazione di una *trasformazione affine* sull'input, seguita da una trasformazione non lineare e da un'ulteriore trasformazione lineare. La funzione di espansione replica alcuni bit del proprio input, la permutazione invece li per-

Algoritmo 1.3 Funzione di Feistel

function FEISTEL(z, sk_i)
    $z_i \leftarrow$ EXPANSION(z) $\oplus$ sk_i
    $z_o \leftarrow$ S-BOX(z_i)
    $y \leftarrow$ PERMUTATION(z_o)
    return y
end function

muta secondo una tabella precisa (si veda [15], [19] per le tabelle in merito). Anche le S-Box sono dei mapping tabellati, riportati in appendice A. Le S-Box in particolare sono stati oggetto di studi per effettuare una crittoanalisi efficace sul DES. Questi risultati, insieme ad una descrizione esauriente sul loro funzionamento, sono riportati nel secondo capitolo.

Capitolo 2

Le S-Box del DES

2.1 Funzionamento

In termini teorici si può affermare che il DES ha radici abbastanza profonde. Esso si può dire essere conseguenza quasi diretta della teoria della comunicazione formulata da Shannon [17] nel 1949. I risultati degli studi riportati in [17] costituiscono il fondamento della Teoria dell'informazione, gli strumenti di tale teoria verranno successivamente utilizzati in [16] per definire i concetti di *Diffusione* e *Confusione*, è stato poi Feistel nel 1973 a dare a questi concetti una connotazione pratica. L'obiettivo di questa tesi non è l'analisi del funzionamento del DES, sul quale esiste un'ampia letteratura [15], ma è di interesse approfondire il funzionamento delle S-Box, la loro importanza all'interno del sistema DES, e discutere i risultati di alcuni studi effettuati nel tentativo di violarle. Una S-Box è in generale una mappa

$$S(\cdot) \ : \mathbb{Z}_2^{48} \ \rightarrow \mathbb{Z}_2^{32}, \tag{2.1.1}$$

dove ogni elemento di input $S_{IX} \in \mathbb{Z}_2^{48}$ e di output $S_{OX} \in \mathbb{Z}_2^{32}$ sono della forma

$$S_{IX} = (x_1, x_2, \ldots, x_{48}), \tag{2.1.2}$$
$$S_{OX} = (y_1, y_2, \ldots, y_{32}). \tag{2.1.3}$$

Definendo

$$Si_{IX} = (x_{(i-1)*6+1}, x_{(i-1)*6+2}, \ldots, x_{(i-1)*6+6}), \tag{2.1.4}$$
$$Si_{OX} = (y_{(i-1)*4+1}, y_{(i-1)*4+2}, \ldots, y_{(i-1)*4+4}), \tag{2.1.5}$$

con $i = 1, \ldots, 8$, la (2.1.2) e (2.1.3) possono essere riscritte come

$$S_{IX} = (S1_{IX}, S2_{IX}, \ldots, S8_{IX}), \tag{2.1.6}$$
$$S_{OX} = (S1_{OX}, S2_{OX}, \ldots, S8_{OX}). \tag{2.1.7}$$

Ogni coppia (Si_{IX}, Si_{OX}) è tale che risulti

$$Si_{OX} = Si(Si_{IX}), \tag{2.1.8}$$

dove $Si(\cdot)$ è una mappa opportuna del tipo

$$Si(\cdot) \ : \ \mathbb{Z}_2^6 \rightarrow \mathbb{Z}_2^4. \tag{2.1.9}$$

Per definire le S-Box bisogna definire i vari mapping $Si(\cdot)$, e questi mapping sono definiti tramite delle opportune tabelle, come quella in tabella 2.1, le altre sono nell'appendice A. Questi mapping vanno interpretati nel modo segue, ricordando che ogni input delle $Si(\cdot)$ è una stringa di 6 bit, il bit più significativo, e il bit meno significativo formano una stringa di due bit che indicizza la riga, i 4 bit centrali invece indicizzano la colonna. Le Si_{OX} sono quindi i valori riportati in tali entry. Diversi sono stati i lavori mirati a studiare le proprietà delle S-Box, quali siano i criteri di progettazione che si attuano a queste e quali siano i possibili miglioramenti che si possono attuare a queste. Saranno descritti alcuni risultati che sottolineano questi aspetti, nella esposizione di questi sarà adottata la notazione utilizzata nei rispettivi riferimenti.

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	10

Tabella 2.1: Tabella per S1.

2.1.1 Gli studi di Brickell, Moore e Purtill

Gli autori di [3] indicano le S-Box come le componenti di maggior non linearità all'interno della funzione di Feistel istanziata dal DES. Gli studi esposti sottolineano alcuni principi di progettazione adottati nella implementazione delle S-Box. Riprendendo la notazione in [3] verrà indicato con $S(\cdot)$ una generica tabella di (2.1.9), mentre con + verrà indicato lo XOR bit a bit tra due stringhe di bit di medesima lunghezza. In [3] vengono elencate le seguenti proprietà che caratterizzano le S-Box:

P0 Ogni riga di una S-Box è una permutazione degli interi da 0 a 15;

P1 Le S-Box non sono trasformazioni lineari o affini del proprio input;

P2 Se due input distinti differiscono per un solo bit allora gli output differiranno per almeno due bit;

P3 $S(x)$ e $S(x + 001100)$ differiscono almeno per due bit;

P4 $S(x) \neq S(x + 11ef00)$ comunque siano fissati e ed f;

P5 Le S box sono scelte in modo tale da minimizzare la differenza tra il numero di 1 e 0 in qualunque output quando un singolo bit di input è mantenuto costante.

La proprietà P5 all'apparenza difficile da capire sostanzialmente asserisce che se si prende un generico output delle S-Box, e si fissa un bit nell'input si ha che per tale output il numero di bit posti a 0 e il numero di bit posti a 1 sono più o meno uguali, la differenza è minima per costruzione delle S-Box. Gli autori del documento hanno generato inoltre delle S-Box random, rispettanti non tutte ma alcune delle proprietà elencate. In base ai loro studi essi affermano che le S-Box godono della proprietà

P1' . I singoli bit ottenuti come output dalle S-Box non sono ottenibili per somma di 3 o 4 bit utilizzati per indicizzare la colonna della tabella che la definisce.

La proprietà **P1**' stabilisce che non si possono usare certe classi di funzioni per approssimare il funzionamento delle S-Box.

2.1.2 I criteri di progettazione di Dawson e Tavares

Dawson e Tavares [8] hanno suggerito di studiare le S-Box secondo due aspetti, il primo aspetto vede le S-Box come entità *statiche* (figura 2.1) ovvero dato un certo input **X** si otterra un certo output **Y**; di conseguenza idealmente le S-Box, da un punto di vista statico, non dovrebbero permettere di inferire l'input oppure un output se si possiedono informazioni parziali, ad esempio coppie input/output precedenti. Il secondo aspetto è quello *dinamico* (figura 2.2), ovvero deve rimanere incerto come varierà l'input o l'output anche se si hanno informazioni parziali su variazioni precedenti. Questi criteri *ideali* sono stati esposti in modo formale avvalendosi della definizione di *Entropia* data da Shannon [17] per la teoria della comunicazione. In base a queste proprietà ideali Dawson e Tavares hanno definito dei criteri di progettazione ben precisi, suddividendoli in statici e dinamici.

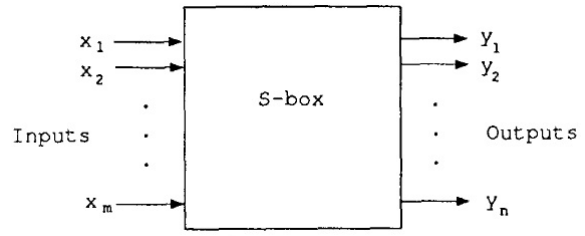


Figura 2.1: Visione Statica delle S-Box

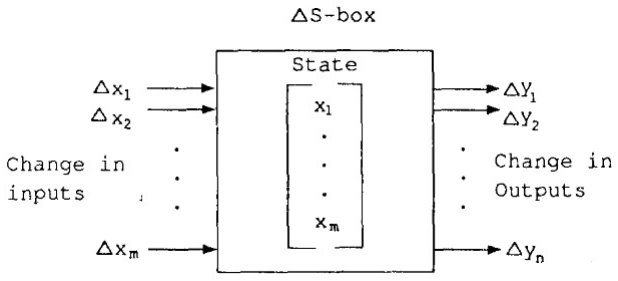


Figura 2.2: Visione Dinamica delle S-Box

Proprietà statiche. Come detto in precedenza le proprietà statiche analizzano le relazioni input output, non tenendo conto di come questi variano rispetto allo stato precedente. Un primo criterio di progettazione che dovrebbe essere

adottato è quello di **Indipendenza Input Output**, secondo tale criterio la S-Box va scelta in modo tale che

$$P(y_j|a_1x_1, \ldots, a_mx_m) = P(y_j), \tag{2.1.10}$$

dove $m < j$ e $a_k \in \{0, 1\}$, e indicano se il rispettivo x_k sia noto o meno, diremo che il criterio di indipendenza Input Output è di ordine r se $\sum_{k=1}^{m} a_k = r$, in breve sono noti r ingressi. Un criterio simile a quello appena esposto è quello di **Indipendenza Output Input**, che è analogo al precedente, ma semplicemente si invertono ingressi con uscita. Il criterio di **Indipendenza Output Output**, indica che la conoscenza di uscite precedenti non influenza l'incertezza sull'uscita attuale, questo criterio può essere espresso come

$$P(y_j|a_1y_1, \ldots, a_ny_n) = P(y_j), \tag{2.1.11}$$

dove $n < j$ e anche in questo caso si può parlare di *ordine di indipendenza*. Ulteriore criterio statico è la **Non-Linearità**, cioè il funzionamento di una S-Box non può essere espresso in termini di un sistema di equazioni lineari. Come ulteriore proprietà statica si ha la **Completezza di Informazione**, che asserisce che ogni bit dell'output deve essere funzione di tutti i bit dell'input. Infine la proprietà di **Invertibilità**, che deve essere valida per le S-Box di dimensione quadrata.

Proprietà dinamiche. Le proprietà dinamiche sono sostanzialmente tre

- **Indipendenza Dinamica Input Output**;
- **Indipendenza Dinamica Output input**;
- **Indipendenza Dinamica Output Output**.

Queste tre proprietà sono una estensione delle precedenti equivalenti statiche, ad esempio la prima di queste può essere espressa come

$$P(\Delta y_j|a_1\Delta x_1, \ldots, a_n\Delta x_m) = P(\Delta y_j), \tag{2.1.12}$$

dove al solito $m < j$ ed è possibile fissare un ordine di validità di tale proprietà. La non linearità vale automaticamente se vale nel caso statico. Le figure 2.1 e 2.2 esplicitano la differenza tra la visione statica e dinamica appena suggerita, la visione statica fa si che l'output attuale dipenda direttamente e soltanto dall'input attuale. La visione dinamica fa si che la differenza tra due output successivi dipenda dalla differenza dalla differenza tra due input successivi e da uno stato attuale, che sarebbe l'input attuale.

2.1.3 Equazioni di Matsui

In questa sezione indichiamo con *Equazioni di Matsui* il metodo utilizzato da Mitsuru Matsui per approssimare il comportamento delle S-Box nel suo metodo di *Crittoanalisi Lineare* [14], in questa tesi il lavoro di Matsui costituisce un primo esempio di approssimazione delle S-Box a scopo di crittoanalisi, i lavori esposti nelle due sezioni precedenti rappresentavano dei risultati nell'ambito

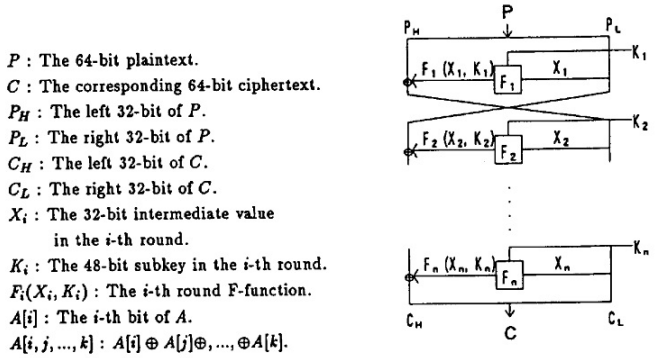


Figura 2.3: Notazione di Matsui [14].

dello studio delle S-Box, e della definizione di criteri di progettazione di cifrari *DES-Like*. In figura 2.3 è indicata la notazione utilizzata da Matsui per l'esposizione del suo lavoro.

Lo scopo della crittoanalisi lineare è di trovare una espressione della forma:

$$P\left[i_1, i_2, \ldots, i_a\right] \oplus C\left[j_1, j_2, \ldots, j_b\right] = K\left[k_1, k_2, \ldots, k_c\right], \tag{2.1.13}$$

che sia valida per una probabilità $p \neq 0.5$. Le equazioni di Matsui per le S-Box sono ricavate a partire dalla seguente definizione

Definizione 2.1. *Per una data S-Box* S_a $(a = 1, 2, \ldots, 8)$*,* $1 \leq \alpha \leq 63$ *e* $1 \leq \beta \leq 15$*, definiamo* $NS_a(\alpha, \beta)$ *come il numero di input a 64 bit di* S_a *tale che lo XOR dei bit di tali input mascherati con una maschera alpha coincida con lo XOR dei bit dell'output mascherato con una maschera* β*. Ovvero*

$$NS_a(\alpha, \beta) = \sharp\left\{x \mid 0 \leq x < 64,\ \left(\oplus_{s=0}^{5}\left(x[s] \cdot \alpha[s]\right)\right) = \left(\oplus_{t=0}^{3}\left(S_a(x)[t] \cdot \beta[t]\right)\right)\right\}, \tag{2.1.14}$$

dove $\cdot$ *indica l'operazione di AND tra due bit e* $\oplus_{i=1}^{n} b[i] = b[1] \oplus \ldots \oplus b[n]$.

Si osservi nella (2.1.14) che l'espressione

$$\left(\oplus_{s=0}^{5}\left(x[s] \cdot \alpha[s]\right)\right) = \left(\oplus_{t=0}^{3}\left(S_a(x)[t] \cdot \beta[t]\right)\right) \tag{2.1.15}$$

è lineare, quindi la (2.1.14) conteggia il numero di input tali che rispetto a qualche criterio, dettato dalle maschere α e β, valgano delle equazioni lineari. In breve il rapporto

$$p = \frac{NS_a(\alpha, \beta)}{64} \tag{2.1.16}$$

esprime la probabilità che valga un determinato vincolo lineare identificato dalle maschere α e β. Se $p = 1$ allora il vincolo lineare è sempre valido, mentre se $p \neq 1$ allora quel vincolo lineare non è sempre valido, ma potrebbe esserlo per la maggior parte degli ingressi delle S-Box. Nel lavoro di Matsui le equazioni vengono prese in considerazione quando esse valgono con una probabilità $p \neq$

0.5, in quanto essi dimostrano che l'efficacia di uno dei vincoli lineari del tipo (2.1.15) è misurabile come

$$\eta = |p - 0.5|,$$

da tale misura segue infatti che se $p = 0.5$ allora l'efficacia di quel vincolo a scopo di crittoanalisi è nulla. Una volta determinati vincoli validi, si ha che per la generica S-Box vale un sistema del tipo

$$\begin{cases} \alpha[0]_0 \cdot x[0] \oplus \ldots \oplus \alpha[5]_0 \cdot x[5] = \beta[0]_0 \cdot S_a(x)[0] \oplus \ldots \oplus \beta[3]_0 \cdot S_a(x)[3] \\ \alpha[0]_1 \cdot x[0] \oplus \ldots \oplus \alpha[5]_1 \cdot x[5] = \beta[0]_1 \cdot S_a(x)[0] \oplus \ldots \oplus \beta[3]_1 \cdot S_a(x)[3] \\ \vdots \\ \alpha[0]_k \cdot x[0] \oplus \ldots \oplus \alpha[5]_k \cdot x[5] = \beta[0]_k \cdot S_a(x)[0] \oplus \ldots \oplus \beta[3]_k \cdot S_a(x)[3] \end{cases}.$$

Nel sistema precedente si ha che $x[i] = k[i] \oplus e[i]$ dove $k[i]$ è l'i-esimo bit della chiave, mentre $e[i]$ è l'i-esimo bit di espansione, per la S-Box S_a, siccome l'attacco che si attuarebbe è di tipo choosen plaintext $e[i]$ è noto. Quindi il sistema precedente diviene della forma

$$Ak = b, \tag{2.1.17}$$

dove

$$A = \begin{pmatrix} \alpha[0]_0 & \alpha[1]_0 & \ldots & \alpha[5]_0 \\ \alpha[0]_1 & \alpha[1]_1 & \ldots & \alpha[5]_1 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha[0]_k & \alpha[1]_k & \ldots & \alpha[5]_k \end{pmatrix}, \tag{2.1.18}$$

mentre k è il vettore di bit della sottochiave mentre b è un opportuno vettore di termini noti. Il sistema in questo caso il sistema fornirà per soluzione la chiave che con probabilità p sarà corretta.

2.1.4 Equazioni di Courtois

Nella sezione precedente abbiamo visto un primo esempio di descrizione delle S-Box mediante un set di equazioni *lineari* ma valide con una certa probabilità. Courtois [5] ha condotto invece diversi studi sull'uso di metodi algebrici, esatti non probabilistici, per l'approssimazione delle S-Box e per la crittoanalisi complessiva del DES. L'approccio si fonda sull'affermazione di Shannon, secondo la quale la rottura di un cifrario dovrebbe richiedere la risoluzione di grandi sistemi di equazioni. I lavori di Courtois sono iniziati con l'AES, fondato sulla teoria dei campi finiti e l'aritmetica polinomiale, successivamente si è spostato verso il DES poichè da un punto di vista pratico non è mai stato violato. La differenza del DES rispetto all'AES sta nel fatto che non è basato su metodi algebrici, di conseguenza è particolarmente sentita la difficoltà del trovare un sistema di equazioni che descrivano le S-Box. In [5] vengono date delle nozioni base, che partono con la seguente definizione

Definizione 2.2 (Grado I/O)**.** . *Data una funzione $f : GF(2)^n \to GF(2)^m$, $f(x) = y$, con $x = (x_0, \ldots, x_{n-1})$, $y = (y_0, \ldots, y_{m-1})$. Il **grado I/O** di f è il più piccolo grado di una relazione algebrica*

$$g(x_0, \ldots, x_{n-1}; y_0, \ldots, y_{m-1}) = 0, \tag{2.1.19}$$

che rimane valida per ogni coppia $(x; y)$ tale che $y = f(x)$.

Il numero minimo, e molto spesso il numero esatto, di equazioni di un certo tipo che esistono per una S-Box possono essere ottenute applicando il seguente teorema.

Teorema 2.1. *Per ogni S-Box di dimensione $n \times m$, $F : (x_1, \ldots, x_n) \to (y_1, \ldots, y_m)$ e per qualunque sottoinsieme T di t monomi in x_i e y_j, rispetto a tutti i possibili 2^{m+n}, se $t > n$, ci sono almeno $t - 2^n$ equazioni di Input Output linearmente indipendenti (relazioni algebriche), che coinvolgono solo i monomi in T, e sono valide con probabilità 1, cioè per ogni (x, y) tali che $y = F(x)$.*

In questo caso rispetto al metodo di Matsui esposto nella sezione precedente non si può descrivere la S-Box con un sistema lineare, ma con un sistema di equazioni che possono essere anche di quarto grado. Il vantaggio di questo metodo risiede nel fatto che esso è esatto, lo svantaggio è che esso in generale richiede l'uso risolutori SAT (soddisfacibilità di fbf). La costruzione dei vincoli è differente da quella di Matsui, prendendo come spunto [5] si ipotizzi di avere un vincolo della forma

$$\sum \alpha_{ijk} x_i y_j y_k + \sum \beta_{ijk} x_i x_j y_k + \sum \gamma_{ij} x_i y_j + \sum \delta_i x_i + \sum \epsilon_i y_i + \eta = 0. \tag{2.1.20}$$

Ipotizzando il vincolo (2.1.20) sia valido allora per ogni

$$(x_0, \ldots, x_5; y_0, \ldots, y_3) = (x_0, \ldots, x_5; S(x_0, \ldots, x_5))$$

la (2.1.20) sarà verificata. Si dimostra, avvalendosi del Teorema 2.1, che il numero di tali vincoli è almeno 67; quindi da questi vincoli si hanno delle relazioni I/O. In un ambito di choosen plaintext (per esempio) vuol dire che è noto l'output della funzione di espansione, per una S-Box sono sei bit, quindi $(e_0, \ldots, e_5)$, osserviamo l'output della S-Box. Dal vincolo (2.1.20) si ricavano gli ingressi che soddisfano il vincolo, e si stimano le chiavi. La ricerca degli ingressi alle S-Box è rinconducibile ad un problema di soddisfaciblità polinomiale, da qui deriva l'uso dei risolutori SAT. Chiaramente più piccolo è il grado più veloce sarà la risoluzione.

2.1.5 S-Box accoppiate

Dagli studi di [3] segue che data una S-Box $Si(\cdot)$ gli output di questa sono uniformente distribuiti in $\mathbb{Z}_2^4$, visto che ogni riga di una S-Box è una permutazione dei valori $0, \ldots, 15$. In ogni caso, a causa della funzione di espansione del DES, gli input di S-Box vicine sono collegati da certi bit. Per esempio, nel primo round, lo XOR del bit di input in posizione 5 (contando da sinistra a destra) di $S1(\cdot)$ e del bit di input in posizione 1 di $S2(\cdot)$ è lo stesso dello XOR tra il bit in posizione 33 e il bit in posizione 49 della chiave (questo risultato segue dalla costruzione dell'algoritmo di schedulazione della chiave, [7]). Analogamente lo XOR dei bit di input in posizione 6 per $S1(\cdot)$ e del bit di input in posizione 2 per $S2(\cdot)$ è lo stesso tra i bit in posizione 17 e 57 della chiave. Queste osservazioni sono state riportate dagli autori di [7] per formalizzare un attacco alle S-Box, infatti lo XOR tra alcuni bit di input di S-Box adiacenti vincolano lo XOR di alcuni bit della chiave di 56 bit e inoltre la distribuzione dei bit di output, data questa informazione su alcuni bit di input, non è uniforme. In [7]

si indica un numero binario a n bit con W, e con W_i l'i-esimo bit da sinistra, ovvero l'i-esimo bit più significativo. Fissata una chiave siano $I, J \in \mathbb{Z}_2^6$ due input per due S-Box adiacenti, indicate con $Sp(\cdot)$ e $Sq(\cdot)$, e si indichi inoltre $X = Sp(I), Y = Sq(J) \in \mathbb{Z}_2^4$ gli output di tali S-Box. Si definisce l'insieme

$$S_{X,Y}(s,t) = \sharp\left\{I, J \in \mathbb{Z}_2^6 \mid I_5 \oplus J_1 = s, I_6 \oplus J_2 = t, Sp(I) = X, Sq(J) = Y\right\}, \tag{2.1.21}$$

che sarebbe il numero di coppie $(I, J) \in \mathbb{Z}_2^6 \times \mathbb{Z}_2^6$ tali che sistema

$$\begin{cases} I_5 \oplus J_1 = s \\ I_6 \oplus J_2 = t \\ Sp(I) = X \\ Sp(J) = Y \end{cases}$$

sia verificato. Sicuramente si avrà $S_{X,Y}(s,t) \leq 2^{12}$, in quanto 2^{12} sono le possibili coppie (I, J) sulle quali controllare la validità della condizione in (2.1.21). Il rapporto

$$p = \frac{S_{X,Y}(s,t)}{2^{12}} \tag{2.1.22}$$

definisce una densità di probabilità congiunta degli output X, Y condizionato al vincolo di alcuni bit di input. La distribuzione (2.1.22), che in [7] viene dimostrata sperimentalmente essere non uniforme, è il punto iniziale per il metodo di crittoanalisi proposto. La formalizzazione dell'attacco coinvolge strumenti quali la trasformata Walsh e la teoria della probabilità, e per motivi di spazio non viene trattata in dettaglio.

2.1.6 Crittoanalisi differenziale

Come ultimo esempio, che però vista la sua importanza per questa tesi sarà trattato in modo più approfondito nel prossimo capitolo, si accenna la vulnerabilità trovata da Biham e Shamir, descritta in [2], delle S-Box. Con riferimento alla terminologia utilizzata da Dawson e Tavares si può dire che la vulnerabilità trovata da Biham e Shamir è stata una vulnerabilità sulle proprietà dinamiche delle S-Box. Come già detto in precedenza, e ribadito anche in [3], ogni riga delle S-Box contiene una permutazione degli interi fra 0 e 15; questo implica che la distribuzione degli output è uniforme, inoltre le S-Box sono per loro costruzione fortemente non lineari, e avere informazioni su uno o più input e uno o più output nulla ci dice sull'output successivo, quindi queste sono *staticamente forti*. Biham e Shamir invece hanno osservato che esiste una distribuzione statistica tra differenze degli input e differenze degli output fortemente non uniformi, questa vulnerabilità delle S-Box è alla base di tutta la crittoanalisi differenziale. L'attacco è di tipo chosen plaintext, per ogni coppia di input X, X^* si osservano gli output $Y = S(X), Y^* = S(X^*)$, e si osserva che la probabilità $p(\Delta Y|\Delta X)$ non è uniforme, di volta in volta l'attacco stima un insieme di chiavi, da intersecare con quello precedentemente, questo fa si che con poche iterazioni si riesce a violare la S-Box.

2.2 Sommario sullo stato dell'arte

In questo capitolo sono stati esposti diversi lavori che evidenziano come le S-Box del DES siano state negli anni oggetto di studio per elaborare delle tecniche

efficienti di attacco al DES nel suo stato attuale, ma anche per definire in modo appropriato dei criteri di progettazione. I lavori di Brickel evidenziano le proprietà originali delle S-Box del DES, i lavori Dawson rappresentano un contributo nella definizione di criteri generali di progettazione delle S-Box, Matsui ha invece determinato un metodo probabilistico per la linearizzazione dei mapping, Courtois ne ha elaborato uno deterministico utilizzabile però con risolutori SAT, Davies ha studiato le distribuzioni congiunte delle S-Box, infine Biham e Shamir hanno sfruttato una vulnerabilità del DES. Esistono altri lavori che in questo capitolo non sono stati menzionati, ad esempio esistono, vedasi [4], tecniche di *improvement* delle S-Box base su DFT (Discrete Fourier Transform), così come tecniche di attacco, altre tecniche che sono miste delle precedenti (Differential Power Analisys in [12]). Come già detto però lo scopo di questo capitolo non è esporre gli attacchi in dettaglio, ma solo evidenziare come nel tempo si sia evoluto lo sviluppo della ricerca nel determinare altre vulnerabilità delle S-Box non note in precedenza. Inoltre è stato introdotto il principio su cui si fonda la crittoanalisi differenziale, oggetto del prossimo capitolo. La crittoanalisi differenziale in questa tesi è stata rielaborata mediante un approccio basato su Reti Bayesiane. L'uso di Reti Bayesiane permette di elaborare un modello matematico caratterizzante l'attacco, tramite il quale si possono dimostrare diverse proprietà utili all'implementazione del sistema Bayesiano per attaccare il DES.

Capitolo 3

La Crittoanalisi Differenziale

3.1 Introduzione alla crittoanalisi differenziale

La crittoanalisi differenziale [2] è un metodo che analizza gli effetti tra coppie di testo in chiaro sulle differenze tra le risultanti coppie di testo cifrato. Queste differenze possono essere utilizzate per assegnare delle probabilità alle possibili chiavi, e di conseguenza permette di restringere lo spazio di ricerca. Solitamente il metodo lavora su molte coppie di testo in chiaro, la cui differenza è assegnata, usando la differenza dei rispettivi testi cifrati. In sistemi DES-like la differenza scelta è lo XOR dei due testi in chiaro. In questa introduzione viene mostrato come in [2] l'analisi delle differenze permette di perpretare un attacco alle S-Box.

Di seguito viene esposta la notazione originale utilizzata da Biham e Shamir

n_x: Un numero esadecimale è indicato con un pedice x, per esempio $10_x = 16$;

X^*, X: Ad ogni punto intermedio durante la cifratura di coppie di messaggi, X e X^* sono i rispettivi valori delle due esecuzioni dell'algoritmo, mentre $X' = X \oplus X^*$;

$P(X)$: L'applicazione della funzione di permutazione $P(\cdot)$ all'input X. Si noti che P indica un generico testo in chiaro;

$E(X)$: Applicazione della funzione di espansione $E(\cdot)$ all'input X;

$IP(X)$: Applicazione della permutazione iniziale $IP(\cdot)$ all'input X;

P: Testo in chiaro;

T: Testo cifrato ottenuto da P (prima della permutazione inversa $IP^{-1}(\cdot)$. T^* Sarebbe un'altro test cifrato, mentre $T' = T \oplus T^*$;

(L, R): Parte sinitra e destra, di 32 bit, di P;

(l, r): Parte destra e sinistra di 32 bit, di T;

$a, \ldots, j$: 32 bit di input ad ogni applicazione della funzione di Feistel F;

$A, \ldots, J$**:** 32 bit di output della funzione di Feistel F;

$Si(\cdot)$**:** S-Box i-esima;

$Si_{EX}, Si_{KX}, Si_{IX}, Si_{OX}$**:** . Rispettivamente sono, gli output della funzione di espansione, la sottochiave, lo xor tra output della funzione di espansione e sottochiave, output della S-Box, i-esima.

La figura 3.1 mostra l'applicazione della notazione al DES ridotto a 8 rounds.

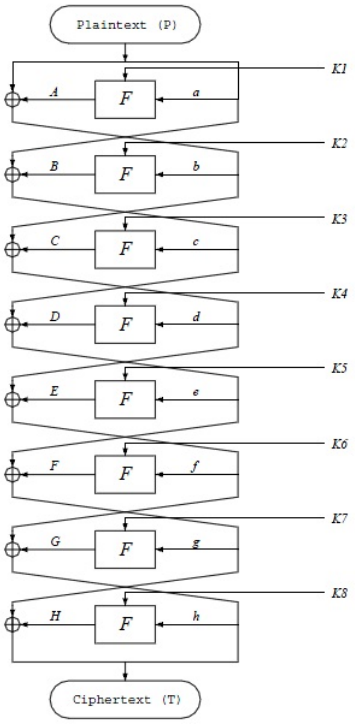


Figura 3.1: Notazione di Biham e Shamir su DES a 8 Rounds [2].

L'ipotesi fatta nella crittoanalisi differenziale è che le sottochiavi siano indipendenti, ovvero:

Definizione 3.1. *Una* chiave indipendente *è una lista di n sottochiavi che non sono necessariamente derivate dall'algoritmo di schedulazione.*

sostanzialmente una *chiave indipendente* è un insieme di sottochiavi che non si influenzano a vicenda. Risulta banale vedere che vale il seguente teorema:

Teorema 3.1. *Se vale l'ipotesi di indipendenza tra le sottochiavi il numero possibile di chiavi indipendenti del DES è* 2^{768}.

Dimostrazione. Ogni sottochiave è di 48 bit, sotto l'ipotesi di indipendenza, il numero totale di bit necessario a caratterizzare le particolari chiavi indipendenti sono 16×48, ovvero si ha un gruppo di 48 bit per ogni round, il numero possibili di chiavi indipendenti è quindi $2^{16\times 48} = 2^{768}$. □

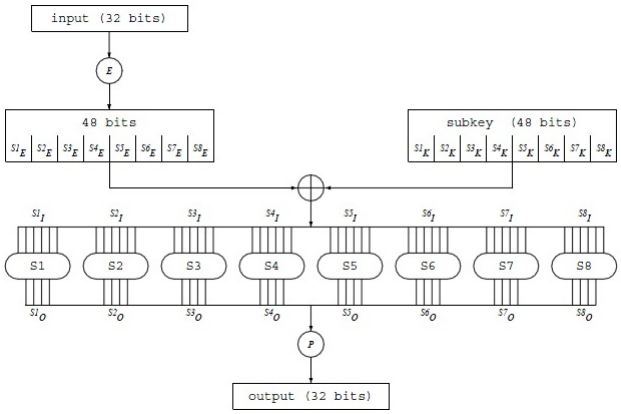


Figura 3.2: Schema a Blocchi della funzione di Feistel [2].

Nell'ambito della crittoanalisi differenziale si fa sempre l'assunzione di indipendenza, questo per semplificarne il trattamento teorico. Risulta utile richiamare il funzionamento della funzione di Feistel istanziata per il DES $F(\cdot,\cdot)$, questa è un mapping del tipo:

$$F(\cdot,\cdot) : \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{32} \tag{3.1.1}$$

il cui funzionamento è riassunto in figura 3.2.

La funzione di Feistel opera su un blocco di 32 bit, e l'output dipende da un parametro, la sottochiave K, a 48 bit. Ipotizzando di fissare K il primo passo è quello di applicare sull'input $X \in \mathbb{Z}_2^{32}$ la funzione di espansione

$$E : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{48}, \tag{3.1.2}$$

e risulta quindi, se $X \in \mathbb{Z}_2^{32}$, che $E(X) = X_E$, con $X_E \in \mathbb{Z}_2^{48}$.
Successivamente si calcola l'input $X_I \in Z_2^{48}$ della $S(\cdot)$, l'input è definito come

$$X_I = X_E \oplus K. \tag{3.1.3}$$

Dalla (3.1.3) segue che presi X_E, X_E^* si ha

$$X_E \oplus X_E^* = X_E \oplus (K \oplus K) \oplus X_E^* = (X_E \oplus K) \oplus (X_E^* \oplus K) = X_I \oplus X_I^* \tag{3.1.4}$$

ovvero lo XOR degli output della funzione di espansione è il medesimo degli input delle S-Box, ed è quindi indipendente dalla chiave.
Il passo successivo ancora è il calcolo di

$$X_O = S(X_I), \tag{3.1.5}$$

dove $X_O \in \mathbb{Z}_2^{32}$.
L'ultimo passo è quello di calcolare, mediante una permutazione dei bit di X_O, tramite una opportuna funzione di permutazione $P(\cdot)$, l'output Y della funzione

di Feistel. Quindi la funzione di Feistel può essere così riassunta

$$\begin{cases} F(X,K) = Y \\ Y = P(X_O) \\ X_O = S(X_I) \\ X_I = X_E \oplus K \\ X_E = E(X) \end{cases} .$$

3.1.1 Proprietà della funzione di Feistel

Supponiamo siano dati $X' = X \oplus X^*$, avvalendoci della definizione di funzione espansione si vede che

$$E(X') = E(X \oplus X^*) = E(X) \oplus E(X^*), \tag{3.1.6}$$
$$P(X') = P(X \oplus X^*) = P(X) \oplus P(X^*); \tag{3.1.7}$$

ovvero all'interno di F i mapping $E(\cdot)$ e $P(\cdot)$ sono lineari. Come detto nel capitolo precedente molti lavori, ad esempio quelli di [3], sono stati mirati a studiare le proprietà delle S-Box, tra queste si ricorda che non sono funzioni lineari o affini dei loro input, di conseguenza se queste vengono violate tutto il singolo Round sarebbe facilmente crittoanalizzabile.

3.2 Vulnerabilità delle S-Box

Nel capitolo dedicato allo stato dell'arte sono stati evidenziati alcuni lavori sulle S-Box, poco spazio è stato dedicato alla crittoanalisi differenziale, questo perchè si preferisce trattare la sua esposizione originale in modo sufficientemente approfondito, per comprendere poi la base del lavoro di questa tesi. La vulnerabilità notata da Biham e Shamir è stata derivata dall'analisi da tabelle simili a quella mostrata in figura 3.3. Nel DES ogni S-Box ha un $64 \cdot 64$ possibili coppie di input, ma ci sono solo $16 \cdot 16$ possibili coppie di output. Quello che si osserva è che se viene dato un valore a sei bit, che è lo XOR tra due input alle S-Box, statisticamente è più probabile che gli output abbiano determinate differenze piuttosto che altre. Questa osservazione informale, viene formalizzata da Biham e Shamir mediante due definizioni.

Definizione 3.2. *Una tabella che mostra la distribuzione degli input XOR e degli output XOR di tutte le possibili coppie di una S-Box è chiamata* ***tabella di distribuzione delle coppie in XOR della S-Box****. In questa tabella ogni riga corrisponde ad un particolare input XOR mentre ogni colonna corrisponde ad un particolare output XOR.*

La tabella della definizione appena data fornisce informazioni *quantitative*; per poter sfruttare queste informazioni serve un ulteriore definizione.

Definizione 3.3. *Sia X' un valore a sei bit e sia Y' un valore a quattro bit. Diremo che X' può causare Y' se esiste almeno una coppia X, X^* tali che $X \oplus X^* = X'$ e $S(X) \oplus S(X^*) = Y'$. Se una tale coppia esiste la* ***may cause*** *verrà indicata con $X' \to Y'$.*

Input XOR	Output XOR															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
									⋮							
30_x	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31_x	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32_x	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33_x	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34_x	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
35_x	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36_x	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37_x	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Figura 3.3: Tabella di distribuzione delle coppie in XOR della S1

Infine per dare una misura alla *may cause* bisogna metterla in relazione con la teoria della probabilità:

Definizione 3.4. *Diremo che* $X' \rightarrow Y'$ *con probabilità* p *se esistono* $64 * p$ *coppie di input alle S-Box ove l'input XOR è pari a* X' *e l'output XOR sia pari a* Y'*.*

3.3 Attacco alle S-Box

Nella sezione precedente sono state riportate le definizioni date in [2] per formalizzare le vulnerabilità delle S-Box, di seguito si commenta un possibile attacco a queste. Vista la linearità di $E(\cdot)$ e $P(\cdot)$ è possibile trascurare l'effetto di queste e considerare direttamente le S-Box. Ovvero per perpetrare un attacco sarà considerato lo schema in figura 3.4, in questo schema con Si_{EX} è l'output della funzione di espansione, mentre con $Si_{IX} = Si_{EX} \oplus Si_{KX}$ è indicato l'input alla S-Box i-esima, con $S(\cdot)$ si indica il mapping della S-Box mentre con $Si_{OX} = Si(Si_{IX})$. Detto questo l'attacco si attua nel modo seguente, come primo passo si fissa un valore a sei bit, Si'_{IX} come input XOR alla $Si(\cdot)$, ovvero un valore per il quale esistono due ingressi Si_{IX} ed Si^*_{IX} tali che $Si_{IX} \oplus Si^*_{IX} = Si'_{IX}$. Tuttavia vale che

$$Si'_{IX} = Si_{IX} \oplus Si^*_{IX} = (Si_{EX} \oplus Si_{KX}) \oplus (Si^*_{EX} \oplus Si_{KX}) = Si_{EX} \oplus Si^*_{EX} \quad (3.3.1)$$

quindi sicuramente esistono due output della funzione di espansione tali che il loro XOR sia esattamente uguale a Si'_{IX}. In un attacco di tipo choosen plaintext la coppia (Si_{EX}, Si^*_{EX}) può essere fissata. Lo scopo dell'attacco è trovare un

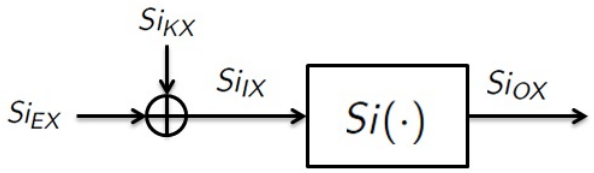


Figura 3.4: Schema a blocchi per l'attacco ad una S-Box.

insieme di coppie (Si_{IX}, Si^*_{IX}) tali che sia possibile stimare un insieme di chiavi. Supponendo l'attacco porti a trovare (Si_{IX}, Si^*_{IX}) una chiave candidata la si può ottenere dalla soluzione del sistema

$$\begin{cases} Si_{EX} \oplus Si_{KX} = Si_{IX} \\ Si^*_{EX} \oplus Si_{KX} = Si^*_{IX} \end{cases};$$

se il sistema ammette soluzione allora la Si_{KX} è una possibile chiave candidata. Il problema quindi è ricondotto alla ricerca della coppia (Si_{IX}, Si^*_{IX}) corrispondente alla coppia $(Si_{EX_1}, Si^*_{EX_1})$. Per poterla trovare si sfruttano le tabelle di distribuzione delle coppie di quella data S-Box. Siccome l'attacco è di tipo chosen plaintext se si da in input un Si_{EX_1} si otterrà il corrispettivo Si_{OX}, in modo analogo a $Si^*_{EX_1}$ corrisponderà Si^*_{OX}, allora all'interno della tabella di distribuzione possiamo ottenere il valore della entry in corrispondenza degli indici $[Si_{IX} \oplus Si^*_{IX}, Si_{OX} \oplus Si^*_{OX}]$, detto N il valore tale entry per definizione delle tabelle di distribuzione questo valore fornisce il numero di coppie (Si_{IX}, Si^*_{IX}) che rispettano il vincolo. Indicando queste coppie con $(Si_{IX_1}, Si^*_{IX_1}), \ldots, (Si_{IX_N}, S^*_{IX_N})$ si consideri il sistema

$$\begin{cases} Si_{EX_1} \oplus Si_{KX_1} = Si_{IX_1} \\ Si^*_{EX_1} \oplus Si_{KX_1} = Si^*_{IX_1} \\ \vdots \\ Si_{EX_1} \oplus Si_{KX_N} = Si_{IX_N} \\ Si^*_{EX_1} \oplus Si_{KX_N} = Si^*_{IX_N} \end{cases},$$

che fornirà, rispetto alle incognite $Si_{KX_1}, ..., Si_{KX_N}$, un insieme di soluzioni S_1. L'insieme S_1 è un sottoinsieme di tutte le possibili sottochiavi, all'interno delle quali sicuramente risiede la vera sottochiave utilizzata. Il risultato è stato quindi quello di ridurre lo spazio delle chiavi. Questa procedura la si può ripetere con una nuova coppia $Si_{EX_2}, Si^*_{EX_2}$ tale che $Si_{EX_2} \oplus Si^*_{EX_2} = Si'_{IX}$. Ripetendo la procedura esposta prima si otterrà un nuovo insieme di soluzioni S_2 con la stessa proprietà dell'insieme S_1, allora visto che

$$Si_{KX} \in S_1 \wedge Si_{KX} \in S_2 \Rightarrow Si_{KX} \in S_1 \cap S_2 \tag{3.3.2}$$

quindi una nuova stima la si può ottenere intersecando gli insiemi S_1 ed S_2. Ripetendo la procedura si dovrebbe arrivare a ottenere uno spazio di chiavi molto ridotto, se non addirittura un insieme con un solo elemento, che necessariamente deve essere la chiave. La condizione iniziale della procedura è che $S_0 = \{00_x, \ldots, 3F_x\}$, cioè coincida con tutti i valori che Si_{KX} può assumere.

3.3.1 Implementazione dell'attacco alle S-Box secondo la formulazione originale

La discussione appena tenuta permette di derivare il codice C++ che conduce al generico passo iterativo dell'attacco.

```
1   set<int> iterative_phase(int xor_input, int e, int (*s_box)(int)) {
2   /*
3   Calcolo del secondo output della funzione di espansione tale che
4   e xor e_star = xor_input (il primo d questi è fissabile liberamente).
5   */
6   int e_star = xor_input^e;
7   int (*S)(int) = s_box;
8   /*
9   Calcolo degli output, e dell'output xor
10  */
11  int o = S(e^key);
12  int o_star = S(e_star^key);
13  int o_prime = o^o_star;
14
15  /*
16  Calcoliamo gli input alle S-Box
17  */
18  set<pair<int,int>> insieme_coppie;
19  for(int i = 0; i < 0x3F; i++) {
20    int i_star = i^xor_input;
21    if((S(i)^S(i_star)) == o_prime) {
22      pair<int,int>* coppia_input = new pair<int,int>(i,i_star);
23      insieme_coppie.insert(*coppia_input);
24    }
25  }
26  /*
27  Calcoliamo le chiavi possibili
28  */
29  set<pair<int,int>>::iterator it;
30  set<int> chiavi;
31  for(it = insieme_coppie.begin(); it != insieme_coppie.end(); it++) {
32    int k = e^((*it).first);
33    int k_star = e^((*it).second);
34    chiavi.insert(k);
35    chiavi.insert(k_star);
36  }
37  return chiavi;
38 }
```

Listing 3.1: Procedura iterativa di attacco alle S-Box

La spiegazione del codice aiuterà a comprendere meglio come una S-Box vada violata. Gli input del codice nel listato 3.1 sono due interi e un puntatore a funzione, degli interi in input sono sempre considerati i sei bit meno significativi, il puntatore a funzione invece serve a identificare la particolare S-Box da violare. L'intero `xor_input` è la differenza di input fissata mentre `e` è un parametro libero, che identifica un'output della funzione di espansione $E(\cdot)$. La riga *6* calcola l'altro parametro(`e_star`), che fissato il primo è unico, quest'altro parametro sarebbe il secondo output della funzione di espansione, derivato dalla differenza `xor_input` e `e`. La riga *7* dichiara e inizializza la variabile di tipo puntatore a funzione, che è solo una variabile utilizzata per identificare la S-Box che si vuole attaccare. Le righe *11, 12* e *13* calcolano rispettivamente gli output delle S-Box dovuti a `e` ed `e_star`, entrambi in XOR con la chiave (incognita) `key`, da trovare. `o_prime` è invece l'output XOR. Si osservi

in riga *18* la dichiarazione di una lista di coppie, queste saranno tanto quante sono previste dalla *tabella di distribuzione degli XOR della S-Box*. Il ciclo che va da riga *19* a riga *25* ha lo scopo di determinare le possibili coppie di input alle S-Box tali che Si'_{IX} = `x_input` mentre Si'_{OX} = `o_prime`, ogni coppia trovata che rispetta il criterio viene riposta nella lista dichiarata in riga *18*. Determinate queste coppie si procede al calcolo dell'insieme di chiavi che riduce lo spazio delle chiavi, questo è lo scopo del secondo ciclo che va da riga *31* a *37*, in questo ciclo si itera sugli elementi della lista, di volta in volta si effettua lo xor degli elementi della coppia con gli output della funzione di espansione, le chiavi trovate si aggiungono così alla lista di chiavi dichiarata in riga *30*. L'insieme di chiavi così ottenuto viene ritornato dalla funzione. Il codice C++ implementato fa uso di funzioni della libreria standard ed è stato implementato allo scopo di chiarificare l'approccio originale della crittoanalisi differenziale sulla singola S-Box. Il listato in 3.2 mostra invece come avviene l'attacco complessivo alla S-Box.

```
1  set<int> attacco(list<int> xor_input, list<int> e, int (*s_box)(int)) {
2    /*
3    Inizializzazione dello spazio chiavi
4    ogni chiave è una possibile candidata
5    */
6    set<int> s1;
7    for(int i = 0x00; i <= 0x3F; i++) s1.insert(i);
8
9
10   list<int>::iterator it_e, it_xor_input;
11   it_xor_input = xor_input.begin();
12   for(it_e = e.begin(); it_e != e.end(); it_e++) {
13     set<int> s2 = iterative_phase((*it_xor_input),(*it_e),s_box);
14     cout << "Here!" << endl;
15     s1 = get_intersection(s1,s2);
16     it_xor_input++;
17   }
18
19   return s1;
20 }
```

Listing 3.2: Procedura di rottura di una S-Box

Nella funzione `attacco` le righe *6-7* fissano la condizione iniziale, le righe del loop *11-16* calcolano lo spazio delle chiavi stimato per intersezioni successive. La funzione `get_intersection` ritorna l'insieme intersezione degli input `s1` ed `s2`. Il costo computazionale dell'attacco dipende complessivamente dalle coppie di output della funzione di espansione scelte per l'attacco, alcune sequenze potrebbero portare l'attacco a convergere più velocemente di altre.

3.3.2 Esempio di attacco

E' stato esposto come implementare le procedure necessarie per rompere una S-Box, secondo la formulazione originale della crittoanalisi differenziale, in questa sottosezione si vuole fornire un esempio che mostri i vari passi. L'esempio in questione è ispirato ai dati di [18], in questo documento, come altri (ad esempio [10]), è riportato un tutorial sulla crittoanalisi differenziale applicata alla singola S-Box. In questa tesi si utilizzerà l'esempio esposto in [18] che mostra gli insiemi di chiavi stimati di ogni attacco.

Primo attacco: Come primo passo fissiamo $Si_{EX} = 08_x$ e $Si'_{IX} = 0C_x$, applicando la `iterative_phase` le righe *6-7* produrranno $Si^*_{EX} = 04_x$. Come incida la chiave nel calcolo di Si_{IX}, Si^*_{IX} non è noto, visto che la chiave non è nota ma l'attacco è di tipo chosen plaintext saranno noti quindi gli output $Si_{OX} = A_x$, $Si^*_{OX} = 7_x$ e $Si'_{OX} = D_x$. Il loop (righe *18-25*) produrrà l'insieme di coppie

$$\begin{aligned}\texttt{insieme\_coppie} = \quad & \{(01_x, 0D_x), (0D_x, 01_x), (12_x, 1E_x), \\ & (1E_x, 12_x), (36_x, 3A_x), (3A_x, 36_x)\}\,.\end{aligned} \tag{3.3.3}$$

Iterando su tutte le coppie della (3.3.3) si otterrà l'insieme di chiavi

$$\texttt{S2} = \{09_x, 05_x, 1A_x, 16_x, 3E_x, 32_x\}\,. \tag{3.3.4}$$

Se $\texttt{S1} = \{00_x, \ldots, 3F_x\}$ era la condizione iniziale, tramite `S2` si avrà l'aggiornamento

$$\texttt{S1} = \texttt{S1} \cap \texttt{S2} = \{09_x, 05_x, 1A_x, 16_x, 3E_x, 32_x\}\,. \tag{3.3.5}$$

Secondo attacco: Nel secondo attacco si fissa $Si_{EX} = 38_x$, $Si'_{IX} = 0C_x$, la S-Box è la $S1(\cdot)$ nuovamente, questa volta si avrà che $Si_{OX} = 0B_x$, $Si^*_{OX} = 1_x$ e $Si'_{OX} = A_x$. Nella seconda iterata si avrà

$$\begin{aligned}\texttt{insieme\_coppie} = \quad & \{(22_x, 2E_x), (2E_x, 22_x), (30_x, 3C_x), \\ & (3C_x, 30_x), (34_x, 38_x), (38_x, 34_x)\}\,,\end{aligned} \tag{3.3.6}$$

e quindi

$$\texttt{S2} = \{16_x, 1A_x, 04_x, 08_x, 00_x, 0C_x\}\,, \tag{3.3.7}$$

infine l'aggiornamento produrrà

$$\texttt{S1} = \texttt{S1} \cap \texttt{S2} = \{16_x, 1A_x\}\,. \tag{3.3.8}$$

Terzo attacco: Si sceglie $Si_{EX} = 3B_x$, $Si'_{IX} = 10_x$ e al solito per S-Box la S1. Saltando alcuni passi si avrà che $Si'_{OX} = A_x$. Le coppie in input stimate saranno

$$\begin{aligned}\texttt{insieme\_coppie} = \quad & \{(01_x, 11_x), (11_x, 01_x), (21_x, 31_x), \\ & (31_x, 21_x), (2F_x, 3F_x), (3F_x, 2F_x)\}\,,\end{aligned} \tag{3.3.9}$$

le chiavi stimate

$$\texttt{S2} = \{3A_x, 2A_x, 1A_x, 0A_x, 14_x, 04_x\}\,, \tag{3.3.10}$$

il che produrrà l'aggiornamento

$$\texttt{S1} = \texttt{S1} \cap \texttt{S2} = \{1A_x\}\,, \tag{3.3.11}$$

il che ci permette di affermare con certezza che la chiave è $Si_{KX} = 1A_x$.

Capitolo 4

Teoria della Probabilità applicata alle stringhe binarie

4.1 Motivazioni

Per giustificare l'implementazione della Rete Bayesiana, e dimostrare che l'approccio proposto di crittoanalisi differenziale porta agli stessi risultati di quello classico, è stato necessario approfondire alcuni aspetti legati alle stringhe binarie. Lo scopo di questo capitolo è sviluppare alcune definizioni e dimostrare alcuni lemmi che permettano di giustificare l'implementazione di alcuni algoritmi proposti nel seguito. Nel capitolo precedente si è visto che in [2] la distribuzione non uniforme delle differenze è stata osservata sperimentalmente, in questo capitolo invece si forniranno gli strumenti necessari per costruire il modello matematico che godrà di alcune proprietà che giustificheranno il metodo della crittoanalisi differenziale, per esempio la convergenza del metodo oppure la possibilità di dare uno speed-up al metodo.

4.2 Definizioni

Si consideri una funzione $f(\cdot)$ che trasformi una stringa X di n bit in una stringa Y di m bit, in termini informali il problema che si vuole affrontare è studiare il grado di incertezza di Y se è noto il grado di incertezza di X e la trasformazione che ha portato X a modificarsi in Y. Ovvero ipotizzando che X sia una variabile aleatoria con una propria densità di probabilità $p_X(x) = Pr\{X = x\}$, nota $f(\cdot)$, qual'è la densità di probabità $p_Y(y) = Pr\{Y = y\}$?. Nonostante esista un'ampia letteratura in merito alla statistica delle funzioni misurabili, per questa tesi è necessario specializzare il ragionamento per le funzioni booleane. L'interesse della statistica delle funzioni booleane è giustificato dal fatto che la crittoanalisi differenziale è un metodo probabilistico e che il DES in generale, e il mapping di Feistel in particolare, è composizione di funzioni booleane. Si supponga di avere una stringa binaria di n bit, ovvero una stringa della forma $x = x_n x_{n-1} \dots x_1$,

da tale stringa è possibile costruire un *unico* vettore appartenente a $\mathbb{Z}_2^n$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix};$$

viceversa avendo un vettore $\vec{x}$ è possibile costruire una stringa binaria di n bit prendendo le componenti e concatenandole. Quindi esiste un isomorfismo tra le stringhe binarie di n bit e i vettori di $\mathbb{Z}_2^n$, l'esistenza di questo isomorfismo in questa tesi sarà sempre sottointeso. Quando si avrà una stringa binaria aleatoria di n bit, o equivalentemente una variabile aleatoria booleana di n bit, si sottointenderà sempre il dominio $\mathbb{Z}_2^n$, ovvero questa stringa in generale potrebbe essere una qualunque stringa binaria.

Definizione 4.1. *Sia X una variabile aleatoria di n bit, e sia $D_X \subseteq \mathbb{Z}_2^n$ allora X è **uniformemente distribuita** in D_X, oppure **incerta** in D_X se risulta*

$$p_X(x) = \begin{cases} \frac{1}{\sharp(D_X)} & \text{se } x \in D_X \\ 0 & \text{se } x \in \mathbb{Z}_2^n - D_X \end{cases}$$

dove $\sharp(D_X)$ è il numero di elementi dell'insieme D_X.

La precedente definisce l'incertezza di una variabile booleana in un dominio generico D_X. La seguente definizione invece descrive il grado massimo di incertezza.

Definizione 4.2. *Sia X una variabile aleatoria di n bit uniformente distribuita in D_X. Se $D_X = \mathbb{Z}_2^n$ allora X è detta **totalmente incerta** .*

Si supponga di avere due variabili aleatorie di n bit X, Y uniformemente distribuite in D_X e D_Y, se il dominio D_X possiede un numero inferiore di elementi di D_Y, ovvero $\sharp(D_X) \leq \sharp(D_Y)$ ha senso dire che la variabile aleatoria X è **meno incerta** della variabile aleatoria Y, in quanto il range di valori che può assumere X è più piccolo di quello di Y. Quanto detto caratterizza una variabile aleatoria di n bit incerta, con relativi gradi e confronti, tuttavia di contro si hanno le variabili aleatorie *certe*, ovvero quelle variabili che da un punto di vista statistico assumono con probabilità 1 un determinato valore. Se X è una stringa di n bit si può caratterizzare la sua certezza con una generalizzazione della funzione $\delta(\cdot)$ di dirac discreta, ovvero

Definizione 4.3. *Sia X una stringa binaria di n bit definiamo l'applicazione $\delta_n(X)$ come*

$$\delta_n(X) = \prod_{i=1}^{n} \delta(x_i), \tag{4.2.1}$$

dove x_i è un bit di $X = x_1 \dots x_i \dots x_n$ e $\delta(\cdots)$ è la delta discreta classica.

Vale ora la seguente

Proposizione 4.1. *Sia X una stringa di n bit allora $\delta_n(X) = 1 \Leftrightarrow X = 0^n$*

Dimostrazione. Se $X = x_1 \dots x_n$ e $X = 0^n$ risulta che

$$x_i = 0 \; \forall i = 1, \dots, n, \tag{4.2.2}$$

il che implica $\delta(x_i) = 1$ per ogni $i = 1, \dots, n$ e quindi $\prod_{i=1}^{n} \delta(x_i) = 1$. Viceversa invece supponendo per assurdo che $\delta_n(X) = 1$ e $X \neq 0^n$ vuol dire che esiste almeno un $x_i \neq 0$ il che implica che per tale x_i si ha che $\delta(x_i) = 0$ ma questo comporterebbe che $0 = \prod_{i=1}^{n} \delta(x_i) = \delta_n(X)$, il che va contro le ipotesi fatte. □

Dalla proposizione precedente segue immediatamente che $\delta_n(X \oplus X_0) = 1 \Leftrightarrow X = X_0$.

Definizione 4.4. *Sia X una stringa aleatoria di n bit diremo che X è **certa** se risulta*

$$p_X(x) = \delta_n(x \oplus x_0),$$

per qualche $x_0 \in \mathbb{Z}_2^n$.

Si osservi che se X è una stringa aleatoria di n bit *incerta* in D_X, con $\sharp(D_X) = 1$ risulta che X è una stringa *certa* in D_X. Le definizioni appena date e la proposizione si possono generalizzare nel caso di distribuzioni congiunte; per esempio si dirà che la coppia di stringhe X, Y rispettivamente di n e m bit, sono uniformemente distribuite nel dominio $D_X \times D_Y \subseteq \mathbb{Z}_2^n \times \mathbb{Z}_2^m$ se

$$p_{X,Y}(x,y) = \begin{cases} \frac{1}{\sharp(D_X)\sharp(D_Y)} & \text{se } (x,y) \in D_X \times D_Y \\ 0 & \text{altrimenti} \end{cases},$$

in particolare la coppia (X, Y) è totalmente incerta se $D_X \times D_Y = \mathbb{Z}_2^n \times \mathbb{Z}_2^m$. Date due stringhe binarie X, Y di n e m bit si definisce

$$\delta_{n,m}(X, Y) = \delta_n(X)\delta_m(Y),$$

e quindi una coppia di stringhe di n e m bit aleatorie si dirà certa se la loro distribuzione è della forma $p_{X,Y}(x, y) = \delta_{n,m}(x, y)$, e così via si può procedere per terne, quadruple e via discorrendo. Una volta formalizzato ciò che qui verrà inteso come certezza o incertezza di una variabile aleatoria è di interesse vedere se c'è una qualche relazione tra queste definizioni e le funzioni che trasformano una variabile X in una Y.

4.3 Lemmi sulle funzioni booleane

Si ipotizzi di avere una variabile di n bit X e un valore di n bit noto Y, si consideri l'equazione

$$X \oplus Y = Z, \tag{4.3.1}$$

con Z nota, è banale vedere che tale equazione ha un'unica soluzione nella variabile X. Il seguente risultato generalizza la 4.3.1

Lemma 4.1. *Siano X, Y e Z tre stringhe binarie di n bit,con Z fissato, allora l'equazione*

$$X \oplus Y = Z$$

ammette esattamente 2^n soluzioni distinte nelle variabili X e Y.

Dimostrazione. La dimostrazione procede in modo in modo costruttivo, fissando Y pari a Y_0 si consideri l'equazione

$$X \oplus Y_0 = Z, \tag{4.3.2}$$

per quanto detto in precedenza questa equazione ammette un'unica soluzione, $X = Y_1 \oplus Z$. Quindi la coppia $(X, Y) = (Y_1 \oplus Z, Y_1)$ è una soluzione. Fissando $Y = Y_2 \neq Y_1$ si avrà un'altra soluzione $(X, Y) = (Y_2 \oplus Z, Y_2)$, e così via. Poichè è possibile fissare Y in un totale di 2^n modi distinti si hanno le seguenti coppie di soluzioni della 4.3.2:

$$S = \{(X, Y) \mid X \oplus Y = Z\} = \{(Y_1 \oplus Z, Y_1), \ldots, (Y_{2^n} \oplus Z, Y_{2^n})\}; \tag{4.3.3}$$

Poichè $\sharp(S) = 2^n$, il numero totale di soluzioni è 2^n. □

Il seguente lemma invece è un primo esempio di teoria della probabilità applicata alle funzioni booleane. Esso asserisce che se si ha una coppia di stringhe aleatorie di n bit, dove un elemento è completamente incerto e il secondo è certo, allora il loro XOR darà per risultante sarà una stringa completamente incerta.

Lemma 4.2. *Siano X e Y due stringhe aleatorie binarie di n bit statisticamente indipendenti caratterizzate dalle seguenti funzioni di probabilità*

- $p_X(x) = \frac{1}{2^n}$;
- $p_Y(y) = \delta(y \oplus y_0)$.

Sia inoltre Z una striga binaria tale che

$$Z = X \oplus Y = f(X, Y),$$

allora su Z è implicitamente definita la densità di probabilità

$$p_Z(z) = \frac{1}{2^n}. \tag{4.3.4}$$

Dimostrazione. f è una funzione misurabile, e quindi poichè f è definita a partire da variabili aleatorie anche Z sarà una variabile aleatoria, ha senso quindi associare a questa una propria densità di probabilità che va caratterizzata. Allora

$$p_Z(z) = Pr(Z = z) = Pr\{X \oplus Y = z\}.$$

Dal lemma 4.1 segue che il numero di soluzioni distinte per l'equazione $X \oplus Y = z$ è 2^n, quindi detto

$$S = \{(x_1, y_1), \ldots, (x_{2^n}, y_{2^n})\},$$

si ha che

$$Pr\{X \oplus Y = z\} = Pr\{(X, Y) = (x_1, y_1) \cup \ldots \cup (X, Y) = (x_{2^n}, y_{2^n})\},$$

siccome gli eventi sono tutti disgiunti vale che

$$Pr\{(X, Y) = (x_1, y_1) \cup \ldots \cup (X, Y) = (x_{2^n}, y_{2^n})\} = \sum_{i=1}^{2^n} Pr\{(X, Y) = (x_i, y_i)\}. \tag{4.3.5}$$

Visto che X e Y sono variabili aleatorie statisticamente indipendenti vale che

$$Pr\{(X,Y) = (x_i, y_i)\} = p_{X,Y}(x_i, y_i) = p_X(x_i)p_Y(y_i). \tag{4.3.6}$$

Sostituendo il secondo membro della (4.3.6) nel secondo membro della (4.3.5) si ha che

$$p_Z(z) = \sum_{i=1}^{2^n} p_X(x_i)p_Y(y_i); \tag{4.3.7}$$

sostituendo nella (4.3.7) le espressioni di $p_X(x_i) = \frac{1}{2^n}$ e $p_Y(y_i) = \delta(y_i \oplus y_0)$ si arriva a

$$p_Z(z) = \frac{1}{2^n} \sum_{i=1}^{2^n} \delta(y_i \oplus y_0) = \frac{1}{2^n}; \tag{4.3.8}$$

e quindi

$$p_Z(z) = \frac{1}{2^n}, \tag{4.3.9}$$

come volevasi dimostrare. Ovvero che Z è completamente incerta. □

Allo stesso risultato del lemma precedente si giunge se ambo le variabili X e Y sono completamente incerte.

Lemma 4.3. *Siano X e Y due stringhe binarie aleatorie di n bit statisticamente indipendenti tali che*

- $p_X(x) = \frac{1}{2^n}$;
- $p_Y(y) = \frac{1}{2^n}$.

Sia inoltre $Z = X \oplus Y$, allora risulta

$$p_Z(z) = \frac{1}{2^n}.$$

Dimostrazione. Per gli stessi motivi del lemma 4.2 Z è una variabile aleatoria, ragionando in modo analogo a quanto fatto con tale lemma si giunge all'uguaglianza

$$p_Z(z) = \sum_{i=1}^{2^n} p_X(x_i)p_Y(y_i). \tag{4.3.10}$$

Sostituendo nella (4.3.10) $p_X(x_i) = \frac{1}{2^n}$ e $p_Y(y_i) = \frac{1}{2^n}$ si ottiene

$$p_Z(z) = \sum_{i=1}^{2^n} \frac{1}{2^n 2^n} = \frac{1}{2^{2n}} \sum_{i=1}^{2^n} 1 = \frac{2^n}{2^{2n}} = \frac{1}{2^n}, \tag{4.3.11}$$

e quindi

$$p_Z(z) = \frac{1}{2^n}, \tag{4.3.12}$$

come volevasi dimostrare. □

Alcuni risultati interessanti riguardanti le funzioni booleane, ma di carattere più generale dei precedenti, sono i seguenti.

Lemma 4.4. *Siano X e Y due stringhe aleatorie di n bit caratterizzate da una densità di probabilità congiunta $p_{X,Y}(x,y)$ e sia Z una stringa di m bit tale che $Z = f(X,Y)$ una funzione misurabile. Se Z è un evento singolare si ha che*

$$p_{X,Y,Z}(x,y,z) = p_{X,Y}(x,y)\delta_m(z \oplus f(x,y)).$$

Dimostrazione. Si consideri $I^{-1}(Z)$ come l'insieme delle coppie (X,Y) tali che $f(X,Y) = Z$ (l'insieme controimmagine) visto che f è misurabile in particolare $I^{-1}(Z)$ sarà misurabile, ed ha quindi senso calcolarne la misura di probabilità. Ora si osservi che

$$p_{X,Y,Z}(x,y,z) = Pr\{X = x, Y = y, Z = z\}, \tag{4.3.13}$$

poichè $Z = f(X,Y)$ vale l'implicazione

$$Z = z \Rightarrow (X,Y) \in I^{-1}(z), \tag{4.3.14}$$

e quindi

$$Pr\{X = x, Y = y, Z = z\} = Pr\left\{X = x, Y = y, (X,Y) \in I^{-1}(z)\right\}. \tag{4.3.15}$$

Si consideri il sistema

$$\begin{cases} X = x \\ Y = y \\ (X,Y) \in I^{-1}(z) \end{cases} \Rightarrow (x,y) \in I^{-1}(z) \tag{4.3.16}$$

per il quale possono presentarsi le seguenti situazioni:

1. $(x,y) \notin I^{-1}(z)$;
2. $(x,y) \in I^{-1}(z)$.

Se il primo caso è verificato il sistema (4.3.16) non ha soluzione e quindi l'evento argomento della (4.3.15) è l'insieme vuoto $\emptyset$e la probabilità dell'insieme vuoto è nulla. Viceversa se vale il secondo caso l'evento argomento della (4.3.15) è l'evento $E = \{X = x, Y = y\}$ e quindi la (4.3.15) si riduce a

$$Pr\left\{X = x, Y = y, (X,Y) \in I^{-1}(z)\right\} = Pr\{X = x, Y = y\} = p_{X,Y}(x,y), \tag{4.3.17}$$

poichè si rientra nel caso 1 quando $Z \neq f(X,Y)$ e nel caso 2 quando $Z = f(X,Y)$ si conclude che

$$p_{X,Y,Z}(x,y,z) = p_{X,Y}(x,y)\delta_m(z \oplus f(x,y)). \tag{4.3.18}$$

□

Dal lemma precedente segue inoltre che

Lemma 4.5. *Siano X e Y due stringhe binarie aleatorie di n bit e sia Z una stringa di m bit tale che $Z = f(X, Y)$, f misurabile. Se le stringhe X, Y sono caratterizzate dalla densità congiunta $p_{X,Y}(x, y)$, con $p_{X,Y}(x, y) \neq 0$, allora si ha che*

$$p_{Z|X,Y}(x, y, z) = \delta_m(z \oplus f(x, y)).$$

Dimostrazione. Dalla definizione di probabilità condizionata si ha che

$$p_{Z|X,Y}(x, y, z) = \frac{p_{X,Y,Z}(x, y, z)}{p_{X,Y}(x, y)}; \tag{4.3.19}$$

dal lemma 4.4 si ha che

$$p_{X,Y,Z}(x, y, z) = p_{X,Y}(x, y)\delta_m(z \oplus f(x, y)), \tag{4.3.20}$$

sostituendo la (4.3.20) nella (4.3.19) si ha che

$$p_{Z|X,Y}(x, y, z) = \delta_m(z \oplus f(x, y)). \tag{4.3.21}$$

□

Capitolo 5

Rete Bayesiana per le S-Box

5.1 Costruzione della Rete

Per costruire la rete Bayesiana si procederà seguendo lo stile di [2], ovvero si partirà dalle S-Box per poi estendere ad un numero generico di Round. Sia Si_{EX} l'i-esimo blocco di 6 bit dell'output della funzione di espansione del Round X e sia Si_{KX} l'i-esimo blocco di sei bit della sottochiave di 48 bit del round X, visto che l'obiettivo della crittoanalisi delle S-Box è la ricerca delle sottochiavi, e che l'attacco è di tipo chosen plaintext, ha senso assumere che le variabili Si_{KX} e Si_{EX} siano uniformemente distribuita in $\mathbb{Z}_2^6$. L'input alla i-sima S-Box del round X è $Si_{IX} = f(Si_{EX}, Si_{KX})$ e visto il lemma 4.5 si ha che

$$p_{Si_{IX}|Si_{EX},Si_{KX}}(i, e, k) = \delta_6(i \oplus f(e, k)). \tag{5.1.1}$$

Visto che in particolare risulta

$$f(Si_{EK}, Si_{KX}) = Si_{EK} \oplus Si_{KX} \tag{5.1.2}$$

la (5.1.1) diventa

$$p_{Si_{IX}|Si_{EX},Si_{KX}}(i, e, k) = \delta_6(i \oplus e \oplus k). \tag{5.1.3}$$

Questa relazione statistica può essere indicata, in termini di reti bayesiane come in figura 5.1. Risulta chiaro che si considera un secondo blocco output di $E(\cdot)$,

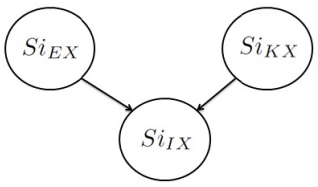


Figura 5.1: S-Box B-NET parziale 1.

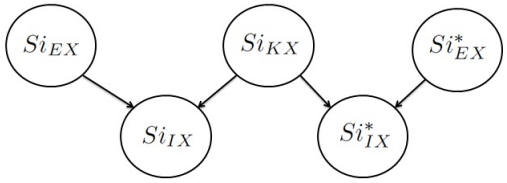


Figura 5.2: S-Box B-NET parziale 2.

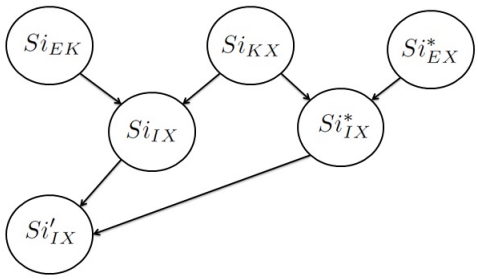


Figura 5.3: S-Box B-NET parziale 3.

definito come Si^*_{EX}, si ha che

$$Si^*_{IX} = Si_{KX} \oplus Si^*_{EX}, \tag{5.1.4}$$

e quindi, sempre in virtù del lemma 4.5, e della (5.1.4), vale che

$$p_{Si^*_{IX}|Si^*_{EX},Si_{KX}}(i^*, e^*, k) = \delta_6(i^* \oplus e^* \oplus k). \tag{5.1.5}$$

Di conseguenza la S-Box B-NET di figura 5.1 si modifica come in 5.2 . Si consideri ora la coppia (Si_{IX}, Si^*_{IX}), ponendo

$$Si'_{IX} = Si_{IX} \oplus Si^*_{IX}, \tag{5.1.6}$$

e sfruttando il lemma 4.5, si giunge a

$$p_{Si'_{IX}|Si_{IX},Si^*_{IX}}(i', i, i^*) = \delta_6(i' \oplus i \oplus i^*). \tag{5.1.7}$$

Quindi è possibile aggiungere un ulteriore nodo alla rete in figura 5.2 (vedasi figura 5.3). Infine definendo Si'_{OX} come

$$Si'_{OX} = Si(Si_{IX}) \oplus Si(Si^*_{IX}), \tag{5.1.8}$$

e sempre in virtù del lemma 4.5 vale allora che

$$p_{Si'_{OX}|Si_{IX},Si^*_{IX}}(o', o, o^*) = \delta_4(o' \oplus Si(o) \oplus Si(o^*)), \tag{5.1.9}$$

il che permette di aggiungere un ulteriore nodo che porta alla costruzione della rete in figura 5.4.

Nella sostanza la teoria sviluppata nel capitolo precedente ha permesso la costruzione della Rete Bayesiana rappresentata in figura 5.4. Il risultato ottenuto lo si può riassumere con il seguente teorema.

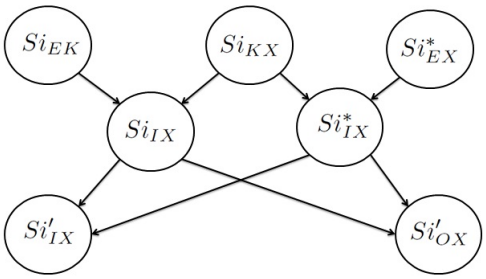


Figura 5.4: S-Box B-NET completa.

Teorema 5.1. *Si consideri la rete Bayesiana rappresentata in figura 5.4 (S-Box B-NET) e si considerino le ipotesi fatte sulle variabili della rete allora le seguenti densità di probabilità sono quelle da associare ai vari nodi della rete:*

1. $p_{Si_{EK}}(e) = \frac{1}{2^6}$;
2. $p_{Si^*_{EK}}(e^*) = \frac{1}{2^6}$;
3. $p_{Si_{KX}}(k) = \frac{1}{2^6}$;
4. $p_{Si_{IX}|Si_{EX},Si_{KX}}(i,e,k) = \delta_6(i \oplus e \oplus k)$;
5. $p_{Si^*_{IX}|Si^*_{EX},Si_{KX}}(i^*,e^*,k) = \delta_6(i^* \oplus e^* \oplus k)$;
6. $p_{Si'_{IX}|Si_{IX},Si^*_{IX}}(i',i,i^*) = \delta_6(i' \oplus i \oplus i^*)$;
7. $p_{Si'_{OX}|Si_{IX},Si^*_{IX}}(o',i,i^*) = \delta_6(o' \oplus Si(i) \oplus Si(i^*))$.

Dimostrazione. Basta avvalersi della teoria sviluppata

1. $\Rightarrow$ è vero perchè Si_{EX} è assunto essere distribuito in maniera uniforme;
2. $\Rightarrow$ è vero perchè Si^*_{EX} è assunto essere distribuito in maniera uniforme;
3. $\Rightarrow$ è vero perchè Si_{KX} è assunto essere distribuito in maniera uniforme;
4. $\Rightarrow$ è vero poichè $Si_{IX} = f(Si_{EX}, Si_{KX})$ e vale quindi il lemma 4.5;
5. $\Rightarrow$ è vero poichè $Si^*_{IX} = f(Si^*_{EX}, Si_{KX})$ e vale quindi vale il 4.5;
6. $\Rightarrow$ è vero poichè $Si'_{IX} = f(Si_{IX}, Si^*_{IX})$ e vale quindi vale il lemma 4.5;
7. $\Rightarrow$ è vero poichè $Si'_{OX} = f(Si_{IX}, Si^*_{IX})$ e vale quindi vale il lemma 4.5.

□

Osservazioni Vale la pena discutere le assunzioni fatte in merito alle densità a priori dei nodi Si_{EX}, Si^*_{EX} e Si_{KX}. Visto l'obiettivo è attaccare le S-Box sotto le ipotesi di chosen plaintext, non vi sono vincoli sulla scelta del testo in chiaro, in questo caso Si_{EX}, Si^*_{EX}. Poichè questi appartengono al dominio di $\mathbb{Z}_2^6$, e non vi sono preferenze nella scelta ha senso assumere la distribuzione in tale dominio uniforme. L'assunzione che Si_{KX} sia uniformemente distribuito all'interno del suo dominio deriva dall'assenza di informazione su quale sia la chiave corretta di conseguenza viene assunto che Si_{KX} sia *completamente incerta* in $\mathbb{Z}_2^6$.

5.2 Attacco singolo ad una singola S-Box

Si ipotizzi di avere una coppia di Si_{EX}, Si^*_{EX} tale che $Si_{EX} \oplus Si_{EX} = Si'_{IX}$, e di osservare l'output Si'_{OX}. Il problema della crittoanalisi differenziale si riconduce a massimizzare la likelihood

$$\hat{k} = argmax_{k \in \mathbb{Z}_2^6} \left(p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}}(k, e, e^*, i', o') \right) \tag{5.2.1}$$

nota la struttura della S-Box B-NET. Determinare una espressione della likelihood sarà equivalente a elaborare un algoritmo per la soluzione del problema posto.

5.2.1 Espressione della Likelihood

Nella discussione che segue verranno omessi i parametri della likelihood, sottointendendo sempre la loro presenza. Lo scopo di questa sezione è ricavare una espressione della likelihood per un singolo attacco ad una generica S-Box $Si(\cdot)$, per poterla ricavare sono necessari due lemmi.

Lemma 5.1. *Nella S-Box B-NET vale che*

$$p_{Si'_{OX}|Si'_{IX}} = \frac{N(Si'_{IX} \to Si'_{OX})}{2^6}, \tag{5.2.2}$$

dove $N(Si'_{IX} \to Si'_{OX})$ è il numero di coppie (Si_{IX}, Si^*_{IX}) tali che il sistema

$$\begin{cases} Si'_{IX} = Si_{IX} \oplus Si^*_{IX} \\ Si'_{OX} = Si(Si_{IX}) \oplus Si(Si^*_{IX}) \end{cases}$$

è verificato.

Dimostrazione. Basta effettuare gli opportuni passaggi per l'inferenza bayesiana. Seguendo la struttura della S-Box B-NET si ha che

$$p_{Si'_{OX}|Si'_{IX}} = \sum_{Si_{IX},Si^*_{IX}} \left[\frac{p_{Si'_{OX},Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si_{IX}} \times p_{Si^*_{IX}}}{p_{Si'_{IX}}} \right], \tag{5.2.3}$$

dove l'esplicitazione dell'operazione di prodotto è fatta per una maggiore leggibilità. Nel generico addendo della sommatoria a secondo membro della (5.2.3) per il denominatore vale che

$$p_{Si'_{IX}} = \frac{1}{2^6}, \tag{5.2.4}$$

mentre nel numeratore si ha che

$$p_{Si_{IX}} = p_{Si^*_{IX}} = \frac{1}{2^6}; \tag{5.2.5}$$

La validità delle (5.2.4) e (5.2.5) deriva dalla validità del sistema

$$\begin{cases} Si^*_{IX} = Si^*_{EX} \oplus Si_{KX} \\ Si_{IX} = Si_{EX} \oplus Si_{KX} \\ Si'_{IX} = Si_{IX} \oplus Si^*_{IX} \end{cases} \tag{5.2.6}$$

e dalla costruzione della S-Box B-NET, la quale è tale che i nodi $Si_{EX}, Si^*_{EX}, Si_{KX}$ sono uniformemente distribuiti in $\mathbb{Z}_2^6$. Applicando il lemma 4.3 alle prime due equazioni del sistema (5.2.6) si ha che anche i nodi Si_{IX}, Si^*_{IX} hanno una distribuzione uniforme in $\mathbb{Z}_2^6$, segue quindi la validità della (5.2.5). Visto che i nodi Si_{IX}, Si^*_{IX} sono uniformemente distribuiti a loro volta vale la (5.2.4) sempre per il lemma 4.3. Sostituendo le (5.2.4) e (5.2.5) nel secondo membro della (5.2.3) si arriva all'uguaglianza

$$p_{Si'_{OX}|Si'_{IX}} = \frac{1}{2^6} \sum_{Si_{IX}, Si^*_{IX}} p_{Si'_{OX}, Si'_{IX}|Si_{IX}, Si^*_{IX}}. \tag{5.2.7}$$

Nella (5.2.7), visto che i nodi Si_{IX} e Si^*_{IX} D-Separano i nodi Si'_{OX} e Si'_{IX}, si ha

$$p_{Si'_{OX}|Si'_{IX}} = \frac{1}{2^6} \sum_{Si_{IX}, Si^*_{IX}} \left[p_{Si'_{OX}|Si_{IX}, Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX}, Si^*_{IX}} \right]. \tag{5.2.8}$$

Nella (5.2.8) si consideri il generico addendo della somma a secondo membro, per il teorema 5.1 si ha che

$$\begin{aligned} &p_{Si'_{OX}|Si_{IX}, Si^*_{IX}} p_{Si'_{IX}|Si_{IX}, Si^*_{IX}} = \\ &\quad \delta_4(Si'_{OX} \oplus Si(Si_{IX}) \oplus Si(Si^*_{IX}))\delta_6(Si'_{IX} \oplus Si_{IX} \oplus Si^*_{IX}) = \\ &\quad\quad \delta_{4,6}(Si'_{OX} \oplus Si(Si_{IX}) \oplus Si(Si^*_{IX}), Si'_{IX} \oplus Si_{IX} \oplus Si^*_{IX}) \end{aligned} \tag{5.2.9}$$

il quale è diverso da 0 ogni qualvolta il sistema

$$\begin{cases} Si'_{IX} = Si_{IX} \oplus Si^*_{IX} \\ Si'_{OX} = Si(Si_{IX}) \oplus Si(Si^*_{IX}) \end{cases} \tag{5.2.10}$$

è verificato. La somma a secondo membro della (5.2.8) conteggia quindi il numero di coppie (Si_{IX}, Si^*_{IX}) tali che sia valido il sistema (5.2.10) quindi

$$\sum_{Si_{IX}, Si^*_{IX}} \left[p_{Si'_{OX}|Si_{IX}, Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX}, Si^*_{IX}} \right] = N(Si'_{IX} \to Si'_{OX}), \tag{5.2.11}$$

che porta alla conclusione

$$p_{Si'_{OX}|Si'_{IX}} = \frac{N(Si'_{IX} \to Si'_{OX})}{2^6}. \tag{5.2.12}$$

□

Lemma 5.2. *Nella S-Box B-NET vale che*

$$p_{Si_{EX}, Si^*_{EX}|Si'_{IX}, Si'_{OX}} = \frac{1}{2^6}, \tag{5.2.13}$$

per ogni coppia (Si_{EX}, Si^*_{EX}) *tale che* $Si_{EX} \oplus Si^*_{EX} = Si'_{IX}$

Dimostrazione. Per il primo membro della (5.2.13) vale l'uguaglianza

$$p_{Si_{EX}, Si^*_{EX}|Si'_{IX}, Si'_{OX}} = \sum_{Si_{IX}, Si^*_{IX}, Si_{KX}} p_{Si_{EX}, Si^*_{EX}, Si_{IX}, Si^*_{IX}, Si_{KX}|Si'_{OX}, Si'_{IX}}, \tag{5.2.14}$$

nella quale, nel secondo membro, si può riscrivere il generico addendo come

$$p_{Si_{EX},Si^*_{EX},Si_{IX},Si^*_{IX},Si_{KX}|Si'_{OX},Si'_{IX}} = \frac{p_{Si_{EX},Si^*_{EX},Si_{IX},Si^*_{IX},Si_{KX},Si'_{OX},Si'_{IX}}}{p_{Si'_{OX}|Si'_{IX}} \times p_{Si'_{IX}}}. \tag{5.2.15}$$

Osservando che il numeratore della (5.2.15) esprime una distribuzione congiunta, per le proprietà delle reti Bayesiane si può scrivere l'uguaglianza

$$\begin{aligned} p_{Si_{EX},Si^*_{EX},Si_{IX},Si^*_{IX},Si_{KX},Si'_{OX},Si'_{IX}} = \\ p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si_{IX}|Si_{EX},Si_{KX}} \times \\ p_{Si^*_{IX}|Si^*_{EX},Si_{KX}} \times p_{Si_{EX}} p_{Si^*_{EX}} p_{Si_{KX}}. \end{aligned} \tag{5.2.16}$$

Dalla (5.2.16), visto che i nodi $Si_{EX}, Si^*_{EX}, Si_{KX}$ sono uniformemente distribuiti in $\mathbb{Z}_2^6$, segue l'uguaglianza

$$\begin{aligned} &p_{Si_{EX},Si^*_{EX},Si_{IX},Si^*_{IX},Si_{KX},Si'_{OX},Si'_{IX}} = \\ &\frac{1}{2^{18}} p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si_{IX}|Si_{EX},Si_{KX}} \times p_{Si^*_{IX}|Si^*_{EX},Si_{KX}}. \end{aligned} \tag{5.2.17}$$

Si osservi ora che per il denominatore della (5.2.15) vale l'uguaglianza

$$p_{Si'_{OX}|Si'_{IX}} p_{Si'_{IX}} = \frac{N(Si'_{IX} \to Si'_{OX})}{2^6} \frac{1}{2^6} = \frac{N(Si'_{IX} \to Si'_{OX})}{2^{12}}, \tag{5.2.18}$$

che segue dal lemma 5.1. Sostituendo l'ultimo membro della (5.2.18) e la (5.2.17) in (5.2.15), e quest'ultima in (5.2.14) si ha

$$\begin{aligned} p_{Si_{EX},Si^*_{EX}|Si'_{IX},Si'_{OX}} = \frac{1}{2^6 N(Si'_{IX} \to Si'_{OX})} \cdot \\ \sum_{Si_{IX},Si^*_{IX},Si_{KX}} \Big[p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times \\ p_{Si_{IX}|Si_{EX},Si_{KX}} \times p_{Si^*_{IX}|Si^*_{EX},Si_{KX}} \Big]. \end{aligned} \tag{5.2.19}$$

Nel generico addendo della (5.2.19) si può osservare che i fattori $p_{Si_{IX}|Si_{EX},Si_{KX}}$ e $p_{Si^*_{IX}|Si^*_{EX},Si_{KX}}$ rendono la somma non nulla laddove il sistema

$$\begin{cases} Si_{KX} \in \mathbb{Z}_2^6 \\ Si_{IX} = Si_{EX} \oplus Si_{KX} \\ Si^*_{IX} = Si^*_{EX} \oplus Si_{KX} \end{cases} \tag{5.2.20}$$

è verificato. Applicando le condizioni in (5.2.20) alla somma in (5.2.19) si ha l'uguaglianza

$$\begin{aligned} &p_{Si_{EX},Si^*_{EX}|Si'_{IX},Si'_{OX}} = \\ &\frac{1}{2^6 N(Si'_{IX} \to Si'_{OX})} \sum_{\substack{Si_{KX} \in \mathbb{Z}_2^6 \\ Si_{IX} = Si_{EX} \oplus Si_{KX} \\ Si^*_{IX} = Si^*_{EX} \oplus Si_{KX}}} \Big[p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \Big]. \end{aligned} \tag{5.2.21}$$

Si vuole ora calcolare il valore della sommatoria in (5.2.21). Si osservi che per ipotesi si ha che $Si_{EX} \oplus Si^*_{EX} = Si'_{IX}$, quindi per ogni $Si_{KX} \in \mathbb{Z}_2^6$ si avrà che $Si_{IX} \oplus Si^*_{IX} = Si'_{IX}$, quindi $p_{Si'_{IX}|Si_{IX},Si^*_{IX}}$ è sempre pari a 1, mentre $p_{Si'_{OX}|Si_{IX},Si^*_{IX}}$ è non nullo solo quando $Si(Si_{EX} \oplus Si_{KX}) \oplus Si(Si^*_{EX} \oplus Si_{KX})$ non è nullo; quindi si ha che

$$\sum_{\substack{Si_{KX}\in\mathbb{Z}_2^6 \\ Si_{IX}=Si_{EX}\oplus Si_{KX} \\ Si^*_{IX}=Si^*_{EX}\oplus Si_{KX}}} \left[p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX},Si^*_{IX}}\right] = N(Si'_{IX} \to Si'_{OX}). \tag{5.2.22}$$

Sostituendo la (5.2.22) in (5.2.21) si giunge a

$$p_{Si_{EX},Si^*_{EX}|Si'_{IX},Si'_{OX}} = \frac{1}{2^6}, \tag{5.2.23}$$

come volevasi dimostrare. □

Si consideri la $p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}}$ per la regola di Bayes vale che

$$p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} = \frac{p_{Si_{KX},Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}}}{p_{Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}}}. \tag{5.2.24}$$

Visti i lemmi 5.1 e 5.2 il denominatore del secondo membro della (5.2.24) si può scrivere come

$$\begin{aligned} p_{Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} &= p_{Si_{EX},Si^*_{EX}|Si'_{IX},Si'_{OX}} p_{Si'_{IX},Si'_{OX}} = \\ &\underbrace{p_{Si_{EX},Si^*_{EX}|Si'_{IX},Si'_{OX}}}_{\frac{1}{2^6}} \underbrace{p_{Si'_{OX}|Si'_{IX}}}_{\frac{N(Si'_{IX}\to Si'_{OX})}{2^6}} \underbrace{p_{Si'_{IX}}}_{\frac{1}{2^6}} = \\ &\frac{N(Si'_{IX} \to Si'_{OX})}{2^{18}}, \end{aligned} \tag{5.2.25}$$

mentre il numeratore del secondo membro della (5.2.24) può essere riscritto come

$$\begin{aligned} p_{Si_{KX},Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} &= \sum_{Si_{IX},Si^*_{IX}} p_{Si_{KX},Si_{EX},Si_{IX},Si^*_{IX}Si^*_{EX},Si'_{IX},Si'_{OX}} = \\ &\sum_{Si_{IX},Si^*_{IX}} \Big[p_{Si'_{IX},Si'_{OX}|Si_{KX},Si_{EX},Si_{IX},Si^*_{IX}Si^*_{EX}} \times \\ &p_{Si_{IX},Si^*_{IX}|Si_{KX},Si_{EX},Si^*_{EX}} \times p_{Si_{KX},Si_{EX},Si^*_{EX}}\Big]. \end{aligned} \tag{5.2.26}$$

Nella (5.2.26) si ha che

$$p_{Si_{KX},Si_{EX},Si^*_{EX}} = p_{Si_{KX}} p_{Si_{EX}} p_{Si^*_{EX}} = \frac{1}{2^{18}} \tag{5.2.27}$$

$$p_{Si_{IX},Si^*_{IX}|Si_{KX},Si_{EX},Si^*_{EX}} = p_{Si_{IX}|Si_{KX},Si_{EX}} p_{Si^*_{IX}|Si_{KX},Si^*_{EX}} \tag{5.2.28}$$

$$p_{Si'_{IX},Si'_{OX}|Si_{KX},Si_{EX},Si_{IX},Si^*_{IX}Si^*_{EX}} = p_{Si'_{IX},Si'_{OX}|Si_{IX},Si^*_{IX}}. \tag{5.2.29}$$

Da (5.2.26), (5.2.27), (5.2.28) e (5.2.29) segue che

$$p_{Si_{KX},Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} = \frac{1}{2^{18}} \sum_{Si_{IX},Si^*_{IX}} \left[p_{Si'_{IX},Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si_{IX}|Si_{KX},Si_{EX}} \times p_{Si^*_{IX}|Si_{KX},Si^*_{EX}} \right]; \tag{5.2.30}$$

vista la D-Separabilità dei nodi Si'_{OX}, Si'_{IX} dati Si_{IX}, Si^*_{IX} si giunge a

$$p_{Si_{KX},Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} = \frac{1}{2^{18}} \sum_{Si_{IX},Si^*_{IX}} \Big[p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si_{IX}|Si_{KX},Si_{EX}} \times p_{Si^*_{IX}|Si_{KX},Si^*_{EX}} \Big]; \tag{5.2.31}$$

sostituendo (5.2.31) e (5.2.25) in (5.2.24) si giunge a

$$\begin{aligned} p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} &= \frac{1}{N(Si'_{IX} \to Si'_{OX})} \sum_{Si_{IX},Si^*_{IX}} \Big[p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times \\ &\quad p_{Si_{IX}|Si_{KX},Si_{EX}} \times p_{Si^*_{IX}|Si_{KX},Si^*_{EX}} \Big] = \\ &\frac{1}{N(Si'_{IX} \to Si'_{OX})} \sum_{\substack{Si_{IX}=Si_{EX}\oplus Si_{KX} \\ Si^*_{IX}=Si^*_{EX}\oplus Si_{KX}}} \left[p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \right]. \end{aligned} \tag{5.2.32}$$

Osservazione La (5.2.32) esprime come calcolare la probabilità che una data chiave sia corretta. Quando si calcola la probabilità che una data chiave Si_{KX} sia corretta, data l'evidenza dovuta all'attacco effettuato, si può osservare che esiste un dominio $D_X \subset \mathbb{Z}_2^6$ tale che per ogni $Si_{KX} \in D_X$ si ha che

$$p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} = \frac{1}{\sharp(D_X)}, \tag{5.2.33}$$

mentre se $Si_{KX} \notin D_X$ si ha che

$$p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}} = 0. \tag{5.2.34}$$

Si può quindi affermare che la distribuzione delle chiavi, dopo un'attacco, è *uniforme in una restrizione* $D_X \subset \mathbb{Z}_2^6$.

5.2.2 Pseudocodice per il calcolo della Likelihood

Ci si pone il problema dell'implementazione del calcolo della probabilità che una chiave sia corretta, vista la teoria sviluppata in merito. Un primo metodo

Algoritmo 5.1 Algoritmo calcolo probabilità che una chiave sia corretta dopo un singolo attacco

```
1: function PROB_KEY_SINGLE_ATTACK(k_x,e_x,e*_x,i'_x,o'_x)
2:     p ← 0
3:     for i_x = 0_x → 3F_x do
4:         for i*_x = 0_x → 3F_x do
5:             p_1 ← δ_6(i_x ⊕ e_x ⊕ k_x)
6:             p_2 ← δ_6(i*_x ⊕ e*_x ⊕ k_x)
7:             p_3 ← δ_6(i'_x ⊕ i_x ⊕ i*_x)
8:             p_4 ← δ_4(o'_x ⊕ Si(i_x) ⊕ Si(i*_x))
9:             p ← p + p_1 * p_2 * p_3 * p_4
10:        end for
11:    end for
12:    return p
13: end function
```

Algoritmo 5.2 Algoritmo calcolo likelihood distribuzione chiavi dopo un singolo attacco

```
1: function LIKELIHOOD_KEY_SINGLE_ATTACK(e_x,e*_x,i'_x,o'_x)
2:     p[2^6] ← 0
3:     for k_x = 0_x → 3F_x do
4:         p[k_x] ← PROB_KEY_SINGLE_ATTACK(k_x, e_x, e*_x, i'_x, o'_x)
5:     end for
6:     return p
7: end function
```

è quello di implementare direttamente

$$p_{Si_{KX}|Si'_{IX},Si'_{OX},Si_{EX},Si^*_{EX}} = \eta \sum_{Si_{IX},Si^*_{IX}} \Big[p_{Si'_{OX}|Si_{IX},Si^*_{IX}} \times p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \times p_{Si_{IX}|Si_{EX},Si_{KX}} \times p_{Si^*_{IX}|Si^*_{EX},Si_{KX}} \Big]. \quad (5.2.35)$$

Il calcolo della (5.2.35) può essere fatto attraverso lo pseudocodice 5.1 che calcola la probabilità non normalizzata, cioè trascura il calcolo de fattore η che non influisce poichè la distribuzione è uniforme in una opportuna restrizione di $\mathbb{Z}_2^6$, come si vede dalla (5.2.32).

I parametri della funzione dell'algoritmo 5.1 sono quattro interi a sei bit k_x, e_x, e^*_x, i^*_x e un intero a quattro bit o'_x, questi sono rispettivamente la sottochiave di sei bit, gli output della funzione di espansione, l'input XOR alla S-Box i-esima e l'output XOR della medesima. La riga 2 inizializza la probabilità a 0, un doppio ciclo for tra le righe 3-11 scorre i possibili input alle S-Box, applicando poi nelle righe 5-8 le definizioni del teorema 5.1, al fine di calcolare incrementalmente la probabilità in riga 9. Per calcolare la likelihood, sempre non normalizzata, si può utilizzare l'algoritmo 5.2

Nell'algoritmo 5.2 gli input sono tre interi a sei bit, gli output della funzione di espansione e l'input XOR alla S-Box, e un intero a quattro bit, l'output XOR

Algoritmo 5.3 Algoritmo ottimizzato di calcolo della probabilità che una chiave sia corretta dopo un singolo attacco.

1: **function** PROB_KEY_SINGLE_ATTACK_V2(k_x,e_x,e^*_x,i'_x,o'_x)
2: $i_x = e_x \oplus k_x$
3: $i^*_x = e^*_x \oplus k_x$
4: **return** $\delta_4(o'_x \oplus Si(i_x) \oplus Si(i^*_x))\delta_6(i'_x \oplus i_x \oplus i^*_x)$
5: **end function**

della medesima S-Box. La riga 2 inizializza un'array di 2^6 celle a 0, il ciclo for tra le righe 3 e 5 calcola la probabilità dei valori delle varie chiavi, e ciascuna probabilità viene memorizzato nell'array dichiarato in riga 2. L'algoritmo 5.2 implementa la (5.2.35).

5.2.3 Calcolo ottimizzato della likelihood

L'algoritmo 5.1 calcola la probabilità di una data chiave con totale di 2^{12} iterazioni, mentre l'algoritmo 5.2 testa 2^6 possibili chiavi; siccome quest'ultimo richiama la funzione *prob_key_single_attack* il numero totale di passi per il calcolo della likelihood è di 2^{18}. Il calcolo può essere ottimizzato se invece di implementare la (5.2.35) si implementa l'ultimo membro della (5.2.32). La riga 2 calcola i valori Si_{IX}, Si^*_{IX}, come dettato dalla condizione della somma in (5.2.32), in riga 4 si calcola invece la probabilità

$$p_{Si'_{OX}|Si_{IX},Si^*_{IX}} p_{Si'_{IX}|Si_{IX},Si^*_{IX}} \tag{5.2.36}$$

secondo quanto dettato dal teorema 5.1. Se nell'algoritmo 5.2 al posto della chiamata a *prob_key_single_attack* si sostituisce la chiamata all'algoritmo 5.3 il numero di passi necessari al calcolo della likelihood sarà 2^6.

5.3 Attacco multiplo ad una singola S-Box

L'attacco della sezione precedente parte dall'ipotesi di avere una chiave Si_{KX} *completamente incerta*, dopo aver dato in input una coppia Si_{EX}, Si^*_{EX}. L'ipotesi iniziale di attacco è che la chiave possa appartenere a $\mathbb{Z}_2^6$, un singolo attacco porterà a concludere che la chiave Si_{KX} possa appartenere ad un dominio $D_K \subset \mathbb{Z}_2^6$. Un singolo attacco però non porta mai ad avere un dominio D_K che abbia un solo elemento, ecco perchè è necessario eseguire più attacchi in successione affinchè si abbia un dominio con una singola chiave, che necessariamente deve essere quella corretta.

5.3.1 Espressione della likelihood

Si supponga quindi di poter disporre di un insieme di coppie X

$$X = \left\{(Si_{EX_1}, Si^*_{EX_1}), \ldots, (Si_{EX_n}, Si^*_{EX_n})\right\}. \tag{5.3.1}$$

Utilizzando la prima coppia ricaviamo un input XOR e un output XOR

$$\begin{aligned} Si'_{IX_1} &= Si_{EX_1} \oplus Si^*_{EX_1} \\ Si'_{OX_1} &= Si(Si_{IX_1}) \oplus Si(Si^*_{IX_1}), \end{aligned}$$

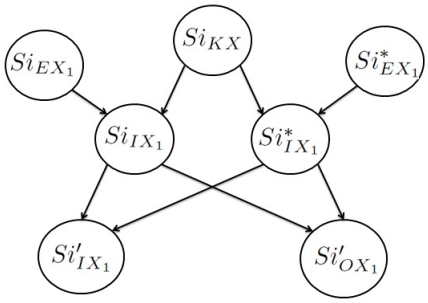


Figura 5.5: S-Box B-NET per il primo attacco

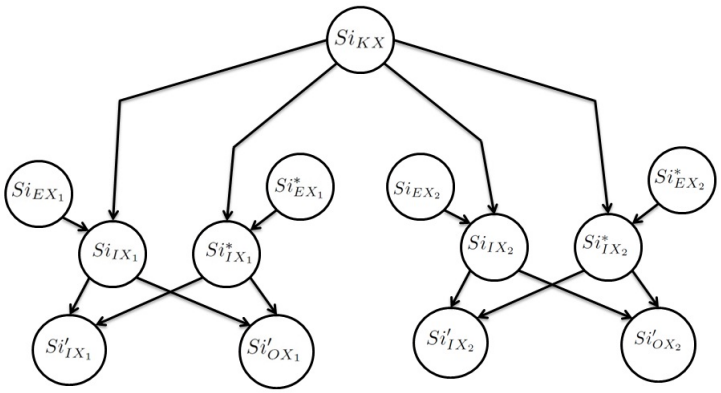


Figura 5.6: S-Box B-NET per due attacchi consecutivi

e si avrà quindi un insieme di evidenza complessivo

$$\Theta_1 = \left\{ Si_{EX_1}, Si^*_{EX_1}, Si'_{IX_1}, Si'_{OX_1} \right\}. \tag{5.3.2}$$

Dall'evidenza in (5.3.2) segue che è possibile attaccare la S-Box $Si(\cdot)$ utilizzando la rete in figura 5.5. L'inferenza sulla rete in figura 5.5 permetterà di ottenere una likelihood

$$p_{Si_{KX}|\Theta_1} \tag{5.3.3}$$

che ridurrà l'incertezza sulla chiave Si_{KX}. Utilizzando la seconda coppia in (5.3.1) si avrà l'insieme evidenza

$$\Theta_2 = \left\{ Si_{EX_2}, Si^*_{EX_2}, Si'_{IX_2}, Si'_{OX_2} \right\}, \tag{5.3.4}$$

che renderà possibile effettuare un'attacco concatenato al precedente ancorando la nuova rete alla rete dell'attacco precedente (figura 5.6), quindi si avrà una likelihood del tipo

$$p_{Si_{KX}|\Theta_1,\Theta_2}. \tag{5.3.5}$$

Poichè la chiave D-Separa i vari attacchi si avrà che

$$p_{Si_{KX}|\Theta_1,\Theta_2} = \eta p_{Si_{KX}|\Theta_1} p_{Si_{KX}|\Theta_2}. \tag{5.3.6}$$

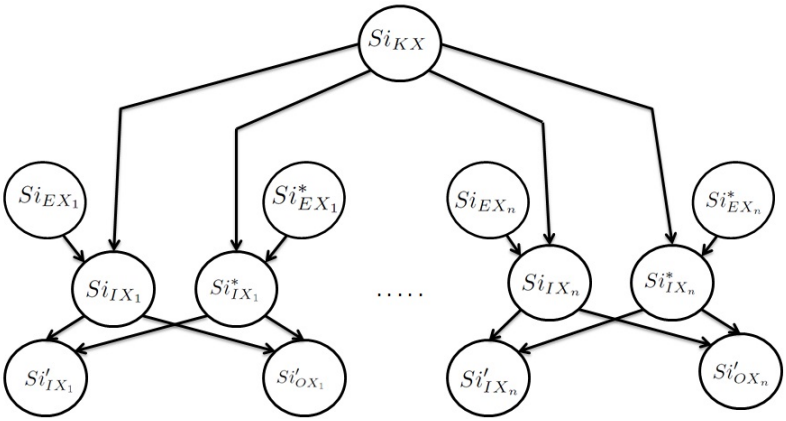


Figura 5.7: S-Box B-Net per un numero di attacchi generico

Generalizzando il ragionamento, dato un insieme di evidenze

$$\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_n\}, \tag{5.3.7}$$

dove

$$\Theta_j = \left\{Si_{EX_j}, Si^*_{EX_j}, Si'_{IX_j}, Si'_{OX_j}\right\}, \tag{5.3.8}$$

si può attaccare una singola S-Box avvalendosi della rete in figura 5.7. Considerando che il nodo Si_{KX} D-separa tutti i tentativi si ha che

$$p_{Si_{KX}|\Theta} = p_{Si_{KX}|\Theta_1\Theta_2\ldots\Theta n} = \eta \prod_{i=1}^{n} p_{Si_{KX}|\Theta_i}. \tag{5.3.9}$$

Quanto detto si può riassumere mediante il seguente teorema

Teorema 5.2. *La likelihood derivante dalla S-Box B-NET per un attacco multiplo è della forma*

$$p_{Si_{KX}|\Theta_1,\Theta_2,\ldots,\Theta_n} = \eta \prod_{i=1}^{n} p_{Si_{KX}|\Theta_i}. \tag{5.3.10}$$

Dimostrazione. Vale l'uguaglianza

$$p_{Si_{KX}|\Theta_1,\Theta_2,\ldots,\Theta_n} = \eta p_{\Theta_1,\Theta_2,\ldots,\Theta_n|Si_{KX}} p_{Si_{KX}}; \tag{5.3.11}$$

poichè Si_{KX} D-Separa le evidenze $\Theta_1, \ldots \Theta_n$, vale inoltre

$$\eta p_{\Theta_1,\Theta_2,\ldots,\Theta_n|Si_{KX}} p_{Si_{KX}} = \eta p_{\Theta_1|Si_{KX}} p_{\Theta_2|Si_{KX}} \ldots p_{\Theta_n|Si_{KX}} p_{Si_{KX}}. \tag{5.3.12}$$

La (5.3.12) può essere riscritta come

$$\eta p_{\Theta_1|Si_{KX}} p_{\Theta_2|Si_{KX}} \cdots p_{\Theta_n|Si_{KX}} p_{Si_{KX}} =$$

$$\eta p_{\Theta_1|Si_{KX}} p_{\Theta_2|Si_{KX}} \cdots p_{\Theta_n|Si_{KX}} \left(\frac{p^n_{Si_{KX}}}{p^{n-1}_{Si_{KX}}} \right) =$$

$$\frac{\eta}{p^{n-1}_{Si_{KX}}} p_{\Theta_1|Si_{KX}} p_{\Theta_2|Si_{KX}} \cdots p_{\Theta_n|Si_{KX}} p^n_{Si_{KX}} =$$

$$\frac{\eta}{p^{n-1}_{Si_{KX}}} \prod_{i=1}^{n} p_{\Theta_i|Si_{KX}} p_{Si_{KX}}, \quad (5.3.13)$$

essendo per ipotesi $p_{Si_{KX}}$ uniforme questo può essere inglobato all'interno del fattore η e quindi segue

$$\eta \prod_{i=1}^{n} p_{\Theta_i|Si_{KX}} p_{Si_{KX}} = \eta \prod_{i=1}^{n} p_{Si_{KX}|\Theta_i}. \quad (5.3.14)$$

□

5.3.2 Pseudocodice per il calcolo della likelihood

La (5.3.10) può essere implementata in modo differenziale, definendo

$$L_n = \prod_{i=1}^{n} p_{Si_{KX}|\Theta_i}. \quad (5.3.15)$$

vale la seguente proposizione.

Proposizione 5.1. *Se vale la* (5.3.15) $\Rightarrow L_n = L_{n-1} \cdot p_{Si_{KX}|\Theta_n}$

Dimostrazione. Per definizione si ha che

$$L_n = \prod_{i=1}^{n} p_{Si_{KX}|\Theta_i}, \quad (5.3.16)$$

che può essere ulteriormente riscritta come

$$\prod_{i=1}^{n} p_{Si_{KX}|\Theta_i} = \left(\prod_{i=1}^{n-1} p_{Si_{KX}|\Theta_i} \right) p_{Si_{KX}|\Theta_n}. \quad (5.3.17)$$

Nella (5.3.17) si ha

$$\prod_{i=1}^{n-1} p_{Si_{KX}|\Theta_i} = L_{n-1}, \quad (5.3.18)$$

segue la tesi

$$L_n = L_{n-1} \cdot p_{Si_{KX}|\Theta_n} \quad (5.3.19)$$

come volevasi dimostrare. □

Algoritmo 5.4 Calcolo probabilità chiave corretta per un attacco multiplo

1: **function** PROB_KEY_MULTIPLE_ATTACK(k_x,$e_{x,list}$,$e^*_{x,list}$,$i'_{x,list}$,$o'_{x,list}$)
2: $p \leftarrow 1$
3: **repeat**
4: $e_x \leftarrow e_{x,list}.pop()$
5: $e^*_x \leftarrow e^*_{x,list}.pop()$
6: $i'_x \leftarrow i'_{x,list}.pop()$
7: $o'_x \leftarrow o'_{x,list}.pop()$
8: $p \leftarrow p *$ PROB_KEY_SINGLE_ATTACK_V2$(k_x, e_x, e^*_x, i'_x, o'_x)$
9: **until** NOT_EMPTY($e_{x,list}$,$e^*_{x,list}$,$i'_{x,list}$,$o'_{x,list}$)
10: **end function**

Algoritmo 5.5 Calcolo likelihood per un attacco multiplo

1: **function** LIKELIHOOD_KEY_MULTIPLE_ATTACK($e_{x,list}$,$e^*_{x,list}$,$i'_{x,list}$,$o'_{x,list}$)
2: $p[2^6] \leftarrow 1$
3: **for** $k_x = 0_x \rightarrow 3F_x$ **do**
4: $p[k_x] =$ PROB_KEY_MULTIPLE_ATTACK$(k_x, e_{x,list}, e^*_{x,list}, i'_{x,list}, o'_{x,list})$
5: **end for**
6: **return** p;
7: **end function**

La (5.3.19) è una successione di likelihood, definita ricorsivamente, necessita quindi una condizione base. Per condizione iniziale si può porre

$$L_0 = 1, \tag{5.3.20}$$

e quindi la likelihood di un attacco multiplo ad una singola S-Box è la soluzione della relazione di ricorrenza

$$\begin{cases} L_n = L_{n-1} \cdot p_{Si_{KX}|\Theta_n} \\ L_0 = 1 \end{cases} . \tag{5.3.21}$$

Quanto detto guida in modo diretto ad una implementazione del codice necessario a calcolare la probabilità che una chiave Si_{KX} sia corretta dopo un numero n di attacchi.

La riga 1 inizializza la probabilità, tra le righe 3 e 9 avviene invece l'aggiornamento delle probabilità secondo quanto dettato dal sistema (5.3.21). Il calcolo della likelihood in caso di attacco multiplo si può effettuare implementando lo pseudocodice in 5.5

5.3.3 Ottimizzazione

Si può calcolare la likelihood utilizzando la seguente proposizione.

Proposizione 5.2. *Sia L_{n-1} la likelihood derivata dopo $n-1$ attacchi alla S-Box $Si(\cdot)$; se risulta*

$$L_{n-1} = 0 \tag{5.3.22}$$

per una qualche chiave $k_x \in \mathbb{Z}_2^6$ allora per la medesima chiave si ha che

$$L_n = 0. \tag{5.3.23}$$

Algoritmo 5.6 Calcolo probabilità chiave corretta per un attacco multiplo, ottimizzato.

```
1: function PROB_KEY_MULTIPLE_ATTACK_V2(k_x,e_{x,list},e*_{x,list},i'_{x,list},o'_{x,list})
2:     p ← 1
3:     repeat
4:         e_x ← e_{x,list}.pop()
5:         e*_x ← e*_{x,list}.pop()
6:         i'_x ← i'_{x,list}.pop()
7:         o'_x ← o'_{x,list}.pop()
8:         p ← p * PROB_KEY_SINGLE_ATTACK_V2(k_x, e_x, e*_x, i'_x, o'_x)
9:     until p ≠ 0
10: end function
```

Algoritmo 5.7 Calcolo likelihood per un attacco multiplo, ottimizzato con liste.

```
1: function LIKELIHOOD_KEY_MULTIPLE_ATTACK_V2(e_{x,list},e*_{x,list},i'_{x,list},o'_{x,list})
2:     p_list ← ∅
3:     for k_x = 0_x → 3F_x do
4:         p = PROB_KEY_MULTIPLE_ATTACK_V2(k_x, e_{x,list}, e*_{x,list}, i'_{x,list}, o'_{x,list})
5:         if (p ≠ 0) then p_list.add(k_x)
6:         end if
7:     end for
8:     return p_list;
9: end function
```

Dimostrazione. Ipotizzando di valutare la L_{n-1} in una chiave k_x per la quale essa risulti nulla per la proposizione 5.1 risulta che

$$L_n = L_{n-1} p_{Si_{KX}|\Theta_n}, \tag{5.3.24}$$

tuttavia poichè per ipotesi $L_{n-1} = 0$ in corrispondenza di k_x si avrà che

$$L_n = 0 \tag{5.3.25}$$

in corrispondenza di k_x. □

Una implementazione ottimizzata sfrutta la proposizione 5.2 per evitare il calcolo della likelihood in corrispondenza di chiavi per i quali essa risulti sicuramente nulla.

L'algoritmo 5.6 è sostanzialmente identico all'algoritmo 5.4, se non per il fatto che lo stop dell'algoritmo è dato dal prima probabilità diversa pari a 0 che si incontra scorrendo le liste date in input, piuttosto che aspettare che le liste divengano vuote. Il listato 5.7 ottimizza la rappresentazione del dominio della likelihood, quelli diversi da 0 non vengono inseriti nella lista p_{list}. Siccome la distribuzione L_n è uniforme nel sottoinsieme di chiavi dove questa è diversa da 0, l'uso delle liste ottimizza la rappresentazione delle likelihood. Data una p_{list} per poter calcolare la probabilità di correttezza di una data chiave bisogna scorrere la lista e vedere se tale chiave è presente. Per aggiornare la likelihood si può usare l'algoritmo in 5.8.

Algoritmo 5.8 Aggiornamento della likelihood.

```
1: function UPDATE_LIKELIHOOD(e_x, e*_x, i'_x, o'_x, p_list)
2:     p_list,rval ← ∅
3:     for all k_x ∈ p_list do
4:         p = PROB_KEY_SINGLE_ATTACK_V2(k_x, e_x, e*_x, i'_x, o'_x)
5:         if p ≠ 0 then
6:             p_list,rval ← p_list,rval.add(k_x)
7:         end if
8:     end for
9:     return p_list,rval
10: end function
```

5.4 Attacco alla funzione di Feistel

Per attaccare la funzione di Feistel si può generalizzare l'approccio definito in precedenza. Per l'attacco alla funzione di Feistel effettuato in ambito chosen plaintext il testo scelto è un ingresso alla funzione di Feistel Z_X, ed è possibile osservare l'output Y_X. Nella sezione precedente visto che Si_{EX} era un parametro libero la sua distribuzione è stata assunta uniforme in $\mathbb{Z}_2^6$. Quando però si attacca la funzione di Feistel Si_{EX} e Sj_{EX}, con $i \neq j$ non sono in generale indipendenti, noto Z_X, quindi non è possibile costruire una rete Bayesiana che attacca le singole $Si(\cdot)$ in modo indipendente, in quanto i vari gruppi di sei bit Si_{EX} della funzione di espansione non sono tra loro indipendenti. Noto il funzionamento della funzione di espansione questo aspetto è intuitivo, ma formalmente viene trattato in appendice, con le sue conseguenze nella costruzione della Rete Bayesiana. In appendice, inoltre, viene dimostrata la seguente proposizione.

Proposizione 5.3. *Sia $E(\cdot)$ $\mathbb{Z}_2^{32} \to \mathbb{Z}_2^{48}$ la funzione di espansione definita nella funziona di Feistel istanziata dal DES allora le seguenti affermazioni sono valide.*

1. *$E(\cdot)$ è lineare;*
2. *$E(\cdot)$ è iniettiva;*
3. *$E(\cdot)$ non è suriettiva.*

Dalla proposizione 5.3 segue che gli output S_{EX} della funzione di espansione sono uniformemente distribuiti in un dominio $D_{S_{EX}} \subset \mathbb{Z}_2^{48}$. Vale inoltre il 4.5 quindi risulta

$$p_{S_{EX}|Z}(e, z) = \delta_{48}(e \oplus z). \tag{5.4.1}$$

L'approccio utilizzato per attaccare le singole S-Box $Si(\cdot)$ non è quindi generalizzabile per attaccare la funzione di Feistel istanziata dal DES in quanto la funzione di espansione produce output i cui bit non sono tra loro indipendenti, ecco perchè si deve riformulare il tutto per costruire una nuova rete per attaccare la funzione di Feistel. In breve la proposizione 5.3 permette di inferire che da due input statisticamente indipendenti della funzione di espansione si hanno due output statisticamente indipendenti, non solo ma essi sono uniformemente distribuiti. Di seguito viene indicata la notazione utilizzata

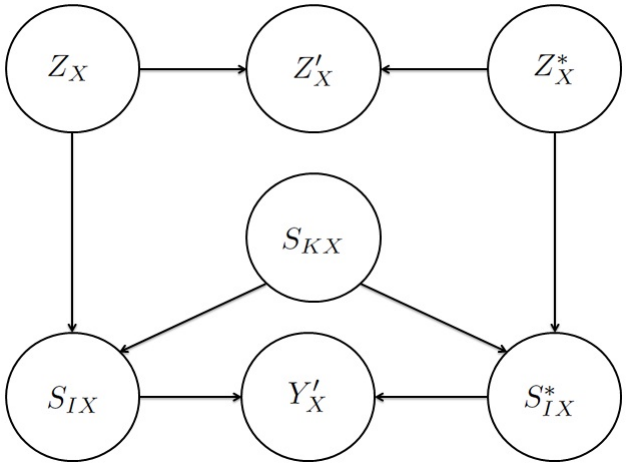


Figura 5.8: Rete per l'attacco alla funzione di Feistel istanziata dal DES

Z_X, Z^*_X : Due possibili testi in chiaro di 32 bit in ingresso alla funzione di Feistel del round X;

S_{KX} : Una sottochiave di 48 bit che parametrizza il comportamento della funzione di Feistel;

S_{IX}, S^*_{IX} : Due ingressi di 48 bit alla S-Box del round X;

Y_X, Y^*_X : Due output della funzione di Feistel, di 32 bit;

$P(\cdot)$: Funzione di permutazione utilizzata nella funzione di Feistel, un mapping $P(\cdot) : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$;

$E(\cdot)$: Funzione di espansione utilizzata nella funzione di Feistel, un mapping $E(\cdot) : \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{48}$;

Z'_X, Y'_X : Rispettivamente $Z'_X = Z_X \oplus Z^*_X$ e $Y'_X = Y_X \oplus Y^*_X$;

$S(\cdot)$: S-Box, $S(\cdot) : \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{32}$.

Con riferimento alla notazione appena introdotta le seguenti relazioni sono inoltre valide le seguenti

$$Z'_X = Z_X \oplus Z^*_X, \tag{5.4.2}$$
$$S_{IX} = E(Z_X) \oplus S_{KX}, \tag{5.4.3}$$
$$S^*_{IX} = E(Z^*_X) \oplus S_{KX}, \tag{5.4.4}$$
$$Y_X = P(S(S_{IX})), \tag{5.4.5}$$
$$Y^*_X = P(S(S^*_{IX})), \tag{5.4.6}$$
$$Y'_X = Y_X \oplus Y^*_X. \tag{5.4.7}$$

Dalle equazioni precedenti segue la struttura della rete seguente in figura 5.8.

Assumendo che i nodi Z_X, Z^*_X, S_{KX} siano uniformemente distribuiti nel loro dominio di definizione si ha che le seguenti densità di probabilità sono quelle da associare ai vari nodi della rete.

$$p_{S_{KX}}(k) = \frac{1}{2^{48}}, \tag{5.4.8}$$

$$p_{Z_X}(z) = \frac{1}{2^{32}}, \tag{5.4.9}$$

$$p_{Z^*_X}(z^*) = \frac{1}{2^{32}}, \tag{5.4.10}$$

$$p_{Z'_X|Z_X,Z^*_X}(z', z, z^*) = \delta_{32}(z' \oplus z \oplus z^*), \tag{5.4.11}$$

$$p_{Z'_X|Z_X,Z^*_X}(z', z, z^*) = \delta_{32}(z' \oplus z \oplus z^*), \tag{5.4.12}$$

$$p_{S_{IX}|Z_X,S_{KX}}(i, z, k) = \delta_{48}(i \oplus E(z) \oplus k), \tag{5.4.13}$$

$$p_{S^*_{IX}|Z^*_X,S_{KX}}(i^*, z^*, k) = \delta_{48}(i^* \oplus E(z^*) \oplus k), \tag{5.4.14}$$

$$p_{Y'_X|S_{IX},S^*_{IX}}(y', i, i^*) = \delta_{32}(y' \oplus P(S(i) \oplus S(i^*))). \tag{5.4.15}$$

La dimostrazione della correttezza delle precedenti è analoga a quella del teorema 5.1. La sottochiave utilizzata dalla funzione di Feistel sarà quella chiave tale che

$$\hat{k} = argmax_{k \in \mathbb{Z}_2^{48}} \left(p_{S_{KX}|Z_X,Z^*_X,Z'_X,Y'_X}(k, z, z^*, z', y') \right), \tag{5.4.16}$$

quindi vale la pena soffermarsi sull'espressione della likelihood

$$p_{S_{KX}|Z_X,Z^*_X,Z'_X,Y'_X}(k, z, z^*, z', y'), \tag{5.4.17}$$

e anche qui verranno sottointesi gli argomenti rispetto ai quali si calcola la likelihood. Nel seguito verranno dimostrate alcune proposizioni, ove si utilizzerà una notazione introdotta in [1], ove gli indici delle sommatorie sono delle condizioni su dei parametri da cui dipendono i vari addendi delle sommatorie. Nel caso in questione molto spesso le condizioni sotto le quali vengono effettuate le somme saranno indicate dalla verifica di alcuni sistemi lineari.
Vale la seguente proposizione

Proposizione 5.4. *Per la rete in figura 5.8 vale che*

$$p_{Y'_X|Z_X,Z^*_X} = \frac{1}{2^{48}} \sum_{\substack{S_{KX} \in \mathbb{Z}_2^{48} \\ S_{IX} = S_{KX} \oplus E(Z_X) \\ S^*_{IX} = S_{KX} \oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}}. \tag{5.4.18}$$

Dimostrazione. Data la struttura della rete valgono le uguaglianze

$$p_{Y'_X|Z_X,Z^*_X} = \sum_{S_{IX},S^*_{IX},S_{KX}} p_{Y'_X,S_{IX},S^*_{IX},S_{KX}|Z_X,Z^*_X} =$$
$$\sum_{S_{IX},S^*_{IX},S_{KX}} \left[p_{Y'_X|S_{IX},S^*_{IX},S_{KX},Z_X,Z^*_X} \times p_{S_{IX},S^*_{IX}|S_{KX},Z_X,Z^*_X} \times p_{S_{KX}|Z_X,Z^*_X} \right]; \tag{5.4.19}$$

nella (5.4.19) si ha la validità delle uguaglianze seguenti

$$p_{Y'_X|S_{IX},S^*_{IX},S_{KX},Z_X,Z^*_X} = p_{Y'_X|S_{IX},S^*_{IX}}, \tag{5.4.20}$$

$$p_{S_{IX},S^*_{IX}|S_{KX},Z_X,Z^*_X} = p_{S_{IX}|S_{KX},Z_X} p_{S^*_{IX}|S_{KX},Z^*_X}, \tag{5.4.21}$$

$$p_{S_{KX}|Z_X,Z^*_X} = p_{S_{KX}} = \frac{1}{2^{48}}. \tag{5.4.22}$$

Sostituendo la (5.4.20), (5.4.21) e (5.4.22) nell'ultimo membro della (5.4.19) si ottiene

$$p_{Y'_X|Z_X,Z^*_X} = \frac{1}{2^{48}} \sum_{S_{IX},S^*_{IX},S_{KX}} \left[p_{Y'_X|S_{IX},S^*_{IX}} \times p_{S_{IX}|S_{KX},Z_X} \times p_{S^*_{IX}|S_{KX},Z^*_X} \right]. \quad (5.4.23)$$

Visto che risulta

$$p_{S_{IX}|S_{KX},Z_X} = \delta_{48}(S_{IX} \oplus S_{KX} \oplus E(Z_X)), \quad (5.4.24)$$
$$p_{S^*_{IX}|S_{KX},Z^*_X} = \delta_{48}(S^*_{IX} \oplus S_{KX} \oplus E(Z^*_X)), \quad (5.4.25)$$

il secondo membro della (5.4.23) diviene

$$\frac{1}{2^{48}} \sum_{S_{IX},S^*_{IX},S_{KX}} \left[p_{Y'_X|S_{IX},S^*_{IX}} \times p_{S_{IX}|S_{KX},Z_X} \times p_{S^*_{IX}|S_{KX},Z^*_X} \right] =$$

$$\frac{1}{2^{48}} \sum_{S_{IX},S^*_{IX},S_{KX}} \left[p_{Y'_X|S_{IX},S^*_{IX}} \times \delta_{48,48}(S_{IX} \oplus S_{KX} \oplus E(Z_X), S^*_{IX} \oplus S_{KX} \oplus E(Z^*_X)) \right] =$$

$$\frac{1}{2^{48}} \sum_{\substack{S_{KX} \in \mathbb{Z}_2^{48} \\ S_{IX} = S_{KX} \oplus E(Z_X) \\ S^*_{IX} = S_{KX} \oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}}. \quad (5.4.26)$$

□

Vale inoltre la seguente

Proposizione 5.5. *Per la rete in figura 5.8 vale che*

$$p_{Y'_X|Z'_X} = \frac{1}{2^{32}} \sum_{\substack{Z_X \in \mathbb{Z}_2^{32} \\ Z^*_X = Z_X \oplus Z'_X}} p_{Y'_X|Z_X,Z^*_X}. \quad (5.4.27)$$

Dimostrazione.

$$p_{Y'_X|Z'_X} = \sum_{Z_X,Z^*_X} p_{Y'_X,Z_X,Z^*_X|Z'_X} = \sum_{Z_X,Z^*_X} \left[\frac{p_{Y'_X|Z_X,Z^*_X} \times p_{Z'_X|Z_X,Z^*_X} \times p_{Z_X} \times p_{Z^*_X}}{p_{Z'_X}} \right] \quad (5.4.28)$$

il lemma 4.3 implica

$$p_{Z'_X} = \frac{1}{2^{32}}, \quad (5.4.29)$$

mentre per ipotesi sulla rete

$$p_{Z_X} = p_{Z^*_X} = \frac{1}{2^{32}}, \quad (5.4.30)$$

e quindi si ha l'uguaglianza

$$\sum_{Z_X,Z^*_X} \left[\frac{p_{Y'_X|Z_X,Z^*_X} \times p_{Z'_X|Z_X,Z^*_X} \times p_{Z_X} \times p_{Z^*_X}}{p_{Z'_X}} \right] =$$

$$\frac{1}{2^{32}} \sum_{Z_X,Z^*_X} \left[p_{Y'_X|Z_X,Z^*_X} \times p_{Z'_X|Z_X,Z^*_X} \right]. \quad (5.4.31)$$

Per struttura della rete si ha che

$$p_{Z'_X|Z_X,Z^*_X} = \delta_{32}(Z'_X \oplus Z_X \oplus Z^*_X), \tag{5.4.32}$$

e quindi

$$\frac{1}{2^{32}} \sum_{Z_X,Z^*_X} \left[p_{Y'_X|Z_X,Z^*_X} \times p_{Z'_X|Z_X,Z^*_X} \right] = \frac{1}{2^{32}} \sum_{\substack{Z_X \in \mathbb{Z}_2^{32} \\ Z^*_X = Z_X \oplus Z'_X}} p_{Y'_X|Z_X,Z^*_X}. \tag{5.4.33}$$

□

Proposizione 5.6. *Per la rete in figura 5.8 vale*

$$p_{Y'_X|Z_X,Z^*_X,S_{KX}} = \sum_{\substack{S_{IX} = S_{KX} \oplus E(Z_X) \\ S^*_{IX} = S_{KX} \oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}} \tag{5.4.34}$$

Dimostrazione. Basta osservare che

$$p_{Y'_X|Z_X,Z^*_X,S_{KX}} = \sum_{S_{IX},S^*_{IX}} p_{Y'_X,S_{IX},S^*_{IX}|Z_X,Z^*_X,S_{KX}} =$$

$$\sum_{S_{IX},S^*_{IX}} \left[p_{Y'_X|S_{IX},S^*_{IX}} \times p_{S_{IX}|Z_X,S_{KX}} \times p_{S^*_{IX}|Z^*_X,S_{KX}} \right] =$$

$$\sum_{S_{IX},S^*_{IX}} \left[p_{Y'_X|S_{IX},S^*_{IX}} \times \delta_{48,48}(S_{IX} \oplus E(Z_X) \oplus S_{KX}, S^*_{IX} \oplus E(Z^*_X) \oplus S_{KX}) \right] =$$

$$\sum_{\substack{S_{IX} = S_{KX} \oplus E(Z_X) \\ S^*_{IX} = S_{KX} \oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}} \tag{5.4.35}$$

□

5.4.1 Espressione della likelihood

La teoria sviluppata permette di dimostrare

Teorema 5.3. *Per la rete in figura 5.8 si ha che*

$$p_{S_{KX}|Z_X,Z^*_X,Z'_X,Y'_X} = \frac{\sum_{\substack{S_{IX} = S_{KX} \oplus E(Z_X) \\ S^*_{IX} = S_{KX} \oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}}}{\sum_{\substack{S_{KX} \in \mathbb{Z}_2^{48} \\ S_{IX} = S_{KX} \oplus E(Z_X) \\ S^*_{IX} = S_{KX} \oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}}} \tag{5.4.36}$$

Dimostrazione.

$$p_{S_{KX}|Z_X,Z^*_X,Z'_X,Y'_X} = \frac{p_{Y'_X,Z'_X|S_{KX},Z_X,Z^*_X} \times p_{S_{KX}} \times p_{Z_X} \times p_{Z^*_X}}{p_{Y'_X,Z'_X|Z_X,Z^*_X} \times p_{Z_X} \times p_{Z^*_X}} =$$

$$\frac{p_{Y'_X|Z_X,Z^*_X,S_{KX}} \times p_{Z'_X|Z_X,Z^*_X} \times p_{S_{KX}}}{p_{Y'_X|Z_X,Z^*_X} \times p_{Z'_X|Z_X,Z^*_X}} = \frac{p_{Y'_X|Z_X,Z^*_X,S_{KX}} \times p_{S_{KX}}}{p_{Y'_X|Z_X,Z^*_X}} =$$

$$\frac{1}{2^{48}} \frac{p_{Y'_X|Z_X,Z^*_X,S_{KX}}}{p_{Y'_X|Z_X,Z^*_X}}, \tag{5.4.37}$$

Algoritmo 5.9 Probabilità chiave corretta dopo un singolo attacco

1: **function** PROB_KEY_SINGLE_ATTACK($k_x, z_x, z_x^*, z'_x, y'_x$)
2: $i_x = k_x \oplus E(z_x)$
3: $i_x^* = k_x \oplus E(z_x^*)$
4: **return** $\delta_{32}(y'_x \oplus P(S(i_x)) \oplus P(S(i_x^*)))\delta_{32}(z'_x \oplus z_x \oplus z_x^*)$
5: **end function**

per la proposizione 5.4 e 5.6 risulta

$$p_{Y'_X|Z_X,Z^*_X,S_{KX}} = \sum_{\substack{S_{IX}=S_{KX}\oplus E(Z_X)\\ S^*_{IX}=S_{KX}\oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}}, \tag{5.4.38}$$

$$p_{Y'_X|Z_X,Z^*_X} = \frac{1}{2^{48}} \sum_{\substack{S_{KX}\in\mathbb{Z}_2^{48}\\ S_{IX}=S_{KX}\oplus E(Z_X)\\ S^*_{IX}=S_{KX}\oplus E(Z^*_X)}} p_{Y'_X|S_{IX},S^*_{IX}}. \tag{5.4.39}$$

e quindi si ha la tesi. □

5.4.2 Pseudocodice per il calcolo della likelihood

L'attacco della funzione di Feistel a differenza delle S-Box ha domini molto più grandi, di conseguenza il problema dell'ottimizzazione è molto sentito. Di seguito viene proposto l'algoritmo di aggiornamento della likelihood sulle chiavi. La filosofia è la stessa dell'attacco alle singole S-Box, quindi si riportano in modo diretto gli algoritmi.

Gli input dell'algoritmo 5.9 sono dei valori quattro valori a 32 bit (z_x, z_x^*, z'_x, y'_x), anche qui sono due input della funzione di espansione, un input XOR e l'output XOR, k_x è un valore a 48 bit che sarebbe la sottochiave da cercare. In riga 2 e 3 si ha il calcolo degli input alle $S-Box$, in base agli input dell'algoritmo stesso. In riga 4 si ha l'output secondo quanto dettato dal teorema 5.3, come al solito il valore ritornato è quello di una probabilità non normalizzata, ed è lecito utilizzare questo pseudocodice in virtù del fatto che la distribuzione è uniforme in una opportuna restrizione di $\mathbb{Z}_2^{48}$ (questa affermazione segue da motivi analoghi a quanto detto in precedenza per le S-Box). Lo scopo dell'algoritmo 5.10 è quello di aggiornare una lista contenente i valori per i quali la likelihood è diversa da 0, questa lista è p_{list}, l'algoritmo elabora i vari elementi della lista tramite l'algoritmo 5.9 e ne ritorna una lista aggiornata. L'aggiornamento avviene in base alle evidenze a disposizione. L'algoritmo 5.11 effettua aggiornamenti successivi della lista, in base a più evidenze.

5.4.3 Relazione tra la Feistel B-Net e la S-Box B-Net

Esiste una importante relazione che giustificherà nel seguito alcune implementazioni sperimentali per la stima della likelihood per la funzione di Feistel dopo alcuni attacchi. Vengono di seguito esposte due definizioni ben poste la prima di queste riguarda la S-Box B-NET.

Definizione 5.1. *Sia $Si'_{IX} \in \mathbb{Z}_2^6$, $i \in \{1,\dots,8\}$ e sia $Si'_{OX} \in \mathbb{Z}_2^4$, $i \in \{1,\dots,8\}$. Diremo che Si'_{IX} può causare Si'_{OX}, e verrà indicato con $Si'_{IX} \to Si'_{OX}$ se esiste*

Algoritmo 5.10 Aggiornamento likelihood dopo un singolo attacco

```
1: function UPDATE_LIKELIHOOD(z_x, z*_x, z'_x, y'_x, p_list)
2:     p_list,rval ← ∅
3:     for all k_x ∈ p_list do
4:         p ← PROB_KEY_SINGLE_ATTACK(k_x, z_x, z*_x, z'_x, y'_x)
5:         if p ≠ 0 then
6:             p_list,rval.add(k_x)
7:         end if
8:     end for
9:     return p_list,rval
10: end function
```

Algoritmo 5.11 Calcolo likelihood dopo attacchi multipli

```
1: function LIKELIHOOD_MULTIPLE_ATTACK(z_x,list, z*_x,list, z'_x,list, y'_x,list)
2:     p_list ← Z_2^48
3:     repeat
4:         z_x = z_x,list.pop()
5:         z*_x = z*_x,list.pop()
6:         z'_x = z'_x,list.pop()
7:         y'_x = y_x,list.pop()
8:         p_list ← UPDATE_LIKELIHOOD(z_x, z*_x, z'_x, y'_x, p_list)
9:     until liste_non_vuote
10:    return p_list
11: end function
```

una coppia $(Si_{EX}, Si_{KX}) \in \mathbb{Z}_2^6 \times \mathbb{Z}_2^6$ *tale che l'equazione*

$$Si'_{OX} = Si(Si_{EX} \oplus Si_{KX}) \oplus Si(Si_{EX} \oplus Si'_{IX} \oplus Si_{KX}) \tag{5.4.40}$$

sia soddisfatta.

La seconda definizione riguarda la Feistel B-Net

Definizione 5.2. *Sia* $Z'_X \in \mathbb{Z}_2^{32}$*, e sia* $Y'_X \in \mathbb{Z}_2^{48}$*. Diremo che* Z'_X *può causare* Y'_X*, e verrà indicato con* $Z'_X \to Y'_X$ *se esiste una coppia* $(Z_X, S_{KX}) \in \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32}$ *tale che l'equazione*

$$Y'_X = S(E(Z_X) \oplus S_{KX}) \oplus S(E(Z_X \oplus Z'_X) \oplus S_{KX}) \tag{5.4.41}$$

sia soddisfatta.

Il seguente risultato è di semplice dimostrazione

Proposizione 5.7. *Sia* $Z'_X \in \mathbb{Z}_2^{32}$ *tale che valga l'uguaglianza*

$$E(Z'_X) = (S1'_{IX}, \ldots, S8'_{IX}) \tag{5.4.42}$$

con $Si'_{IX} \in \mathbb{Z}_2^6$ *,*$i = 1, \ldots, 8$ *e sia* $Y'_X \in \mathbb{Z}_2^{32}$ *tale che*

$$P^{-1}(Y'_X) = (S1'_{OX}, \ldots, S8'_{OX}) \tag{5.4.43}$$

con $Si'_{OX} \in \mathbb{Z}_2^4$*,* $i = 1, \ldots, 8$ *allora vale che*

$$Si'_{IX} \to Si'_{OX} \tag{5.4.44}$$

per ogni $i = 1, \ldots, 8$*.*

vale anche la proposizione inversa

Proposizione 5.8. *Siano $Si'_{IX} \in \mathbb{Z}_2^6$, $i = 1, \ldots, 8$ tali esista un $Z'_X \in \mathbb{Z}_2^{32}$ per la quale valga l'uguaglianza*

$$E(Z'_X) = (S1'_{IX}, \ldots, S8'_{IX}) \tag{5.4.45}$$

e siano $Si'_{OX} \in \mathbb{Z}_2^4$, $i = 1, \ldots, 8$ tali che esista un $Y'_X \in \mathbb{Z}_2^{32}$ per il quale valga l'uguaglianza

$$P^{-1}(Y'_X) = (S1'_{OX}, \ldots, S8'_{OX}) \tag{5.4.46}$$

se per ogni i vale che $Si'_{IX} \to Si'_{OX}$ allora vale anche che $Z'_X \to Y'_X$.

Si può infine dimostrare la seguente proposizione

Proposizione 5.9. *La seguente uguaglianza è valida*

$$p_{Y'_X|Z'_X}(y', z') = \prod_{i=1}^{8} p_{Si'_{OX}|Si'_{IX}}(o'_i, i'_i). \tag{5.4.47}$$

Essenzialmente l'ultima proposizione segue dalle 5.7 e 5.8. L'ultima proposizione è utile quando si deve calcolare la statistica di $p_{Y'_X|Z'_X}$; dalla proposizione 5.5 segue che il numero di operazioni necessarie a calcolarla siano di 2^{32} somme, mentre utilizzando il secondo membro della (5.9) queste si riducono a $8 \cdot 2^6 = 2^9$, nettamente inferiore. Inoltre la proposizione 5.9 è utile quando si devono generare valori secondo la distribuzione $p_{Y'_X|Z'_X}$, in quanto si dimostrerà in seguito richiedere in generale campioni casuali a 48 bit, mentre utilizzando il secondo membro della (5.9) basterà prelevare 8 campioni di 6 bit l'uno.

Capitolo 6

Likelihood per round multipli del DES

6.1 Preliminari

Il problema principale nell'attaccare il DES consiste nel fatto che esso è un sistema a round multipli, di conseguenza la difficoltà nella costruzione di una rete bayesiana che possa descrivere il problema da un punto di vista statistico diviene maggiore, e ugualmente difficile è la sua risoluzione. In questo capitolo verranno esposti alcuni esempi di reti, determinadone la soluzione, e implementando quest'ultima in modo da permettere di attaccare più round del DES. In [2] l'attacco a più round del DES si avvale del concetto di caratteristica, una caratteristica è fondamentalmente una lista che descrive la propagazione delle differenze input/output attraverso le diverse funzioni di Feistel istanziate dal DES nelle diverse fasi. In [2] l'attacco al DES viene effettuato sotto l'ipotesi di chosen plaintext attack, quindi i dati sono due coppie di testo in chiaro e le rispettive coppie di testo cifrato. Date due coppie di testo in chiaro la loro differenza è nota, ed essendo note anche i testi cifrati sarà anche nota la differenza di questi. L'attacco viene effettuato attaccando in ordine i vari round del DES, partendo dall'ultimo verso il primo, ed essenzialmente si ipotizza che si propaghino delle determinate differenze che sono valide con una elevata probabilità. Con propagazione delle differenze si intende l'insieme delle differenze delle trasformazioni intermedie dei vari round, in [2] questa propagazione viene indicata come *caratteristica*. In questo capitolo, per uniformità con l'approccio esposto in [2] lo XOR tra due possibili ingressi alla funzione di Feistel del round X è indicata con λ_I^X mentre lo XOR tra due uscite della funzione di Feistel del medesimo round è indicato con λ_O^X, in breve rispetto alla rete in figura 5.8 vengono rinominati i nodi Z'_X e Y'_X rispettivamente con λ_I^X e λ_O^X. Avvalendosi della rete in figura 5.8 si può dimostrare la seguente proposizione.

Proposizione 6.1. *Per ogni Feistel B-NET vale che*

$$p_{\lambda_O^X|\lambda_I^X,S_{KX}} = \frac{1}{2^{32}} \sum_{\begin{cases} Z_X \in \mathbb{Z}_2^{32} \\ S_{IX}=E(Z_X)\oplus S_{KX} \\ S^*_{IX}=E(Z_X\oplus\lambda_I^X)\oplus S_{KX} \end{cases}} p_{\lambda_O^X|S_{IX},S^*_{IX}}. \qquad (6.1.1)$$

In questo capitolo si esporrà una rete generale per attaccare N-Round DES-Like, della quale verrà dimostrata la correttezza; di tale rete verrà determinata la soluzione e infine si passerà ad analizzare alcuni esempi implementativi.

6.2 Rete generale

Ipotizziamo di avere un sistema DES-Like, con sottochiavi indipendenti (stessa ipotesi di [2], fatta per semplificare il trattamento teorico), e che a tale sistema venga dato in input un testo in chiaro P e che venga prodotto un testo cifrato T (notazione utilizzata per uniformità co [2]), e successivamente un testo in chiaro P^*, che produca un testo cifrato T^*, e si considera ogni testo composto da una parte sinistra e destra

$$P = (L, R), \tag{6.2.1}$$
$$P^* = (L^*, R^*), \tag{6.2.2}$$
$$T = (l, r), \tag{6.2.3}$$
$$T^* = (l^*, r^*). \tag{6.2.4}$$

Si costruiscano inoltre gli XOR dei testi in chiaro e cifrati:

$$P \oplus P^* = (L \oplus L^*, R \oplus R^*) = (L', R'), \tag{6.2.5}$$
$$T \oplus T^* = (l \oplus l^*, r \oplus r^*) = (l', r'). \tag{6.2.6}$$

Ipotizzando che P e P^* siano scelti in modo indipendente, se si indica con $Z_1 = R$ l'ingresso alla funzione di Feistel del primo round, dovuto all'ingresso P e con $Z_1^* = R^*$ quello dovuto a P^* si ha che questi a loro volta sono indipendenti, il primo round risulta quindi caratterizzato dalla differenza in input

$$\lambda_I^1 = Z_1 \oplus Z_1^*; \tag{6.2.7}$$

Se si indica con Y_1 e Y_1^* rispettivamente gli output della funzione di Feistel del medesimo round si ha che la differenza in output caratterizza il primo round è

$$\lambda_O^1 = Y_1 \oplus Y_1^*. \tag{6.2.8}$$

Secondo la proposizione 6.1, l'output λ_O^1 è statisticamente dipendente da λ_I^1 e dalla chiave S_{K1} al primo round. Il primo round produrrà, per ognuno dei due input, un blocco di 64 bit la cui parte sinistra e destra sono le seguenti:

$$T_{\text{round 1}} = (Z_1, Y_1 \oplus L_1), \tag{6.2.9}$$
$$T^*_{\text{round 1}} = (Z_1^*, Y_1^* \oplus L_1^*). \tag{6.2.10}$$

In ingresso alla funzione di Feistel del secondo round si avrà la differenza

$$\lambda_I^2 = (Y_1 \oplus L) \oplus (Y_1^* \oplus L^*) = (Y_1 \oplus Y_1^*) \oplus (L \oplus L^*) = \lambda_O^1 \oplus L'; \tag{6.2.11}$$

il secondo round produrrà una differenza in output λ_O^2 che dipenderà appunto da λ_I^2 e S_{K2} secondo la statistica dettata dalla proposizione 6.1. Questo ragionamento si può ripetere per n round iterati, con n generico.

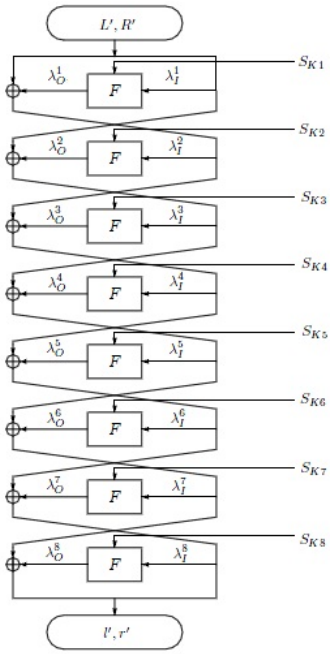


Figura 6.1: Caratteristica di un DES a 8 round secondo la notazione adottata in [2].

I contenuti della discussione appena tenuta sono riassunti in figura 6.1, che mostra con uno schema blocchi come si propaghino le differenze da una funzione di Feistel all'altra. La statistica che lega λ_O^X dato λ_I^X e una chiave S_{KX} è quella della proposizione 6.1. Si osservino i vari nodi XOR che stabiliscono una relazione lineare tra due ingressi e l'uscita, infatti osservando la figura 6.1 si nota che il seguente sistema risulta sempre verificato:

$$\begin{cases} R' = R \oplus R^* \\ L' = L \oplus L^* \\ \lambda_I^1 = R' \\ \lambda_I^2 = L' \oplus \lambda_O^1 \\ \vdots \\ \lambda_I^X = \lambda_O^{X-1} \oplus \lambda_I^{X-2} \\ \vdots \\ r' = \lambda_I^n \\ l' = \lambda_O^n \oplus \lambda_I^{n-1} \end{cases}, \tag{6.2.12}$$

dove $X = 3, \ldots, n$. Il sistema (6.2.12) è lineare, e si traduce nel seguente insieme di vincoli:

$$\vec{\varphi}_n(\Theta, \vec{\lambda_I}, \vec{\lambda_O}) = \begin{cases} 0 = R \oplus R^* \oplus R' \\ 0 = L \oplus L^* \oplus L' \\ \lambda_I^1 = R' \\ \lambda_I^2 \oplus \lambda_O^1 = L' \\ \vdots \\ \lambda_I^X \oplus \lambda_O^{X-1} \oplus \lambda_I^{X-2} = 0 \\ \vdots \\ \lambda_I^n = r' \\ \lambda_O^n \oplus \lambda_I^{n-1} = l' \end{cases}, \tag{6.2.13}$$

dove

$$\Theta = \{R, R^*, R', r', L, L^*, L', l'\}, \tag{6.2.14}$$

$$\vec{\lambda_I} = (\lambda_I^1, \lambda_I^2, \ldots, \lambda_I^n), \tag{6.2.15}$$

$$\vec{\lambda_O} = (\lambda_O^1, \lambda_O^2, \ldots, \lambda_O^n). \tag{6.2.16}$$

Quando si da una coppia di testi in chiaro in ingresso ad un sistema DES secondo le ipotesi fatte si ha che sicuramente il rispetto del vincolo

$$\vec{\varphi}_n(\Theta, \vec{\lambda_I}, \vec{\lambda_O}) = \vec{O}. \tag{6.2.17}$$

La definizione seguente identifica il vincolo $\vec{\varphi}$.

Definizione 6.1. *Il vincolo* (6.2.13) *è detto* ***vincolo differenziale lineare di ordine*** *n, oppure semplicemente* ***v.d.l.*** *di ordine n.*

L'interesse verso il v.d.l. risulterà chiaro nel seguito, infatti permetterà di ridurre la complessità computazionale del calcolo della likelihood. In base a quanto detto la rete che modella la likelihood sulla base delle differenze input/output è quella in figura 6.2. Le densità di questa rete si ricavano in modo diretto applicando il lemma 4.5 sul sistema del v.d.l., e della proposizione 6.1 per la densità sui nodi delle differenze in output della funzione di Feistel.

In questa situazione l'obiettivo è determinare le sottochiavi dei vari round. In ambito di chosen plaintext sono noti gli elementi dell'insieme Θ in (6.2.14), segue quindi che il problema della crittoanalisi differenziale generalizzato è ricondotto alla massimizzazione della likelihood

$$p_{S_{K1}, \ldots, S_{Kn}|\Theta}; \tag{6.2.18}$$

effettuando l'inferenza bayesiana si arriva alla conclusione che

$$\begin{cases} p_{S_{K1}, \ldots, S_{Kn}|\Theta} = \eta \sum_{\substack{\lambda_I^1, \ldots, \lambda_I^n \\ \lambda_O^1, \ldots, \lambda_O^n}} (p_1 p_2 p_3) \\ p_1 = \prod_{X=1}^n p_{\lambda_O^X|\lambda_I^X, S_{KX}} \\ p_2 = \left(p_{\lambda_I^1|R'} \times p_{\lambda_I^2|\lambda_O^1, L'}\right) \times \left(\prod_{X=3}^n p_{\lambda_I^X|\lambda_O^{X-1}, \lambda_I^{X-2}}\right) \times \left(p_{l'|\lambda_I^{n-1}} \times p_{r'|\lambda_O^n, \lambda_I^{n-1}}\right) \\ p_3 = (\prod_{X=1}^n p_{S_{KX}}) \times p_{R'|R, R^*} \times p_{L'|L, L^*} \times p_R \times p_{R^*} \times p_L \times p_{L^*} \end{cases}. \tag{6.2.19}$$

Nella (6.2.19) va osservato che nel generico addendo della prima equazione i fattori p_2 e p_3 fanno si che gli unici addendi non nulli nella somma sono quelli

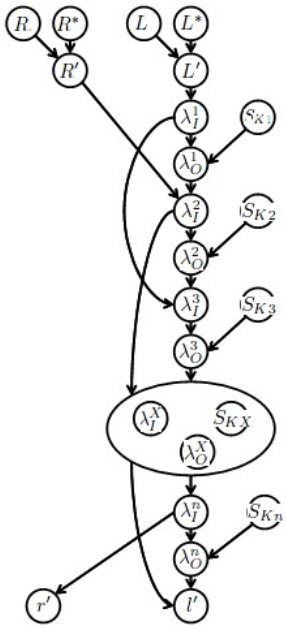


Figura 6.2: Rete per attaccare N-Round del DES.

che fanno si che il v.d.l. della definizione 6.1 sia verificato e quindi in modo più compatto si può scrivere

$$p_{S_{K1},\ldots,S_{Kn}|\Theta} = \eta \sum_{\vec{\varphi}_n(\Theta,\vec{\lambda}_I,\vec{\lambda}_O)=\vec{O}} \prod_{X=1}^{n} p_{\lambda_O^X|\lambda_I^X,S_{KX}}. \tag{6.2.20}$$

Uno dei problemi nell'implementazione della (6.2.20) è che bisogna trovare, per ricerca esaustiva tutti i λ_I^X, λ_O^X per i quali il v.d.l. è verificato, questo problema può essere parzialmente sorvolato se appunto si considera il fatto che $\vec{\varphi}_n$ è lineare, il fatto che sia lineare fa si che esso rappresentabile come un sistema

$$\Phi \cdot \begin{pmatrix} \vec{\lambda_I}^T \\ \vec{\lambda_O}^T \end{pmatrix} = \vec{b} \tag{6.2.21}$$

dove in questo caso i coefficenti del sistema sono espressi in termini di un qualche campo di Galois (è definito implicitamente dal numero di bit delle incognite, ma quale sia in particolare non ha molta importanza visto che la matrice Φ ha per elementi 0 e 1), e l'apice T indica la trasposizione di vettore; quindi utilizzando la teoria dei campi finiti il sistema si può risolvere. Come si vedrà nelle prossime sezioni, nelle quali verranno presentati alcuni esempi, lo studio della matrice Φ permetterà di ridurre la complessità computazionale del calcolo della likelihood. Va inoltre osservato che se il v.d.l. è rispettato allora per il teorema di Rouchè-Capelli [1] si ha che

$$\vec{b} \in Span(\Phi^1, \ldots, \Phi^n), \tag{6.2.22}$$

dove $\Phi^1, \ldots, \Phi^n$ sono le colonne della matrice Φ. Poichè lo scopo è determinare tutte le soluzioni del sistema (6.2.21) si supporrà sempre che (6.2.22) sia verificata.

6.2.1 Rete per un singolo round

Se si vuole modellare un singolo Round del DES si può adoperare la rete in figura 6.3 usando tale rete si vede che il v.d.l. è della forma

$$\varphi_1 = \begin{cases} 0 = R' \oplus R \oplus R^* \\ 0 = L' \oplus L \oplus L^* \\ \lambda_I^1 = R' \\ \lambda_O^1 = L' \oplus l' \\ 0 = r' \oplus R' \end{cases}, \tag{6.2.23}$$

dove i termini a destra di ogni equazione del sistema sono i termini noti (dovuti all'ipotesi di chosen plaintext attack); in modo equivalente può essere scritto come

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_I^1 \\ \lambda_O^1 \end{pmatrix} = \begin{pmatrix} R' \oplus R^* \oplus R \\ L' \oplus L^* \oplus L \\ R' \\ L' \oplus l' \\ r' \oplus R' \end{pmatrix}. \tag{6.2.24}$$

La matrice che caratterizza il sistema

$$\Phi = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{6.2.25}$$

permette di inferire che il rango della matrice Φ è 2, e per la (6.2.22) si ha che il rango della matrice completa del sistema (6.2.24) è anch'essa 2, e visto che le incognite del sistema sono 2 allora la soluzione è unica. Detto quindi S_{φ_1} l'insieme delle soluzioni della (6.2.25), che si ribadisce essere unica in questo caso, si ha che

$$p_{S_{K1}|\Theta} = \eta \sum_{\lambda_I^1, \lambda_O^1 \in S_{\varphi_1}} p_{\lambda_O^1|\lambda_I^1, S_{K1}}. \tag{6.2.26}$$

6.2.2 Rete per due round iterati

Per la modellazione di due round iterati viene proposta la rete mostrata in figura 6.4, in questo caso il sistema che descrive il v.d.l. è

$$\varphi_2 = \begin{cases} 0 = R' \oplus R \oplus R^* \\ 0 = L' \oplus L \oplus L^* \\ \lambda_I^1 = R' \\ \lambda_O^1 \oplus \lambda_I^2 = L' \\ \lambda_I^1 \oplus \lambda_O^2 = l' \\ \lambda_I^2 = r' \end{cases} \Rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_I^1 \\ \lambda_O^1 \\ \lambda_I^2 \\ \lambda_O^2 \end{pmatrix} = \begin{pmatrix} R \oplus R^* \oplus R' \\ L \oplus L^* \oplus L' \\ R' \\ L' \\ l' \\ r' \end{pmatrix}, \tag{6.2.27}$$

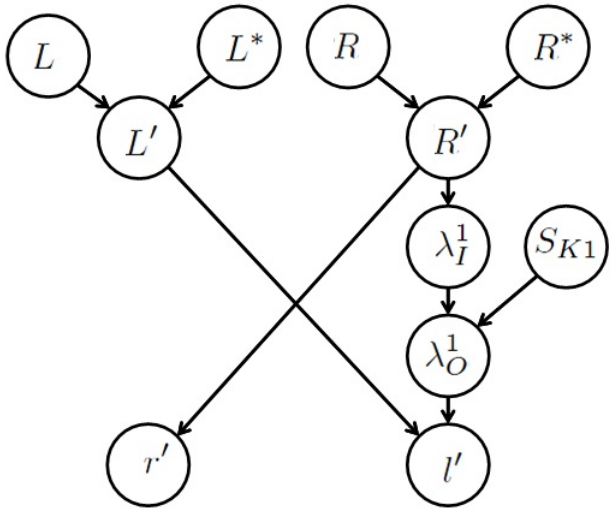


Figura 6.3: Rete per attaccare un singolo Round.

dove la matrice del sistema Φ ha questa volta rango 4 e quindi, considerando sempre la validità della (6.2.22) e il numero di incognite, la soluzione del sistema è unica. Quindi detto S_{φ_2} l'insieme delle soluzioni, unica anche in questo caso, del vincolo φ_2 e ponendo $\Lambda_2 = (\lambda_I^1, \lambda_O^1, \lambda_I^2, \lambda_O^2)$ si ha che

$$p_{S_{K1},S_{K2}|\Theta} = \eta \sum_{\Lambda_2 \in S_{\varphi_2}} \left[p_{\lambda_O^1|\lambda_I^1,S_{K1}} \times p_{\lambda_O^2|\lambda_I^2,S_{K2}} \right]. \tag{6.2.28}$$

6.2.3 Rete per tre round

Generalizzando ulteriormente la rete per tre round è in figura 6.5 i vincoli lineare sono espressi dal sistema lineare

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda_I^1 \\ \lambda_O^1 \\ \lambda_I^2 \\ \lambda_O^2 \\ \lambda_I^3 \\ \lambda_O^3 \end{pmatrix} = \begin{pmatrix} R \oplus R^* \oplus R' \\ L \oplus L^* \oplus L' \\ R' \\ L' \\ 0 \\ l' \\ r' \end{pmatrix} \tag{6.2.29}$$

la cui matrice ha rango 5, con l'ipotesi della (6.2.22) si ha che il rango del sistema è 5. Essendo le incognite 6, e il rango del sistema 5 si ha l'esistenza un solo parametro libero. Detto S_{φ_3} l'insieme delle soluzioni del sistema (6.2.29) e ponendo

$$\Lambda_3 = (\lambda_I^1, \lambda_O^1, \lambda_I^2, \lambda_O^2, \lambda_I^3, \lambda_O^3), \tag{6.2.30}$$

si ha che la likelihood ha l'espressione seguente

$$p_{S_{K1},S_{K2},S_{K3}|\Theta} = \eta \sum_{\Lambda_3 \in S_{\varphi_3}} \left[p_{\lambda_O^1|\lambda_I^1,S_{K1}} \times p_{\lambda_O^2|\lambda_I^2,S_{K2}} \times p_{\lambda_O^3|\lambda_I^3,S_{K3}} \right]. \tag{6.2.31}$$

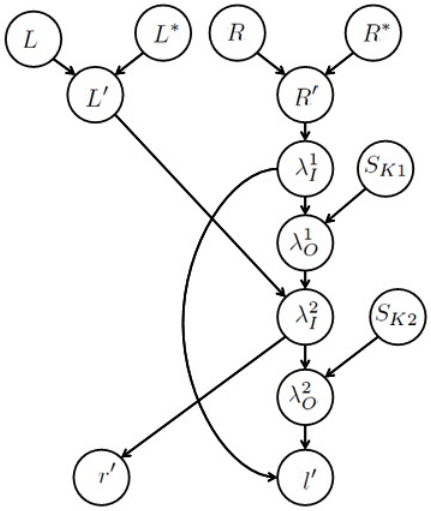


Figura 6.4: Rete crittoanalisi due Round.

6.3 Inferenza approssimata

Da un punto di vista computazionale effettuare un'inferenza diagnostica dopo aver effettuato un'attacco risulta molto onerosa. Va infatti osservato che anche se si usano i vincoli differenziali per poter ridurre la complessità di calcolo degli indici della sommatoria (6.2.20) comunque il calcolo rimane oneroso, se i parametri liberi del v.d.l. sono n allora lo scorrimento di tutti gli indici della (6.2.20) ha una complessità di 2^{32n}, cioè è esponenziale. Altro problema, non meno grave, è il calcolo della statistica $p_{\lambda_O^X|\lambda_I^X,S_{KX}}$, che dalla proposizione 6.1 segue la necessità di effettuare 2^{32} operazioni di somma. Inoltre vi è la massimizzazione della likelihood, che in ogni caso rimane computazionalmente oneroso da effettuare, visto che le sottochiavi sono indipendenti lo spazio delle chiavi è molto grande quindi anche algoritmi di ricerca approssimata dei minimi, come il Simulated Annealing [11] risulta essere oneroso. Risulta invece più efficace ed efficiente utilizzare, e più semplice da implementare, una tecnica di inferenza in avanti [13]. Invece di utilizzare una rete per inferire la chiave mediante un processo di inferenza diagnostico è possibile utilizzare una rete per stimare una possibile propagazione di differenze nei vari round. Questa propagazione statistica rende possibile stimare una possibile differenza in output alla funzione di Feistel dell'ultimo round, e quindi si avrà la possibilità di inferire un insieme di chiavi. Stimare più propagazioni permetterà quindi di estrapolare un istogramma delle varie sottochiavi all'ultimo round. L'istogramma convergerà alla chiave corretta, e la chiave che ha probabilità massima sarà assunta come corretta.

6.3.1 Rete per la statistica delle differenze

Piuttosto che esporre un'algoritmo generale si ritiene più semplice esporre l'approccio tramite un esempio, si supponga di dover attaccare un sistema DES ridotto a tre Round, per quanto detto in precedenza si deve iniziare con lo stu-

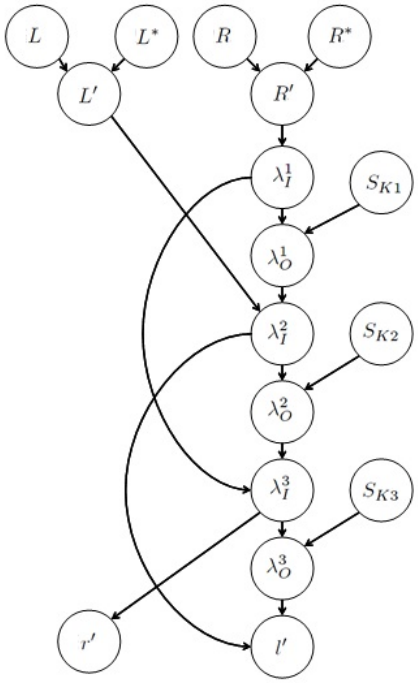


Figura 6.5: Rete per 3 round.

dio della propagazione delle differenze, e questo può avvenire mediante la rete in figura 6.6.

La stima delle differenze può avvenire campionando, in ordine, i vari nodi della rete; L'ordine di campionamento è dato da un qualunque ordinamento topologico del grafo in figura 6.6. Un esempio di campionamento è quello dettato dalla lista seguente:

1. Campionare il nodo R' secondo una distribuzione uniforme;
2. Campionare il nodo L' secondo una distribuzione uniforme;
3. Campionare il nodo $\lambda_I^1 = R'$. Vista l'uguaglianza, l'unico campione di λ_I^1 da campionare sarà campionato precedentemente da R';
4. Campionare il nodo λ_O^1, dato λ_I^1 (questo passo importante sarà spiegato successivamente).
5. Campionare il nodo $\lambda_I^2 = L' \oplus \lambda_O^1$, visti i campionamenti precedenti il campione è univocamente determinato;
6. Campionare il nodo λ_O^2, si veda il passo 4;
7. Campionare il nodo $\lambda_I^3 = \lambda_I^1 \oplus \lambda_O^2$, che è univoco;
8. Campionare il nodo λ_O^3, si veda il passo 4 e 6;
9. Campionare il nodo $r' = \lambda_I^3$, che è univoco;

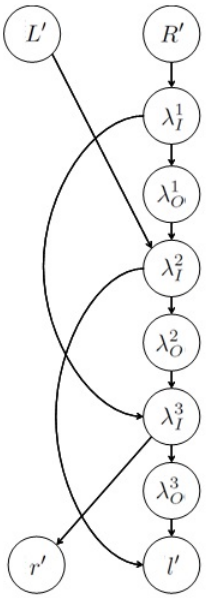


Figura 6.6: Rete per la stima delle propagazione delle differenze.

10. Campionare il nodo $l' = \lambda_O^3 \oplus \lambda_I^2$, che è univoco.

Ipotizzando di voler attaccare il a tre round, con un tipo di attacco chosen plaintext, non ha senso campionare la rete, secondo i passi esposti in precedenza. Si ricorda che l'ipotesi di chosen plaintext prevede di poter scegliere l'input, quindi una coppia di testi in chiaro che genera una data differenza in input, e di poter osservare i rispettivi output, quindi è noto l'output delle differenze. Per costruzione del DES l'ingresso λ_I^3 è noto, e poichè l' è noto se si riesce a campionare qualche valore di λ_I^2 allora utilizzando le S-Box B-NET introdotte nel capitolo 5 e quindi costruire un istogramma delle possibili sottochiavi.

6.3.2 Funzioni di campionamento

L'algoritmo 6.1 determina un possibile valore di λ_I^2, da utilizzare poi per determinare l'output della funzione di Feistel dell'ultimo Round del DES a tre round.

Algoritmo 6.1 Campionamento dalla rete di propagazione delle differenze

```
1: function GET_PROPAGATION(R',L')
2:     λ_I^1 ← R'
3:     λ_O^1 ← SAMPLE(λ_I^1)
4:     λ_I^2 ← L' ⊕ λ_O^1
5:     return λ_I^2
6: end function
```

La chiamata alla funzione *Sample* permette di campionare secondo la distri-

buzione $p_{\lambda_O^X|\lambda_I^X}$; Per definizione risulta

$$\lambda_O^X = P(S(E(Z_X) \oplus S_{KX}) \oplus S(E(Z_X) \oplus E(\lambda_I^X) \oplus S_{KX}), \tag{6.3.1}$$

dove $Z_X \in \mathbb{Z}_2^{32}$ e $S_{KX} \in \mathbb{Z}_2^{48}$; ponendo

$$Z_{IX} = E(Z_X) \oplus S_{KX}, \tag{6.3.2}$$

e supponendo di fissare Z_{IX} arbitrario in $\mathbb{Z}_2^{48}$, esisteranno quindi diverse coppie (Z_X, S_{KX}) tale che la (6.3.2) sia valida. Per poter trovare tutte le coppie si supponga di fissare Z_X, allora esisterà un unico S_{KX} che renderà valida la (6.3.2); quindi per ogni Z_{IX} esistono 2^{32} coppie (Z_X, S_{KX}) che renderanno vera l'equazione. Applicando quanto detto alla proposizione 6.1 si ha che

$$p_{\lambda_O^X|\lambda_I^X} = \sum_{S_{KX}} p_{\lambda_O^X|\lambda_I^X, S_{KX}} p_{S_{KX}} = \frac{1}{2^{48}} \sum_{S_{KX}} p_{\lambda_O^X|\lambda_I^X, S_{KX}}; \tag{6.3.3}$$

per la proposizione 6.1. si ha che

$$p_{\lambda_O^X|\lambda_I^X, S_{KX}} = \frac{1}{2^{32}} \sum_{Z_X} \delta(\lambda_O^X \oplus P(S(E(Z_X) \oplus S_{KX}) \oplus S(E(Z_X) \oplus E(\lambda_I^X) \oplus S_{KX}))); \tag{6.3.4}$$

sostituendo la (6.3.9) nella (6.3.3) si ha

$$\begin{aligned} &p_{\lambda_O^X|\lambda_I^X} = \\ &\frac{1}{2^{32}} \frac{1}{2^{48}} \sum_{S_{KX}} \sum_{Z_X} \delta(\lambda_O^X \oplus P(S(E(Z_X) \oplus S_{KX}) \oplus S(E(Z_X) \oplus E(\lambda_I^X) \oplus S_{KX}))) = \\ &\quad \frac{1}{2^{80}} \sum_{S_{KX}, Z_X} \delta(\lambda_O^X \oplus P(S(E(Z_X) \oplus S_{KX}) \oplus S(E(Z_X) \oplus E(\lambda_I^X) \oplus S_{KX}))). \end{aligned} \tag{6.3.5}$$

Ricordando ora che la (6.3.2) ha per soluzioni 2^{32} coppie (Z_X, S_{KX}) vale l'uguaglianza

$$\begin{aligned} &\sum_{S_{KX}, Z_X} \delta(\lambda_O^X \oplus P(S(E(Z_X) \oplus S_{KX}) \oplus S(E(Z_X) \oplus E(\lambda_I^X) \oplus S_{KX}))) = \\ &\qquad 2^{32} \sum_{Z_{IX}} \delta(\lambda_O^X \oplus P(S(Z_{IX}) \oplus S(Z_{IX} \oplus E(\lambda_I^X)))); \end{aligned} \tag{6.3.6}$$

sostituendo la (6.3.6) nella (6.3.5) si giunge a

$$p_{\lambda_O^X|\lambda_I^X} = \frac{1}{2^{48}} \sum_{Z_{IX}} \delta(\lambda_O^X \oplus P(S(Z_{IX}) \oplus S(Z_{IX} \oplus E(\lambda_I^X)))). \tag{6.3.7}$$

Per poter campionare la (6.3.1) si può utilizzare la relazione

$$\lambda_O^X = P(S(Z_{IX}) \oplus S(Z_{IX} \oplus E(\lambda_I^X))), \tag{6.3.8}$$

dove λ_I^X è fissato. Se Z_{IX} è uniformemente distribuito in $\mathbb{Z}_2^{48}$, si ha che

$$\begin{aligned} Pr\left\{\lambda_O^X = o \mid \lambda_I^X = i\right\} = Pr\left\{Z_{IX} \in I^{-1}(o) \mid \lambda_I^X = i\right\} = \\ \frac{1}{2^{48}} \sum_{Z_{IX}} \delta(\lambda_O^X \oplus P(S(Z_{IX}) \oplus S(Z_{IX} \oplus E(\lambda_I^X)))), \end{aligned} \tag{6.3.9}$$

dove con $I^{-1}(o)$ si indica la controimmagine di o secondo la (6.3.8). Quindi se si prende un valore di Z_{IX} da una distribuzione uniforme, e fissando λ_I^X, dalla (6.3.1) si otterrà un campione di λ_O^Xdistribuito secondo la distribuzione $p_{\lambda_O^X|\lambda_I^X}$. L'algoritmo 6.2 riassume la discussione appena tenuta.

Algoritmo 6.2 Implementazione della funzione Sample

```
1: function SAMPLE(λ_I^X)
2:     Z_X ← RANDOM(0, 2^48 − 1)
3:     λ_O^X ← P(S(Z_IX) ⊕ S(Z_IX ⊕ E(λ_I^X)))
4:     return λ_O^X
5: end function
```

Osservazioni Va precisato che esistono diverse tecniche di campionamento da una distribuzione di probabilità, da utilizzare proprio perchè la distribuzione da cui si campiona non è trattabile, come nel caso esposto in questo capitolo. In questo capitolo si è voluto esporre un metodo, semplice e particolarmente adatto al problema, ma non è da escludere che l'implementazione di altri metodi di campionamento possa portare a risultati migliori.

Capitolo 7

Progettazione del Sistema

Gli studi teorici portati avanti in questa tesi permettono di costruire le basi per implementare i tools da utilizzare nell'ambito della crittoanalisi differenziale. In questo capitolo verranno implementati diversi tools: il primo tools aerve di attaccare le singole S-Box, il secondo tools consente di attaccare fino ad un massimo di cinque round iterati del DES. Questi tools sono parametrizzati, ovvero la loro implementazione dipende da alcuni parametri (come ad esempio il numero di campioni da utilizzare o il tipo di struttura dati per rappresentare la likelihood). Lo scopo di questo capitolo è spiegare il funzionamento dei tools, descrivendo i file di intestazione.

7.1 S_Box_Likelihood

Dalla teoria trattata nel capitolo 5, e dai vari pseudocodici esposti, si può affermare che l'implementazione della Likelihood è generalmente caratterizzata da

- Il numero totali di attacchi effettuati (che coincide con il numero di coppie testi in chiaro testi cifrati da utilizzare).
- La S-Box da attaccare.
- Una struttura dati che rappresenti adeguatamente la likelihood.
- Un metodo di aggiornamento della struttura dati che rappresenta la likelihood.

Il numero di attacchi caratterizza lo stato dell'attacco (e quindi la likelihood risultante), la S-Box è ovviamente quella che si vuole attaccare. Struttura dati e metodo di aggiornamento vanno di pari passo, la struttura dati influenza sia la rappresentazione della likelihood, sia il modo in cui si inferiscono i vari dati associati, ad esempio se si è ad uno stato n quale struttura dati permette un calcolo efficiente della probabilità che una data chiave sia corretta? o il calcolo del fattore di normalizzazione? quale struttura dati fa si che la likelihood sia aggiornata in modo veloce? quale metodo di aggiornamento permette di velocizzare il cambio di stato della likelihood?. Il listato 7.1 mostra lo scheletro della classe implementata.

```
1  class S_Box_Likelihood {
2  public:
3    S_Box_Likelihood(
4      int structure_type,
5      int calculation,
6      int(*s_box)(int)
7      );
8    void update(
9      int32 e_x,
10     int32 e_x_star,
11     int32 i_x_prime,
12     int32 o_x_prime
13     );
14   void update_With_Multiple_Data(
15     list<int32> e_x_list,
16     list<int32> e_x_star_list,
17     list<int32> i_x_prime_list,
18     list<int32> o_x_prime_list
19     );
20   int probability(int32 k_x);
21   int size();
22   void print_key();
23 private:
24   void* likelihood_struct;
25   int data_structure_type;
26   int likelihood_calculation;
27   int n_attack;
28   int (*s)(int);
29 };
```

Listing 7.1: Scheletro della Classe S_Box_Likelihood.

Come si vede la classe è caratterizzata dai seguenti membri privati

- `void* likelihood_struct`: questo puntatore void è poi specializzato per poter essere un array di 2^6 celle di interi, oppure una lista, inizializzata a tutti i possibili valori nel range $[0_x, 3F_x]$.
- `int data_structure_type`: questo è un indicatore del tipo di struttura dati utilizzata per rappresentare la likelihood.
- `int likelihood_calculation`: indicatore del tipo di algoritmo utilizzato per aggiornare la likelihood, questo può basarsi sul metodo generale del calcolo della likelihood mediante la retebayesiana S-Box B-Net oppure usare il modello matematico di soluzione della rete.
- `int n_attack`: un intero che indica lo stato della likelihood.
- `int (*s)(int)`: puntatore a funzione che caratterizza la likelihood, cioè l'S-Box alla quale è associata la likelihood.

La medesima classe è caratterizzata dai seguenti metodi pubblici

- `S_Box_Likelihood(...)` : Il costrutture della classe che prende tre argomenti, il primo argomento è una costante che può essere `ARRAY` oppure `LIST`, questo parametro fa si che il membro privato `likelihood_struct` sia inizializzato, tramite un casting opportuno, ad una lista oppure un array, in seguito il parametro `data_structure_type` viene inizializzato alla costante appena passata. Il secondo parametro specifica il metodo di aggiornamento che può essere `DIRECT_BNET_ALGORITHM` oppure

`LIKELIHOOD_SOLUTION` e influenzano pesantemente il modo in cui il metodo `update` opera l'aggiornamento della likelihood, che nella pratica ancora si traduce nella modalità di aggiornamento della struttura dati `likelihood_struct`. La modalità `DIRECT_BNET_ALGORITHM` effettua l'inferenza diagnostica avvalendosi della struttura della rete, mentre la `LIKELIHOOD_SOLUTION` utilizza direttamente la soluzione della likelihood. L'ultimo parametro serve per specificare la S-Box.

- `void update(...)`: Questo metodo prende in input i parametri `e_x`, `e_x_star` che rappresentano due testi in chiaro, i nodi Si_{EX}, Si^*_{EX} della rete. Il parametro `i_x_prime` è l'input xor tra due testi in chiaro, e nella rete S-Box B-Net rappresenta il nodo Si'_{IX}, il parametro `o_x_prime` rappresenta il nodo Si'_{OX} all'interno della rete. Questo metodo fa si che si passi dallo stato n allo stato $n+1$; esso implementa la transazione

  $$L_n \to L_{n+1};$$

  la trasformazione avviene aggiornando opportunamente la struttura dati `likelihood_struct`.

- `void update_With_Multiple_Data(...)`: ipotizzando di avere delle liste di medesima dimensione m e che la likelihood si trovi allo stadio n il metodo effettua l'aggiornamento

  $$L_n \to L_{n+m}.$$

- `void probability(...)`: il metodo prende per parametro una possibile chiave, accertandosi che essa sia nell'intervallo corretto, essa calcola probabilità non normalizzata, dipende dal particolare stadio della likelihood, e dagli attacchi che sono stati effettuati.

- `void size(...)`: ritorna la dimensione dell'insieme delle chiavi per i quali la likelihood è diversa da 0.

- `print_key(...)`: stampa l'insieme delle chiavi stimate tramite la likelihood.

Il metodo `update(...)` è sicuramente il più importante della classe sul quale vale la pena soffermarsi, è stato detto in precedenza che la classe ha quattro modalità di funzionamento che caratterizza i calcoli effettuati e la struttura dati che la rappresenta. Nel seguito vengono elencati i funzionamenti possibili

Struttura dati ARRAY, calcolo BNET_DIRECT_ALGORITHM. Un array in questo contesto è una struttura dati della forma

$$\vec{p} = (p_0, p_1, \ldots, p_{3F_x})$$

ogni elemento p_{k_x} è la probabilità, non normalizzata, di correttezza di k_x. L'aggiornamento in questo caso procede nel modo seguente, ipotizzando che si parta da una condizione iniziale della forma

$$\vec{p} = (1, 1, \ldots, 1)$$

e si aggiorna ogni valore del vettore secondo il modello

$$p_{k_x} \leftarrow p_{k_x} * p_{Si_{KX}|Si_{EX},Si^*_{EX},Si'_{IX},Si'_{OX}}(k_x, e_x, e^*_x, i'_x, o'_x)$$

avvalendosi dell'algoritmo di inferenza bayesiana sulla S-Box B-NET. In breve si implementa la relazione di ricorrenza (5.3.21) e l'aggiornamento avviene mediante l'implementazione dello pseudocodice dell'algoritmo 5.1.

Struttura dati ARRAY, calcolo LIKELIHOOD_SOLUTION. Si utilizza la medesima struttura dati, con la stessa semantica, quello che cambia è il metodo di calcolo. In questo caso il metodo di calcolo sfrutta la formula della soluzione della likelihood (implementata mediante l'algoritmo 5.3, quindi se l'`update` ha in input e_x, e^*_x, i'_x, o'_x per ogni chiave k_x si controlla la validità del sistema

$$\begin{cases} i_x = e_x \oplus k_x \\ i^*_x = e^*_x \oplus k_x \\ i'_x = i_x \oplus i^*_x \end{cases},$$

se questo ammette soluzione si controlla la validità dell'equazione

$$o'_x = Si(i_x) \oplus Si(i^*_x)$$

se questa è verificata allora con l'attacco corrente la probabilità che questa chiave sia quella corretta è un valore $p \neq 0$ e quindi si aggiorna la componente k_x dell'array $\vec{p}$ come

$$p_{k_x} \leftarrow p_{k_x} * p.$$

Struttura dati LIST, calcolo DIRECT_BNET_ALGORITHM. In questo caso una lista è un insieme di chiavi della forma

$$K = \{k_1, k_2, ..., k_n\}$$

dove ogni elemento della lista è una possibile chiave ritenuta corretta. Il calcolo della likelihood vero e proprio è effettuato con inferenza bayesiana sulla S-Box B-NET ma solo sulle chiavi dell'insieme K se qualche chiave in questo insieme ha probabilità nulla allora viene rimossa dall'insieme K, questo farà si che si otterrà un insieme $K' \subset K$ che conterrà le nuove possibili chiavi stimate come corrette.

Struttura dati LIST, calcolo LIKELIHOOD_SOLUTION Analogamente a quanto detto prima, si ha l'insieme K, con i nuovi dati in ingresso si vede se ogni elemento di K è tale che il sistema

$$\begin{cases} i_x = e_x \oplus k_x \\ i^*_x = e^*_x \oplus k_x \\ i'_x = i_x \oplus i^*_x \end{cases}$$

sia verificato, quando questo non è verificato la chiave k viene rimossa dalla lista.

7.2 DES_Attack

La classe `DES_Attack` essenzialmente sfrutta la tecnica di campionamento per poter effettuare la crittoanalisi su più round iterati del DES. Analogamente alle S-Box la classe ha dei parametri che ne specificano il metodo di aggiornamento, la struttura dati, il numero di campioni per ipotizzare le possibili propagazioni e il numero di round. Questi parametri influenzano la velocità di convergenza e le prestazioni. Il listato 7.2 mostra i dati membro e i metodi principali che verranno descritti di seguito.

```
1  class DES_Attack {
2  public:
3    DES_Attack(
4      int nRound,
5      int nSample,
6      int nHist,
7      int48 k[],
8      int structureType
9    );
10   void attack();
11   int48 key_max_prob();
12 private:
13   int n_round;
14   int n_sample;
15   int n_hist;
16   int48* key;
17   int structure_type;
18   void* structure;
19   void* update_function;
20 };
```

Listing 7.2: Scheletro della Classe DES_Attack.

I membri privati della classe, e le relative descrizioni, sono elencati di seguito:

- `int n_round`: questo dato membro indica il numero di round iterati che si possono violare, l'implementazione proposta supporta fino ad un massimo di cinque round iterati;
- `int n_sample`: è il numero di ipotesi di differenze propagate sfruttate in un singolo attacco; Cioè data una coppia testo in chiaro/testo cifrato verranno ipotizzate diverse possibili propagazioni di differenze, secondo la likelihood dettata dalla rete delle differenze;
- `int n_hist`: questo parametro indica il numero di istogrammi da utilizzare per poter inferire la chiave, questi istogrammi vanno da un minimo di 1 ad un massimo di 8. L'implementazione di un diverso numero di istogrammi è stato dettato da alcuni suggerimenti dati in [2], la loro utilità sarà chiarita nel prossimo capitolo, dedicato ai risultati sperimentali;
- `int48* key`: è l'array delle sottochiavi indipendenti del sistema da violare;
- `int structure_type`: è un intero che indica il tipo di struttura dati da utilizzare per implementare gli istogrammi, essi sono stati essenzialmente implementati in due modi, il primo è classici array, il secondo è mediante liste;

- `void* structure`: un puntatore void che si specializzerà con la particolare struttura dati da utilizzare per gli istogrammi;
- `void* update_function`: è un puntatore void che poi sarà specializzato in un particolare metodo di aggiornamento degli istogrammi, essenzialmente i metodi sono due, il primo è uno schema di conteggio che ricalca la S-Box B-NET con uso di `ARRAY`, con metodo `LIKELIHOOD_SOLUTION`. Il secondo invece inserisce proprio il modulo S-Box B-NET così come implementato nella sezione precedente.

I membri pubblici della classe sono invece i seguenti:

- `DES_Attack(...)`: è il costrutture della classe che permette di settare i parametri privati descritti in precedenza. Il parametro `structureType` può essere `ARRAY_HIST`, `MAP_HIST` oppure `ARRAY_HIST_NO_S_BOX_TOOL`.
- `void attack(...)`: è il cuore della classe, implementa l'ordinamento topologico della rete bayesiana, effettua il campionamento e inferisce possibili valori di sottochiavi.
- `int48 key_max_prob(...)`: questo metodo calcola la chiave che ha probabilità massima sotto diverse ipotesi di indipendenza, il numero di gruppi indipendenti è legato al parametro `n_hist`. Questo parametro influisce pesantemente sulle prestazioni dell'algoritmo.

Le modalità di funzionamento sono elencate di seguito:

Modalità ARRAY_HIST Nella modalità array gli istogrammi sono semplicemente degli array. Nel seguito viene indicata la relazione tra il numero di istogrammi e le dimensioni degli array:

- 8 istogrammi $\rightarrow$ dimensione 2^6;
- 4 istogrammi $\rightarrow$ dimensione 2^{12};
- 2 istogrammi $\rightarrow$ dimensione 2^{24};
- 1 istogrammi $\rightarrow$ dimensione 2^{48}.

Data una possibile propagazione i bit delle sottochiavi sono stimati mediante l'uso delle `S_Box_Likelihood`, con struttura dati `LIST` e metodo `LIKELIHOOD_SOLUTION`. Questa prima implementazione mostra la possibilità del riuso del tools sviluppato in precedenza per inferire le chiavi. Ipotizzando che gli istogrammi siano associati a sottochiavi indipendenti tra di loro la ricerca delle chiave che ha probabilità massima è quella che i cui diversi gruppi di bit indipendenti massimizzano i singoli istogrammi.

Modalità MAP_HIST In questa modalità gli istogrammi non sono degli array, ma delle mappe (struttura dati `map` presente nello standard del C++). Ogni mappa ha per chiave un intero a 48 bit, cioè è indicizzata con una chiave, questa struttura dati è dinamica, non esiste una relazione diretta tra numero di istogrammi e dimensione, quindi in memoria si potrebbe avere un risparmio di risorse. La ricerca dei massimi è analoga al caso precedente.

Modalità ARRAY_HIST_NO_S_BOX_TOOL Questa modalità reimplementa la modalità `LIKELIHOOD_SOLUTION` all'interno della classe `DES_Attack`, in particolare all'interno del metodo `update(...)` di tale classe. Questa reimplementazione diretta è più veloce in quanto sono trascurati diversi controlli (ad esempio le eccezioni) che venivano effettuati all'interno della classe `S_BOX_LIKELIHOOD`.

7.3 Cenni su altri tools sviluppati

I tools che sono stati esposti sono quelli che in fase di sperimentazione hanno dato risultati significativi rispetto agli studi fatti. Si può osservare però che i metodi di uso di entrambi i tools sono per certi aspetti differenti, per altri molto simili; Ad esempio entrambi si avvalgono delle reti bayesiane per la loro implementazione, ma vengono utilizzati in modo diverso; Mentre le S-Box utilizzano la rete mediante un *inferenza diagnostica* la rete per i round multipli del DES viene utilizzata per una *inferenza causale*. A cavallo tra questi due tools ne è stato sviluppato un terzo che implementa direttamente la rete generale esposta nel capitolo precedente; Questo terzo tools implementava l'inferenza diagnostica su tutto il DES in modo esatto, e secondo gli studi fatti, molti dei quali sono stati esposti nel precedente capitolo, si sarebbe dovuto rivelare il migliore da un punto di vista dei risultati. Tuttavia il problema principale sono stati i domini sui quali effettuare l'inferenza, molto grandi per le capacità degli elaboratori utilizzati. Il tools in questione effettuava l'inferenza diagnostica sulla rete, per ogni attacco effettuato e poi ne veniva calcolata la chiave che massimizzava la probabilità, una generalizzazione di quanto fatto le S-Box sostanzialmente. Le differenze tra attacco alle S-Box e all'intero DES mediante l'inferenza diagnostica riguardano i domini ove viene effettuata l'inferenza, mentre bisogna inferire una sottochiave di 6 bit per una singola S-Box, per un singolo round del DES bisogna inferire 48 bit e quindi un totale di 768 se si parla dell'intero DES (sotto le ipotesi di chiavi indipendenti di [2], ipotesi fatte per semplificare il trattamento teorico). Il calcolo della probabilità di una singola chiave inoltre è molto più oneroso che nella singola S-Box, quindi oltre ai domini più grandi bisogna anche tener conto dei singoli calcoli che non sono paragonabili. Per tentare di velocizzare il calcolo è stato fatto ricorso alla multiprogrammazione, ottenendo un aumento delle prestazioni. Calcolata la statistica di ogni chiave bisogna poi trovare quella chiave che massimizza la likelihood, e viste le dimensioni dei domini si è fatto ricorso al simulated annealing. Tuttavia nonostante questi accorgimenti i risultati non sono stati significativi, a causa di tempi lunghi di elaborazione, e si è quindi deciso di abbandonare l'approccio, rimandando però ad un eventuale sviluppo futuro una sua eventuale implementazione. Dopo ulteriori approfondimenti in merito all'uso di schemi di inferenza su reti la cui inferenza diagnostica non è trattabile (si veda [13]) si è optato per uno schema di inferenza in avanti mediante l'uso di alcune ipotesi tratte dalla statistica implicita della rete del DES.

Capitolo 8

Risultati Sperimentali

A fronte della teoria, e della spiegazione dei tools sviluppati sono stati effettuati diversi esperimenti finalizzati a testare l'efficienza e l'efficacia dei vari algoritmi proposti. In questo capitolo vengono riportati i risultati sperimentali relativi ai test effettuati sia in termini di coppie testo in chiaro testo cifrato da utilizzare per perpretare gli attacchi, sia in termini di tempo, visto che i tools possono essere implementati con diverse strutture dati. Si analizzerà prima il comportamento delle `S_Box_Likelihood` e poi del `DES_Attack`, entrambi introdotti nel capitolo precedente.

8.1 Caratteristiche dell'elaboratore utilizzato

Per effettuare i test è stato utilizzato un elaboratore dalle dalle caratteristiche seguenti:

Processore: Intel Core Duo T6600, frequenza 2.20 GHz, 2.20 GHz;

Ram: 4 GB;

Sistema Operativo: Windows 7 a 32 bit.

Il sistema Bayesiano sviluppato non è stato implementato mediante la multiprogrammazione, in quanto non è stato ritenuto necessario utilizzarla.

8.2 Valutazione sperimentale degli attacchi alle S-Box

Lo scopo dei test, in questo caso, è quello di valutare la quantità di dati necessari ad attaccare una S-Box e di misurare i tempi di esecuzione delle diverse implementazioni mostrate nel capitolo precedente. Le seguenti tabelle in figura 8.1-8.4 riportano i risultati di diversi test effettuati su ognuna delle otto S-Box della funzione di Feistel, la prima colonna indica la chiave da cercare, la seconda indica il numero di attacchi che si sono dovuti effettuare e la terza il tempo per portare a termine l'intero attacco. Il numero di attacchi coincide con il numero di coppie testo in chiaro testo cifrato necessarie a violare le S-Box. Alla fine

di ogni tabella viene riportato il valore medio, la deviazione standard e la mediana associata al numero di coppie testo in chiaro testo cifrato utilizzate per violare le S-Box e il tempo speso per attaccare. Le tabelle sono presentate a gruppi di otto; il primo gruppo usa `ARRAY` e `DIRECT_BNET_ALGORITHM`, il secondo gruppo `LIST` e `DIRECT_BNET_ALGORITHM`, il terzo gruppo `ARRAY` e `LIKELIHOOD_SOLUTION` e l'ultimo `LIST` e `LIKELIHOOD_SOLUTION`. Come si vede il numero di attacchi è quasi sempre fisso a 3, mentre i tempi sono sempre dell'ordine dei decimi di secondo per la modalità `DIRECT_BNET_ALGORITHM`, mentre è praticamente nullo nel caso della modalità `LIKELIHOOD_SOLUTION`. Un'aspetto interessante è che la struttura a lista rende l'algoritmo meno performante della versione con array. Quindi si può concludere che l'implementazione con `ARRAY` e in modalità `LIKELIHOOD_SOLUTION` della likelihood, piuttosto che l'inferenza effettiva sulla rete, porta a risultati soddisfacenti in termini di tempi di esecuzione. Avendo osservato che il numero di attacchi necessari a violare le S-Box è in media costante, e ovviamente indipendente dalla struttura dati utilizzata, si ritiene significativo riassumere i risultati tempistici con dei grafici a istogramma, riportati in figura 8.5-8.8; questi istogrammi mettono in relazione le S-Box con le medie, varianze e mediane dei tempi misurate per violarle. In questi istogrammi è maggiormente evidente che l'implementazione migliore del Tool è quella con struttura `ARRAY` e modalità `LIKELIHOOD_SOLUTION`.

8.3 Valutazione sperimentale degli attacchi al DES

I test sugli attacchi al DES sono ricaduti in tre tipologie distinte:

1. Test volti a verificare l'implementazione migliore;

2. Test volti a stabilire se l'assunzione di indipendenza/dipendenza tra i vari bit delle sottochiavi contribuisca a far convergere più velocemente l'algoritmo;

3. Test volti a studiare il numero di ipotesi di propagazione da dover utilizzare in un attacco, ovvero il numero di campioni della distribuzione $p_{\lambda_O^X|\lambda_I^X}$;

4. Test volti a studiare se fissare una determinata differenza in input contribuisca a far convergere più velocemente l'algoritmo.

La prima delle quattro tipologie di test è basata sulla comparazione dei tempi di esecuzione delle tre implementazioni della classe `DES_Attack`. Nel secondo test si assume che gruppi di bit della stessa sottochiave possano essere indipendenti o meno (ad esempio se $K = (K_1, K_2)$ si assume che K_1 e K_2 siano tra loro indipendenti); il numero di gruppi indipendenti si fissa prima di effettuare l'attacco mediante un parametro opportuno del costruttore della classe `DES_Attack`. Il terzo test è mirato a determinare un numero di campioni ottimali della distribuzione $p_{\lambda_O^X|\lambda_I^X}$ da fissare quando si effettua un attacco. Nel quarto test si è valutato il numero di coppie testo in chiaro/testo cifrato da utilizzare in un attacco quando la differenza tra gli input risulta fissata ad un certo valore.

8.3.1 Valutazione dei tempi di esecuzione

La valutazione dei tempi di esecuzione è stata effettuata tentando di violare un DES ridotto a tre round. L'obiettivo dell'attacco è l'individuazione della chiave `k3 = 0x12343456789a`. Gli ingressi sono stati assunti random (il che equivale a dire che non si è fissata una particolare differenza in input). La tabella 8.9 mostra i risultati ottenuti, i test sono stati effettuati con le varie implementazioni proposte, con un diverso numero di ipotesi di propagazione. Dai risultati ottenuti si evince che l'implementazione più efficiente è data dal non uso del modulo `S_Box_Likelihood` come modulo all'interno della classe `DES_Attack`, conviene quindi utilizzare dei semplici array come istogrammi, inferendo la chiave utilizzando la soluzione della likelihood. I tempi da cui si sono estratti media, varianza e mediana riportati nella tabella 8.9 sono associati alla comparsa della chiave corretta in corrispodenza del massimo dell'istogramma delle chiavi. Il grafico in figura 8.10 riassume questi andamenti tempistici, dai quali si nota che l'implementazione migliore è data dalla modalità `ARRAY_NO_S_BOX_TOOL`.

8.3.2 Valutazione dell'ipotesi di interdipendenza delle chiavi

In [2] gli autori indicano che l'assunzione di interdipedenza delle sottochiavi nei loro schemi di conteggio per costruire l'istogramma aumenta la velocità di convergenza del loro algoritmo, è da precisare che il loro approccio è molto differente da quello proposto in questa tesi, in quanto loro non fanno ipotesi multiple sulla propagazione delle differenze, ma ne viene fissata una che è ritenuta la più probabile e sulla base di questa vengono inferite le sottochiavi (a gruppi). L'idea proposta in [2] per attaccare le S-Box è quella di fissare a priori (cioè prima di effettuare un qualunque attacco) una propagazione di differenze, questa propagazione fissata è la più probabile o comunque una propagazione che sia in qualche modo utile e si osserva che nell'ultimo round del DES sono noti con una certa propabilità i bit entranti e uscenti in alcune, ma non tutte, le S-Box. Si attaccano quindi le S-Box i cui ingressi sono statisticamente noti e quindi si estrae un'istogramma relativi alle chiavi di 6 bit dai quali si inferisce la chiave più probabile. Il modo con cui vengono costruiti questi istogrammi vengono definiti in [2] come *counting scheme* (schemi di conteggio). Violate le prime S-Box si ipotizza una propagazione differente, ma che permetta di attaccare le S-Box non violate, mediante il medesimo schema di conteggio. Nell'approccio con reti Bayesiane invece si fanno diverse ipotesi sulla propagazione delle differenze, in base alla differenza ipotizzata si estraggono delle sottochiavi di 48 bit attaccando tutte le S-Box e si costruisce un istogramma, la chiave più probabile viene ritenuta quindi la chiave corretta. Ci si è posto il quesito se l'assunzione di indipendenza tra i vari gruppi di bit delle sottochiavi o meno influenzasse in qualche modo la convergenza dell'algoritmo. In [2] si suggerisce di *contare simultaneamente* le sottochiavi di tutte le S-Box che si attaccano, ovvero ipotizzando di attaccare 3 S-Box su 8 si potrebbero costruire separatamente 3 istogrammi, uno per ogni gruppo di 6 bit di sottochiave relative alla singola S-Box, seguendo il suggerimento dato però contare direttamente un l'occorrenza delle stringhe di 6*3 = 18 bit aumenterebbe la velocità di convergenza, questo equivale ad assumere che i tre gruppi di 6 bit non sono tra loro indipendenti. Nell'approccio con reti bayesiane invece l'assunzione di *dipendenza* a gruppi di 6 bit non

ha velocizzato la convergenza, al contrario è stata più lenta. Confrontando le tabelle 8.11 e 8.12 si può osservare che assumere che i bit siano indipedenti tra di loro a gruppi di 6 implica che l'algoritmo debba sfruttare un numero inferiore di coppie testo in chiaro/testo cifrato rispetto all'assunzione che i bit siano tra loro indipendenti a gruppi di 12. Come nel test precedente le coppie di testo in chiaro sono state scelte in modo random, e lo scopo era violare un DES ridotto a tre round con la medesima chiave nel terzo round del test precedente.

8.3.3 Test sul numero di ipotesi di propagazione

Un ipotesi di propagazione è una lista di differenze della forma

$$\Lambda = \left\{\lambda_I^1, \lambda_O^1, \lambda_I^2, \lambda_O^2, \ldots, \lambda_I^m, \lambda_O^m\right\},$$

dove λ_I^X è una differenza di 32 bit in input alla funzione di Feistel del round X-esimo, mentre λ_O^X è l'output della medesima funzione di Feistel. Ipotizzando di avere un insieme Ψ di ipotesi, ovvero

$$\Psi = \{\Lambda_1, \Lambda_2, \ldots, \Lambda_k\},$$

per numero di ipotesi viene inteso il numero di elementi di Ψ, i test sul numero di ipotesi è mirato a determinare il numero di elementi che deve contenere un insieme Ψ ad ogni attacco. Il test sul numero di ipotesi è stato effettuando fissando il numero di ipotesi da fare per singolo attacco. Anche in questo caso gli input all'algoritmo sono state coppie di testo in chiaro scelte in modo del tutto random. L'oggetto del test è stata la violazione di un DES ridotto a quattro round con sottochiave `k4 = 43213243212a`. La tabella 8.13 mostra che con un numero di campioni pari a 250 si hanno risultati ottimali. Si vuole far presente che comunque sono stati effettuati altri test in merito, con un numero di campioni anche elevato (ad esempio 6000) e si sono osservate prestazioni molto superiori a quelle ottenute con 250 campioni. Dalle tabelle del primo test si evince inoltre che l'aumentare il numero di ipotesi fa aumentare i tempi di esecuzione.

8.3.4 Test sulla scelta delle differenze in input

La scelta delle differenze in input è per questo tipo di attacco fondamentale, in quanto permette di estrarre ipotesi che siano più utili ai fini della convergenza dell'algoritmo. A supporto di quanto detto si confronti la tabella 8.13 con la tabella 8.14; si può osservare che il numero di attacchi per poter rompere il DES ridotto è in questo caso molto inferiore, in media, rispetto al caso esposto in precedenza. Questo speed-up è stato ottenuto fissando

$$L' = 02000000_f,$$
$$R' = 00000000_f.$$

Come ulteriore esempio la tabella 8.15 mostra un attacco al DES ridotto a 5 round, la chiave da determinare era `k5 = 12ab12345678`, in questo caso la differenza in input fissata era

$$L' = 19600000_f,$$
$$R' = 00000000_f;$$

si può osservare che il numero di attacchi, nonostante sia stato concatenato un round in più, è inferiore anche ai 4 round con ingressi random. La figura 8.16 riassume lo speed-up che si può ottenere se si vincolano gli ingressi scelti ad una particolare differenza, il grafico è espresso in scala logaritmica. Concludendo i quattro tipi di test effettuati hanno permesso di determinare la configurazione ottimale per il sistema bayesiano proposto per la crittoanalisi.

S-Box 1. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
2d	3	0,135	
1c	3	0,137	
d	4	0,27	
3e	3	0,171	
d	4	0,224	
1d	3	0,127	
2d	3	0,136	
37	3	0,127	
2c	3	0,122	
8	4	0,185	
2e	4	0,19	
	3,363636364	0,165818182	Media
	0,504524979	0,047771996	Varianza
	3	0,137	Mediana

(a) Test S-Box 1

S-Box 2. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
2e	3	0,153	
b	3	0,131	
28	2	0,057	
31	3	0,129	
9	3	0,119	
2c	3	0,117	
3c	3	0,122	
2d	4	0,181	
3	3	0,136	
26	3	0,119	
1a	3	0,124	
	3	0,126181818	Media
	0,447213595	0,029744977	Varianza
	3	0,124	Mediana

(b) Test S-Box 2

S-Box 3. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
f	3	0,119	
23	4	0,18	
38	4	0,183	
35	3	0,139	
0	3	0,125	
38	3	0,111	
3e	4	0,168	
2e	3	0,115	
27	3	0,11	
11	3	0,112	
8	3	0,115	
	3,272727273	0,134272727	Media
	0,467099366	0,028834323	Varianza
	3	0,119	Mediana

(c) Test S-Box 3

S-Box 4. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo(s)	
1f	2	0,063	
13	3	0,135	
0	3	0,142	
3f	3	0,133	
19	3	0,121	
37	2	0,098	
37	3	0,161	
6	2	0,101	
a	3	0,131	
13	3	0,153	
7	3	0,142	
	2,727272727	0,125454545	Media
	0,467099366	0,028320889	Varianza
	3	0,133	Mediana

(d) Test S-Box 4

S-Box 5. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
20	3	0,177	
19	3	0,133	
10	3	0,128	
15	3	0,134	
11	3	0,133	
b	3	0,132	
3e	2	0,07	
2a	3	0,133	
34	3	0,146	
3d	2	0,076	
14	3	0,129	
	2,818181818	0,126454545	Media
	0,404519917	0,029824029	Varianza
	3	0,133	Mediana

(e) Test S-Box 5

S-Box 6. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
6	3	0,115	
3e	4	0,177	
26	3	0,117	
15	2	0,057	
13	3	0,126	
6	3	0,129	
1c	2	0,058	
3a	3	0,154	
12	3	0,156	
3e	3	0,125	
21	3	0,141	
	2,909090909	0,123181818	Media
	0,53935989	0,037442805	Varianza
	3	0,126	Mediana

(f) Test S-Box 6

S-Box 7. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
35	3	0,14	
27	3	0,112	
1a	3	0,112	
f	3	0,115	
1c	3	0,113	
1a	3	0,115	
25	3	0,107	
f	3	0,125	
6	3	0,121	
27	2	0,06	
13	3	0,122	
	2,909090909	0,112909091	Media
	0,301511345	0,019679708	Varianza
	3	0,115	Mediana

(g) Test S-Box 7

S-Box 8. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
2c	2	0,065	
31	3	0,119	
14	3	0,119	
11	3	0,135	
31	3	0,119	
29	3	0,123	
39	3	0,117	
12	3	0,116	
27	3	0,144	
37	3	0,144	
9	2	0,056	
	2,818181818	0,114272727	Media
	0,404519917	0,028590526	Varianza
	3	0,119	Mediana

(h) Test S-Box 8

Figura 8.1: Tabelle riassuntive dei test sulle S-Box, struttura ARRAY e modalità DIRECT_BNET_ALGORITHM.

S-Box 1. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
3	4	0,128	
14	3	0,107	
21	3	0,11	
3b	3	0,1	
e	5	0,13	
21	3	0,113	
6	3	0,11	
27	3	0,097	
d	3	0,103	
9	3	0,11	
16	3	0,11	
	3,272727273	0,110727273	Media
	0,646669791	0,010267336	Varianza
	3	0,11	Mediana

(a) Test S-Box 1

S-Box 2. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
12	3	0,11	
a	2	0,092	
6	3	0,096	
26	3	0,095	
34	3	0,103	
35	2	0,092	
12	3	0,099	
a	3	0,098	
2f	3	0,105	
30	3	0,098	
2	5	0,282	
	3	0,115454545	Media
	0,774596669	0,05550561	Varianza
	3	0,098	Mediana

(b) Test S-Box 2

S-Box 3. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
2d	3	0,097	
15	3	0,095	
3c	3	0,103	
36	3	0,092	
23	3	0,103	
32	2	0,092	
35	3	0,098	
32	3	0,095	
c	3	0,098	
26	3	0,097	
2	3	0,098	
	2,909090909	0,097090909	Media
	0,301511345	0,00364567	Varianza
	3	0,097	Mediana

(c) Test S-Box 3

S-Box 4. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo	
28	2	0,09	
1f	3	0,095	
1a	3	0,1	
d	2	0,085	
13	3	0,11	
c	3	0,117	
33	3	0,133	
2b	3	0,104	
16	3	0,091	
1b	3	0,114	
28	2	0,091	
	2,727272727	0,102727273	Media
	0,467099366	0,014560844	Varianza
	3	0,1	Mediana

(d) Test S-Box 4

S-Box 5. Modalità ARRAY/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
2e	2	0,088	
3a	3	0,093	
11	3	0,103	
38	2	0,09	
2f	3	0,1	
e	2	0,088	
19	2	0,085	
6	2	0,087	
2e	3	0,1	
14	3	0,09	
11	3	0,088	
	2,545454545	0,092	Media
	0,522232968	0,006164414	Varianza
	3	0,09	Mediana

(e) Test S-Box 5

S-Box 6. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
10	2	0,09	
15	3	0,105	
3c	3	0,117	
32	2	0,1	
3b	3	0,097	
36	3	0,096	
2c	3	0,098	
17	3	0,1	
37	3	0,107	
27	3	0,098	
1d	3	0,097	
	2,818181818	0,100454545	Media
	0,404519917	0,007090326	Varianza
	3	0,098	Mediana

(f) Test S-Box 6

S-Box 7. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
15	3	0,117	
1	2	0,09	
33	3	0,096	
10	3	0,102	
28	3	0,095	
11	2	0,09	
5	3	0,1	
3b	3	0,097	
1a	3	0,1	
2a	3	0,098	
f	3	0,1	
	2,818181818	0,098636364	Media
	0,404519917	0,007256345	Varianza
	3	0,098	Mediana

(g) Test S-Box 7

S-Box 8. Modalità LIST/DIRECT_BNET_ALGORITHM			
Chiave	Attacchi	Tempo (s)	
26	3	0,108	
26	3	0,092	
24	3	0,095	
9	3	0,093	
16	2	0,087	
19	3	0,098	
27	5	0,103	
b	3	0,102	
24	3	0,103	
2d	2	0,085	
0	3	0,09	
	3	0,096	Media
	0,774596669	0,007389181	Varianza
	3	0,095	Mediana

(h) Test S-Box 8

Figura 8.2: Tabelle riassuntive dei test sulle S-Box, struttura LIST e modalità DIRECT_BNET_ALGORITHM.

S-Box 1. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
2a	4	0	
8	4	0	
2f	3	0	
20	4	0	
a	4	0	
2e	3	0	
1f	3	0	
37	4	0	
2f	3	0,002	
18	4	0	
	3,6	0,0002	Media
	0,516397779	0,000632456	Varianza
	4	0	Mediana

(a) Test S-Box 1

S-Box 2. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
22	3	0	
3b	3	0	
37	3	0	
16	3	0	
11	2	0	
3b	3	0	
11	3	0	
1f	3	0	
3b	3	0	
1d	3	0	
16	2	0	
	2,818181818	0	Media
	0,404519917	0	Varianza
	3	0	Mediana

(b) Test S-Box 2

S-Box 3. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
30	3	0	
6	3	0	
32	3	0	
1f	3	0	
19	2	0	
1	3	0	
34	2	0	
32	3	0	
a	3	0	
3a	3	0	
9	5	0	
	3	0	Media
	0,774596669	0	Varianza
	3	0	Mediana

(c) Test S-Box 3

S-Box 4. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
33	3	0	
4	3	0	
20	3	0	
13	3	0	
2f	2	0	
35	3	0	
1a	3	0	
2e	2	0	
1	3	0	
36	2	0	
2a	4	0	
	2,818181818	0	Media
	0,603022689	0	Varianza
	3	0	Mediana

(d) Test S-Box 4

S-Box 5. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
21	3	0	
c	2	0	
1c	3	0	
39	3	0	
15	3	0	
38	3	0	
3d	3	0	
15	3	0	
f	3	0	
1f	2	0	
30	3	0	
	2,818181818	0	Media
	0,404519917	0	Varianza
	3	0	Mediana

(e) Test S-Box 5

S-Box 6. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
1d	3	0	
13	3	0	
3d	3	0	
31	3	0	
b	3	0	
3e	3	0	
e	3	0	
9	2	0	
10	3	0	
1a	2	0	
38	3	0	
	2,818181818	0	Media
	0,404519917	0	Varianza
	3	0	Mediana

(f) Test S-Box 6

S-Box 7. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
30	2	0	
38	3	0	
31	3	0	
1	3	0	
0	3	0	
31	3	0	
d	3	0	
20	3	0	
35	3	0	
f	3	0	
2b	3	0	
	2,909090909	0	Media
	0,301511345	0	Varianza
	3	0	Mediana

(g) Test S-Box 7

S-Box 8. Modalità ARRAY/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
2c	3	0	
c	3	0	
2c	3	0	
3d	3	0	
28	2	0	
17	3	0	
21	3	0	
19	4	0	
32	3	0	
c	4	0	
f	3	0	
	3,090909091	0	Media
	0,53935989	0	Varianza
	3	0	Mediana

(h) Test S-Box 8

Figura 8.3: Tabelle riassuntive dei test sulle S-Box, struttura ARRAY e modalità LIKELIHOOD_SOLUTION.

S-Box 1. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
18	4	0	
d	4	0	
2b	4	0	
e	3	0	
f	4	0,002	
2a	4	0	
1	3	0	
2a	3	0	
3b	4	0	
25	3	0	
	3,6	0,0002	Media
	0,516397779	0,000632456	Varianza
	4	0	Mediana

(a) Test S-Box 1

S-Box 2. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
14	3	0	
d	3	0	
36	3	0,003	
21	3	0	
1c	3	0	
8	3	0	
10	3	0	
35	3	0	
3c	3	0	
19	3	0,003	
39	3	0	
	3	0,000545455	Media
	0	0,00121356	Varianza
	3	0	Mediana

(b) Test S-Box 2

S-Box 3. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
20	3	0	
8	3	0	
37	5	0	
2f	4	0	
4	3	0	
7	3	0	
1b	3	0	
e	4	0	
34	3	0	
3	3	0	
2b	3	0	
	3,363636364	0	Media
	0,674199862	0	Varianza
	3	0	Mediana

(c) Test S-Box 3

S-Box 4. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
14	3	0	
1b	3	0	
3a	3	0,003	
15	2	0	
25	3	0	
3d	3	0	
22	3	0,002	
2d	3	0	
38	3	0	
15	3	0,003	
2d	3	0	
	2,909090909	0,000727273	Media
	0,301511345	0,001272078	Varianza
	3	0	Mediana

(d) Test S-Box 4

S-Box 5. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
8	3	0	
11	2	0	
2f	3	0	
39	4	0	
28	3	0	
25	2	0	
16	3	0,01	
3	3	0	
1a	3	0	
25	2	0	
39	3	0	
	2,818181818	0,000909091	Media
	0,603022689	0,003015113	Varianza
	3	0	Mediana

(e) Test S-Box 5

S-Box 6. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
5	3	0	
21	3	0	
38	3	0	
e	3	0	
10	3	0	
14	3	0	
1a	3	0	
38	3	0	
1	4	0	
2f	3	0	
24	2	0	
	3	0	Media
	0,447213595	0	Varianza
	3	0	Mediana

(f) Test S-Box 6

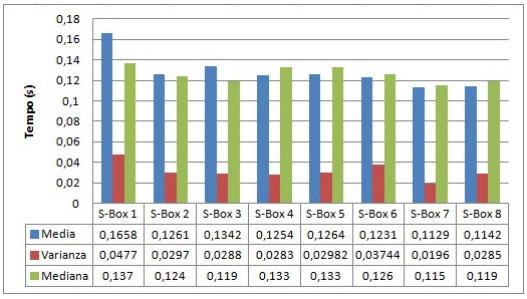
S-Box 7. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
26	3	0	
2b	3	0	
33	3	0	
c	3	0	
3a	3	0	
31	2	0	
20	4	0,01	
3c	3	0	
27	4	0	
36	3	0	
14	3	0	
	3,090909091	0,000909091	Media
	0,53935989	0,003015113	Varianza
	3	0	Mediana

(g) Test S-Box 7

S-Box 8. Modalità LIST/LIKELIHOOD_SOLUTION			
Chiave	Attacchi	Tempo (s)	
3d	3	0	
31	3	0	
2a	3	0	
1d	2	0	
12	3	0	
1f	3	0	
3c	3	0,01	
5	2	0	
27	3	0	
39	3	0	
1b	3	0	
	2,818181818	0,000909091	Media
	0,404519917	0,003015113	Varianza
	3	0	Mediana

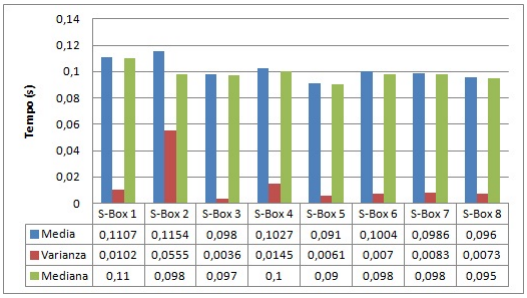
(h) Test S-Box 8

Figura 8.4: Tabelle riassuntive dei test sulle S-Box, struttura LIST e modalità LIKELIHOOD_SOLUTIONE.



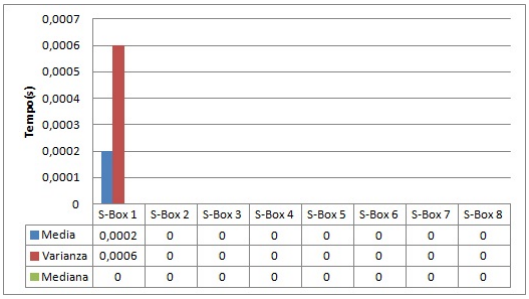
	S-Box 1	S-Box 2	S-Box 3	S-Box 4	S-Box 5	S-Box 6	S-Box 7	S-Box 8
Media	0,1658	0,1261	0,1342	0,1254	0,1264	0,1231	0,1129	0,1142
Varianza	0,0477	0,0297	0,0288	0,0283	0,02982	0,03744	0,0196	0,0285
Mediana	0,137	0,124	0,119	0,133	0,133	0,126	0,115	0,119

Figura 8.5: Istogramma dei tempi, struttura ARRAY e modalità DIRECT_BNET_ALGORITHM.



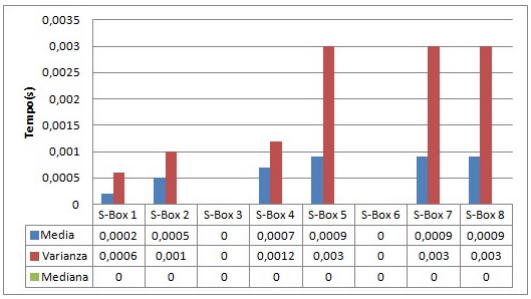
	S-Box 1	S-Box 2	S-Box 3	S-Box 4	S-Box 5	S-Box 6	S-Box 7	S-Box 8
Media	0,1107	0,1154	0,098	0,1027	0,091	0,1004	0,0986	0,096
Varianza	0,0102	0,0555	0,0036	0,0145	0,0061	0,007	0,0083	0,0073
Mediana	0,11	0,098	0,097	0,1	0,09	0,098	0,098	0,095

Figura 8.6: Istogramma dei tempi, struttura LIST e modalità DIRECT_BNET_ALGORITHM.



	S-Box 1	S-Box 2	S-Box 3	S-Box 4	S-Box 5	S-Box 6	S-Box 7	S-Box 8
Media	0,0002	0	0	0	0	0	0	0
Varianza	0,0006	0	0	0	0	0	0	0
Mediana	0	0	0	0	0	0	0	0

Figura 8.7: Istogramma dei tempi, struttura ARRAY e modalità LIKELIHOOD_SOLUTION.



	S-Box 1	S-Box 2	S-Box 3	S-Box 4	S-Box 5	S-Box 6	S-Box 7	S-Box 8
Media	0,0002	0,0005	0	0,0007	0,0009	0	0,0009	0,0009
Varianza	0,0006	0,001	0	0,0012	0,003	0	0,003	0,003
Mediana	0	0	0	0	0	0	0	0

Figura 8.8: Istogramma dei tempi, struttura LIST e modalità LIKELIHOOD_SOLUTION.

Test Tempi (secondi)						
	1 Campione	10 Campioni	100 Campioni	250 Campioni	500 Campioni	
MAP_HIST	1,7151	2,4652	82,6902	24,543	45,2988	Media
	0,582503495	1,30403066	212,6660758	7,233958406	14,9404686	Varianza
	1,4425	2,304	10,6455	26,624	42,1735	Mediana
ARRAY_HIST	1,2566	1,9836	8,6265	23,1202	44,6671	Media
	0,642108367	1,29919823	3,145899209	9,974891866	15,1408243	Varianza
	0,981	1,551	8,215	23,479	42,9165	Mediana
ARRAY_NO_SBOX_TOOL	0,0284	0,0518	0,2382	0,5503	1,2632	Media
	0,018506455	0,02285121	0,112702953	0,121336126	0,553025376	Varianza
	0,0225	0,044	0,2035	0,5325	1,2515	Mediana

Figura 8.9: Test Tempistici delle varie implementazioni, il confronto è tra struttura dati utilizzata e numero di campioni.

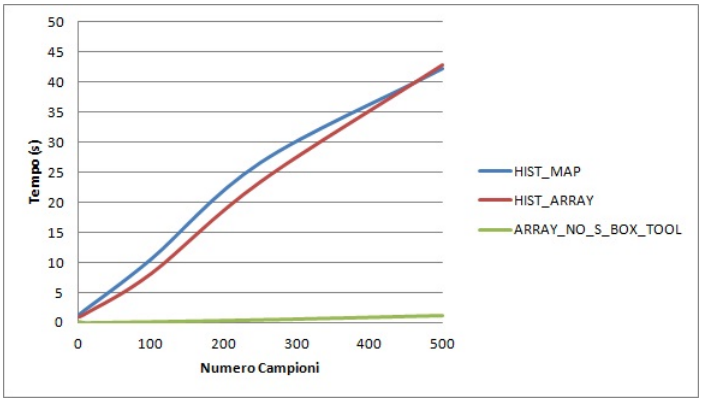


Figura 8.10: Andamento dei tempi di esecuzione (medi) in funzione del numero di campioni utilizzati.

Test Dipendenze - Ipotesi 8 istogrammi					
1 Campione	10 Campioni	100 Campioni	250 Campioni	500 Campioni	
234,3	36,4	15,3	15,6	13,2	Media
121,4862132	24,76197802	5,888784066	7,275529763	4,341018826	Varianza
183	28	14,5	15,5	13	Mediana

Figura 8.11: Tabella indipendenza 8 istogrammi, numero di attacchi.

Test Dipendenze - Ipotesi 4 istogrammi					
1 Campione	10 Campioni	100 Campioni	250 Campioni	500 Campioni	
730,9	119,8	52,7	34,2	39,5	Media
524,9009325	63,60258555	32,39015625	17,29354407	16,27711413	Varianza
656	106	42,5	31	41	Mediana

Figura 8.12: Tabella indipendenza 4 istogrammi, numero di attacchi.

Test Numero ipotesi - Numero attacchi					
1 Campione	10 Campioni	100 Campioni	250 Campioni	500 Campioni	
29213,8	2362,3806	510,8	309,7	252,6	Media
16781,1059	1668,062001	368,2076105	264,7611292	155,8569144	Varianza
25314	2154	501,5	230	227,5	Mediana

Figura 8.13: Studio del numero di ipotesi necessarie a violare un DES a 4 round.

Test Differenza Scelta - Numero Attacchi - 4 Round					
1 Campione	10 Campioni	100 Campioni	250 Campioni	500 Campioni	
324,7	41,4	21,4	16,3	18,8	Media
243,71616	41,55103956	13,11657137	13,04734967	9,784341231	Varianza
303,5	28	18,5	11,5	18,5	Mediana

Figura 8.14: Differenza scelta, test su DES a 4 round con differenza in input fissata.

Test Differenza Scelta - Numero Attacchi - 5 Round					
1 Campioni	10 Campioni	100 Campioni	250 Campioni	500 Campioni	
362,7	67,6	28	27	33,2	Media
198,1060603	25,72590566	11,59501809	9,637888197	13,13857763	Varianza
300	67	29,5	27	34	Mediana

Figura 8.15: Differenza scelta, test su DES a 5 round con differenza in input fissata.

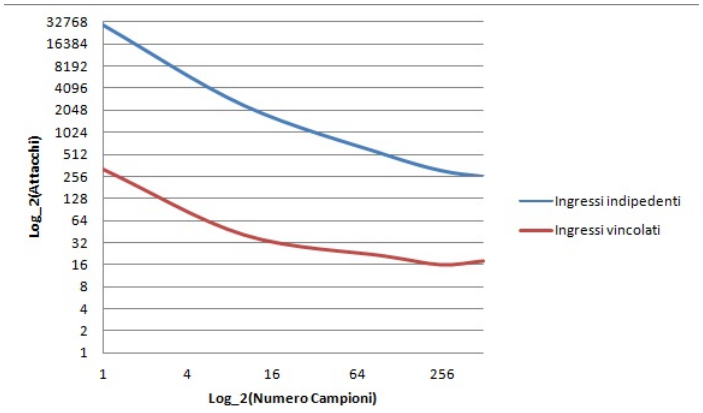


Figura 8.16: Confronto prestazioni tra attacchi attuati scegliendo gli input in modo indipedente e scegliendo gli input vincolati ad una determinata differenza di input.

Capitolo 9

Conclusione e Sviluppi futuri

Il lavoro svolto per la stesura di questa tesi ha permesso di dimostrare che la crittoanalisi differenziale del DES può essere implementata con un approccio alternativo a quello proposto originariamente in letteratura scientifica attraverso l'impiego di reti bayesiane. Il lavoro teorico svolto ha permesso di giustificare le implementazioni proposte e la valutazione sperimentale ha consentito di determinare la configurazione ottimale del sistema proposto. Riassumendo il punto iniziale è stato lo studio diversi lavori presenti in letteratura che dimostrano l'importanza all'interno del DES delle S-Box. Alcuni di questi lavori hanno avuto lo scopo di formalizzare le proprietà delle S-Box, altri di stabilire dei criteri di progettazione espressi in termini proprietà statistiche, e altri invece costituiscono dei tentativi di attacco sfruttando diverse vulnerabilità, alcuni di questi lavori si basano sulla linearizzazione dei comportamenti delle S-Box, altri sul determinare equazioni algebriche esatte che le definiscono e altri ancora sull'uso di DFT. Tra questi lavori è stata analizzato in modo più approfondito la crittoanalisi differenziale, che è stato il punto di partenza dello sviluppo del sistema proposto in questa tesi. Questo sistema è stato sviluppato in modo costruttivo, sviluppando prima una teoria di base e poi una piccola rete bayesiana per attaccare una singola S-Box. Si è generalizzato il ragionamento per attaccare la funzione di Feistel, utilizzata anch'essa a sua volta come strumento teorico per individuare le densità di probabilità da associare alla rete dell'intero DES e per sviluppare la funzione di campionamento di ipotesi sulla propagazione delle differenze. Tutti i tools sviluppati sono stati testati e dai risultati sperimentali si evince che per un opportuno numero di ipotesi il metodo proposto è equivalente all'originale, sia dal punto di vista di dati necessari a violare il DES, che dal punto di vista delle performance. Si ricorda che il sistema per attaccare il DES è stato sviluppato avvalendosi della tecnica di campionamento in avanti, ma altre tecniche di campionamento sono proposte in [13] e si ipotizza che l'uso di tecniche alternative di campionamento possano portare a risultati migliori. Ad esempio si potrebbe pensare di adoperare uno schema di campionamento per importanza piuttosto che bruto in avanti come quello utilizzato in questa tesi; probabilmente il campionamento per importanza permetterà di estrarre delle ipotesi più coerenti con quella che è la struttura della rete. Visto il grande one-

re computazionale dell'implementazione di uno schema di inferenza diagnostico si potrebbero approfondire gli studi in tal verso. Visto che si lavora inoltre su ipotesi tra loro indipendenti si può pensare ad una parallelizzazione dell'algoritmo. Non è da sottovalutare inoltre l'adattare la metodologia alla crittoanalisi lineare, altro metodo statistico, in quanto la crittoanalisi lineare di [14] converge alla chiave corretta del DES con un numero inferiore di dati rispetto alla crittoanalisi differenziale. Un altro importante sviluppo futuro potrebbe essere quello di completare l'attacco a tutti i sedici round del DES, oltre ovviamente a sfruttare l'algoritmo di schedulazione della chiave; per quest'ultimo punto in particolare si osservi che sia l'approccio originale sia l'approccio proposto in questa tesi non sfruttano l'algoritmo di schedulazione, ovvero in realtà le sottochiavi di 48 bit non sono tra loro indipendenti, ma sono legate da una relazione lineare. Altro aspetto utile potrebbe essere una implementazione in hardware dell'approccio proposto in questa tesi, infatti sia le funzioni campionamento che le soluzioni delle likelihood si potrebbero implementare facilmente in hardware, il che inoltre permetterebbe uno sfruttamento più intenso del parallelismo. Un altro miglioramento che si può apportare riguarda lo schema di conteggio per la costruzione dell'istogramma delle sottochiavi, stessi [2] suggeriscono un metodo di conteggio basato su componenti connesse di un grafo, schema che si sostiene occupare una quantità di memoria nettamente inferiore.

Appendice A

Tabelle S-Box

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	10

Tabella A.1: Tabella per S1

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

Tabella A.2: Tabella per S2

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

Tabella A.3: Tabella per S3

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

Tabella A.4: Tabella per S4

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

Tabella A.5: Tabella per S5

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

Tabella A.6: Tabella per S6

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

Tabella A.7: Tabella per S7

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

Tabella A.8: Tabella per S8

Appendice B

Risoluzione dei vincoli lineari

B.1 Soluzione sistemi quadrati

Nel capitolo quinto è stato introdotto il concetto di vincolo differenziale lineare, che altro non è che un sistema lineare. In questa appendice si vuole fornire un metodo generale per la risoluzione di sistemi lineari i cui coefficenti e incognite appartengono ad un campo di Galois, per un trattamento più rigoroso sui campi di Galois in ambito crittografico si possono consultare diversi testi quali [19] e [6], la teoria dei sistemi lineari è legata a quella dell'algebra lineare, per un riferimento in merito si veda [1]. Si supponga di lavorare nel campo $\mathbb{K} = GF(2^n)$, nel seguito indicheremo la somma con $\oplus$ mentre l'operazione di prodotto $\cdot$. Si supponga adesso di avere un sistema lineare quadrato della forma

$$\begin{cases} a_{11} \cdot x_1 \oplus \ldots \oplus a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 \oplus \ldots \oplus a_{2n} \cdot x_n = b_2 \\ \vdots \\ a_{n1} \cdot x_1 \oplus \ldots \oplus a_{nn} \cdot x_n = b_n \end{cases}$$

Ipotizzando che il rango del sistema sia n allora esso ammetterà soluzione unica, quindi sarà possibile applicare il metodo di eliminazione di Gauss, ove però l'aritmetica da considerare è quella dei campi finiti, in questo caso $GF(2^n)$, quindi per ogni coefficente, quindi quando si effettuano combinazioni lineari tra le varie righe del sistema, va tenuto conto che l'aritmetica è quella dei campi finiti, in particolare il calcolo dell'inverso moltiplicativo richiede l'algoritmo esteso di euclide. Nel caso dei vincoli differenziali lineari si ha sempre che

$$a_{ij} \in \{0, 1\}$$

di conseguenza il calcolo si semplifica notevolmente. Piuttosto che esporre un algoritmo generale si mostreranno alcuni esempi di risoluzione di vincoli differenziali lineari. Si supponga di dover risolvere un sistema lineare della forma

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0010 \\ 1101 \\ 0110 \end{pmatrix}$$

Ove l'aritmetica che si sta considerando è quella di $GF(2^4)$ in modulo $m(x) = x^4$. Vista la particolare struttura della matrice del sistema non è necessario effettuare inversi moltiplicativi, in quanto gli elementi sono tutti 1 oppure 0. Come prima cosa si sostutuisce la seconda alla riga del sistema la somma tra la prima riga e la seconda, mentre alla terza si sostituisce la somma tra la prima riga e la terza, il sistema equivalente risultante è

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0010 \\ 1111 \\ 0100 \end{pmatrix}$$

Le operazioni hanno azzerato tutti gli elementi della prima colonna al di sotto della prima riga. Adesso si sostituisce alla terza riga del sistema la somma tra la seconda riga e la terza, in tal modo si ottiene

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0010 \\ 1111 \\ 1011 \end{pmatrix}$$

Si è quindi riusciti a ricondurre la risoluzione di un sistema quadrato generico ad uno più semplice, triangolare, effettuando delle semplici operazioni di XOR tra le varie righe, ove è necessario. Scrivendo il sistema da risolvere in forma estesa

$$\begin{cases} x_1 & & \oplus x_3 = 0010 \\ & \oplus x_2 & \oplus x_3 = 1111 \\ & & \oplus x_3 = 1011 \end{cases}$$

Sostituendo il risultato della terza equazione nella seconda si ha per quest'ultima

$$x_2 \oplus 1011 = 1111 \Rightarrow x_2 = 0100$$

e sostituendo la terza equazione nella prima si arriva a

$$x_1 \oplus 1011 = 0010 \Rightarrow x_1 = 1001$$

quindi la soluzione del sistema è il vettore

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1001 \\ 0100 \\ 1011 \end{pmatrix}$$

B.2 Soluzione sistemi non quadrati

A differenza dell'esempio della sezione precedente, i v.d.l. non sono caratterizzati da una matrice quadrata, ma sempre rettangolare, di conseguenza la soluzione di questi vincoli non è unica. Essendo però il campo su cui si opera finito implica che tutte le soluzioni possono essere trovati per ricerca esaustiva, la complessità però di ricerca è elevata, se le incognite sono m ed il campo è tale che tutti gli elementi si possano rappresentare con n bit si ha che la ricerca ha un costo di 2^{mn} poichè bisogna iterare su tutte i vettori che possono rendere il sistema verificato. Avvalendosi sempre dell'eliminazione di Gauss, riadattata, è possibile rendere la matrice del sistema *a scala* [1], fatto questo è possibile

nuovamente utilizzare la risoluzione all'indietro, a differenza dell'esempio nella sezione precedente può però capitare che sia possibile fissare alcune incognite, e quindi si possono avere diverse risoluzioni all'indietro da dover effettuare. Come esempio si consideri il sistema a scala

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1001 \\ 0011 \\ 1001 \end{pmatrix}$$

che in forma estesa diviene

$$\begin{cases} x_1 \oplus x_2 = 1001 \\ x_2 \oplus x_3 = 0011 \\ x_3 \oplus x_4 = 1001 \end{cases}$$

La risoluzione all'indietro in questo sistema non può essere applicata direttamente, si osservi ad esempio l'ultima equazione nella quale sono presenti due incognite, una delle due può essere quindi fissata liberamente. Supponendo di fissare $x_4 = 0000$ si ha la seguente soluzione

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0011 \\ 1010 \\ 1001 \\ 0000 \end{pmatrix}$$

mentre se si fissa $x_4 = 0001$ si ottiene per soluzione

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0010 \\ 1011 \\ 1000 \\ 0001 \end{pmatrix}$$

e via dicendo, si otterranno tante soluzioni tanti quanti sono i valori ai quali si può fissare il parametro libero. Se i parametri liberi di un sistema sono più di 1 si otterranno tante soluzioni tanti quanti sono i modi in cui si possono fissare i vari parametri liberi.

Appendice C

La funzione di Espansione

C.1 Alcune proprietà della funzione di espansione

Anche se l'oggetto principale di questa tesi è stata la violazione delle S-Box, le altre trasformazioni presenti all'interno della funzione di Feistel hanno delle proprietà particolari che vengono approfondite qui, si osservi che lo spazio delle stringhe binarie di n bit è isomorfo all'insieme dei vettori con coefficenti in $\mathbb{Z}_2^n$, in questa tesi con isomorfismo si intende l'esistenza di una funzione di trasformazione che trasformi una stringa di n bit in un vettore di $\mathbb{Z}_2^n$. Sia infatti $x = b_n b_{n-1} \dots b_1$ una stringa di n bit, il vettore ottenuto partendo da x è

$$\vec{x} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

viceversa dato un vettore $\vec{x}$ è banale costruire in modo univoco una stringa se si prende ogni componente del vettore e la si concatena. Visto l'isomorfismo tra stringhe e vettori allora la linearità di $E(\cdot)$ può essere dimostrata se esiste una matrice E che identifica la trasformazione. Sia $\vec{x} \in \mathbb{Z}_2^n$ la seguente matrice identifica la funzione di espansione

$$E = \begin{pmatrix} \vec{e}_{32}^T \\ I_{e_{1:5}}^T \\ I_{e_{4:9}}^T \\ I_{e_{8:13}}^T \\ I_{e_{12:17}}^T \\ I_{e_{16:21}}^T \\ I_{e_{20:25}}^T \\ I_{e_{24:29}}^T \\ I_{e_{28:31}}^T \\ \vec{e}_1^T \end{pmatrix}$$

Nell'equazione precedente viene indicato con $\vec{e}_i$ il generico vettore colonna della base canonica, con l'apice T il trasposto di trasposto di tale vettore. Mentre

con $I_{e_{i:j}}$ si indica la matrice costruita come

$$I_{e_{i:j}} = \begin{pmatrix} \vec{e}_i & \vec{e}_{i+1} & \dots & \vec{e}_j \end{pmatrix}$$

e anche qui l'apice T indica il trasposto. Che E sia effettivamente la matrice da associare alla funzione di espansione lo si può vedere considerando il generico vettore, il cui trasformato tramite E è il generico output della funzione di espansione, da quanto detto segue la proprietà

Proposizione C.1. *La trasformazione $E(\cdot)$ è lineare.*

Assodato quindi che $E(\cdot)$ è lineare ha senso studiare il rango della matrice associata, che è banale a calcolarsi, perchè essendo costruita mediante *tutti* i vettori della base canonica, e alcuni suoi duplicati, è 32. Da questa osservazione segue che

Proposizione C.2. *Il rango di $E(\cdot)$ è 32*

Visto però che la dimensione dell'immagine è 32 e non 48 segue che la trasformazione di espansione non è suriettiva, cioè esiste almeno un vettore $\vec{y} \in \mathbb{Z}_2^{48}$ tale che per ogni $\vec{x} \in \mathbb{Z}_2^{32}$ si ha $E \cdot \vec{x} \neq \vec{y}$. Tuttavia per il teorema della dimensione [1] segue che il Kernel di $E(\cdot)$ è il solo vettore nullo, in [1] viene dimostrato che se il Kernel di una trasformazione lineare è il solo vettore nullo allora la trasformazione lineare è iniettiva (si dimostra anche il viceversa), quindi vale la

Proposizione C.3. *$E(\cdot)$ è iniettiva.*

C.2 Densità di probabilità legate alla funzione di Espansione

Sia $\vec{X}$ un possibile input della funzione di espansione, e si assuma che esso sia uniformemente distribuito in $\mathbb{Z}_2^{32}$, e sia $\vec{Y} = E \cdot \vec{X}$, la statistica di $\vec{Y}$, si vuole determinare la densità di probabilità di $\vec{Y}$.

$$Pr\left\{\vec{Y} = \vec{y}_0\right\} = Pr\left\{\vec{X} \in I^{-1}(\vec{y}_0)\right\}$$

Si possono verificare due casi

1. $I^{-1}(y_0) = \emptyset$
2. $I^{-1}(y_0) \neq \emptyset$

Nel primo caso si ha che

$$Pr\left\{\vec{X} \in \emptyset\right\} = 0 \Rightarrow Pr\left\{\vec{Y} = \vec{y}_0\right\} = 0$$

nel secondo caso si ha che $I^{-1}(y_0) = \{x_0\}$, in quanto $E(\cdot)$ è iniettiva, a questo punto si ha quindi che

$$Pr\left\{\vec{X} \in \{\vec{x}_0\}\right\} = \frac{1}{2^{32}} \Rightarrow Pr\left\{\vec{Y} = \vec{y}_0\right\} = \frac{1}{2^{32}}$$

Da quanto detto segue, la seguente proposizione

Proposizione C.4. *Ogni elemento dello spazio immagine della trasformazione lineare $E(\cdot)$ è uniformente distribuito. Ovvero per ogni $\vec{y}_0 \in E(\mathbb{Z}_2^{32})$ si ha che*

$$p_{\vec{Y}}(\vec{y}_0) = \frac{1}{2^{32}}$$

Si vuole ora discutere un aspetto interessante della funzione di espansione sia $\vec{x} \in \mathbb{Z}_2^{32}$, allora $E \cdot \vec{x} = \vec{y}$, poichè $\vec{y} \in \mathbb{Z}_2^{48}$ esso sarà della forma

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{48} \end{pmatrix}$$

e si indichi con $\vec{y_i}$, con $i = 1, \ldots, 8$ il vettore ottenuto mediante la trasformazione lineare

$$\vec{y_i} = I^T_{e_{(i-1)*6+1:(i-1)*6+6}} \cdot \vec{y}$$

che estrae l'i-simo gruppo consecutivo di 6 bit da $\vec{y}$. Consideriamo ora i vettori $\vec{y_i}$ e $\vec{y_{i+1}}$ e si supponga siano noti per il bit $\vec{y_i}$ gli ultimi 2 bit, questo due bit, riducono l'inceterezza su $\vec{y_{i+1}}$ in quanto, per costruzione della funzione di espansione, essi devono essere uguali ai primi due bit di $\vec{y_{i+1}}$, chiaramente vale anche il viceversa. Ovviamente $\vec{x}$ condiziona sia $\vec{y_i}$ e $\vec{y_{i+1}}$. Questa relazione può essere rappresentata con il grafo seguente.

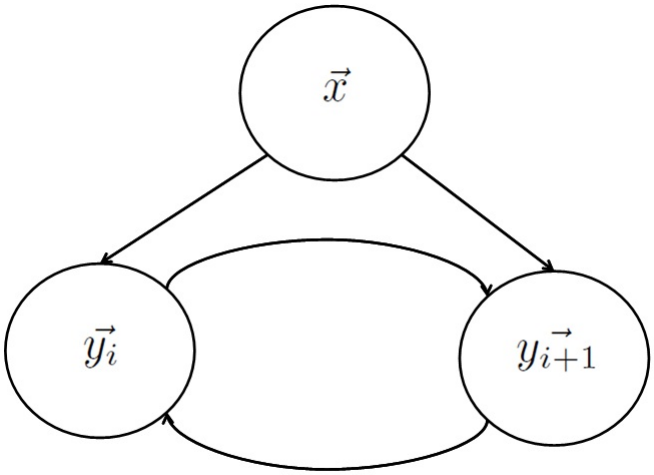


Figura C.1: Grafo di relazione della funzione di espansione

il quale è un grafo diretto, ma non aciclico. In conclusione i vari gruppi di ingressi alle varie S-Box non sono tra loro condizionatamente indipendenti dati l'input alla funzione di espansione. Questa osservazione ha reso necessario modellare il comportamento della funzione di Feistel con una rete apposita, e non concatenando varie S-Box B-Net come poteva essere intuitivo operare.

Elenco delle figure

Elenco delle tabelle

Lista degli Algoritmi

Bibliografia

[1] M. Abate. *Geometria*. McGraw-Hill, 1996.

[2] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, 1991.

[3] Ernest F. Brickell, Judy H. Moore, and M. R. Purtill. Structure in the s-boxes of the des, 1986.

[4] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, and Matthieu Rivain. Attack and improvement of a secure s-box calculation based on the fourier transform. In *Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, 2008.

[5] Nicolas T. Courtois and Gregory V.Bard. Algebraic cryptanalysis of the data encryption standard, 2007.

[6] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[7] Donald Davies and Sean Murphy. Pairs and triplets of des s-boxes. *Journal of Cryptology*, 8:1–25, 1995.

[8] M. Dawson and S.E.Tavares. An expanded set of design criteria for substitution boxes and their use in strengthening des-like cryptosystem, 1991.

[9] H. Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, May 1973.

[10] Howard M. Heys. *A tutorial on Linear and Differential Cryptanalysis*.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[12] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, 1999.

[13] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[14] M. Matsui. Linear cryptanalysis method for des cipher, 1993.

[15] National Institute of Standards and Technology. *FIPS PUB 46-3: Data Encryption Standard (DES)*. October 1999. supersedes FIPS 46-2.

[16] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 1949.

[17] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 1949.

[18] Alan Silvester. *Differential Cryptanalysis and Data Encryption Standard*, 2004.

[19] William Stallings. *Cryptography and Network Security Principle and Practice*. Pearson, 2011.