

Crittografia

Indice

1	Introduzione	3
2	Crittografia classica	5
2.1	Informazioni generali	5
2.2	Cifrari con shift	6
2.3	Cifrari monoalfabetici	7
2.3.1	Crittoanalisi	7
2.4	Cifrario Affine	7
2.4.1	Crittoanalisi	7
2.5	Cifrario di Playfair	8
2.5.1	Crittoanalisi	8
2.6	Cifrario di Hill	8
2.6.1	Crittoanalisi	9
2.7	Cifrario di Vigenère	10
2.7.1	Crittoanalisi	10
2.8	One-Time-Pad	15
3	Cifrari a blocchi	16
3.1	Cifratura di Feistel	16
3.1.1	Principi di Shannon	17
3.1.2	Struttura di Feistel	17
3.2	DES	18
3.2.1	Struttura di DES	18
3.2.2	Decifratura	20
3.2.3	Caratteristiche di DES	20
3.2.4	Modalità operative	21
3.2.5	Crittoanalisi	22
3.3	DES doppio	22
3.3.1	Attacco <i>meet in the middle</i>	22
3.4	DES-X	23
3.5	Blowfish	23
3.6	AES	23
3.7	Considerazioni sulla sicurezza	24

4	Crittografia Asimmetrica	25
4.1	RSA	25
4.2	Crittosistema di El-Gamal	27
4.2.1	Concetto di logaritmo discreto	27
4.2.2	Generatori	27
4.2.3	Generazione delle chiavi	27
4.2.4	Cifratura	28
4.2.5	Decifratura	28
4.3	Crittosistemi su curve ellittiche	28
5	Funzioni di hash	29
6	MAC	31
6.1	HMAC	33
7	Digital Signature Standard	34
8	Certificati	35
9	Secret Sharing	36
9.1	Schema (n, n)	36
9.2	Schema (k, n)	36

Capitolo 1

Introduzione

La crittografia fa parte, insieme alla crittoanalisi, della crittologia.

La **crittografia** è la scienza che studia tecniche e metodologie per **cifrare un testo in chiaro**, al fine di produrre un **testo cifrato comprensibile solo ad un ricevente legittimo**, il quale possiede l'informazione sufficiente (detta chiave) per decifrarlo, recuperando il testo in chiaro.

La **crittoanalisi** studia come decrittare un testo cifrato per ottenere il testo in chiaro: il verbo decrittare indica l'azione compiuta da un'entità che non possiede la chiave per recuperare, in modo legittimo, il testo in chiaro. In letteratura, suddetta entità viene indicata con il termine ascoltatore, oppure avversario o anche nemico. Lo scopo della crittografia è di produrre un messaggio cifrato m in modo che nessun avversario sia in grado di decrittare il contenuto.

La **stegonografia** (dal greco *scrivere nascosto*), indica l'insieme delle tecniche che permettono a due (o più) entità in modo tale da occultare, agli occhi di un ascoltatore, non tanto il contenuto (come nel caso della crittografia), ma l'esistenza stessa di una comunicazione. In altre parole, è l'arte di nascondere un messaggio all'interno di un altro *messaggio contenitore*. Alcuni esempi sono:

- disposizione dei caratteri
- inchiostro invisibile
- contrassegno dei caratteri
- nascondere messaggi all'interno di bit di file multimediali
- ...

Proprietà di sicurezza di un messaggio

1. **Confidenzialità**
2. **Autenticazione**
3. **Non ripudio**

4. **Integrità**

5. **Anonimia** (canali in cui le entità comunicanti devono poter nascondere la loro identità)

Capitolo 2

Crittografia classica

2.1 Informazioni generali

Queste informazioni sono valide per qualsiasi schema crittografico.

Operazioni di trasformazione

In generale, possono essere fatte due operazioni di *trasformazione*:

- **sostituzione:** ciascuno elemento del testo viene mappato un altro elemento
- **trasposizione:** gli elementi del testo in chiaro vengono scambiati di posto

⇒ è fondamentale che le operazioni siano invertibili per recuperare le informazioni

Spazio delle chiavi

Ogni schema di cifratura per essere robusto deve avere uno spazio delle chiavi abbastanza grande, altrimenti è vulnerabile ad un attacco di forza bruta; è una condizione necessaria ma non sufficiente.

Sicurezza

- ***unconditionally secure*:** è impossibile decifrare il testo criptato, indipendentemente dal tempo e risorse computazionali
- ***computationally secure*:** il tempo richiesto per decifrare il messaggio è superiore alla vita utile delle informazioni

Sicurezza perfetta

In un cifrario perfetto, osservando il messaggio cifrato l'avversario non ha alcuna possibilità di ottenere informazioni sul messaggio in chiaro; testo in chiaro e cifrato sono indipendenti tra loro; in termini probabilistici si può definire:

$$Prob(M|C) = Prob(M)$$

ovvero che la distribuzione delle probabilità non è influenzata dal fatto di conoscere il cifrato.

Principio di Kerckhoffs

La sicurezza di un crittosistema deve dipendere solo dalla sicurezza della chiave e non dalla segretezza dell'algoritmo usato.


Tipi di attacco

Si può fare una distinzione di diversi tipi di attacco in base alla conoscenza dell'avversario:

- ***Known Ciphertext Attack***
 - conosce solo il cifrato
- ***Known Plaintext Attack***
 - conosce anche il testo in chiaro; l'obiettivo è trovare la chiave
- ***Chosen Plaintext Attack***
 - può interagire con il sistema a far cifrare un messaggio a sua scelta
- ***Chosen Ciphertext Attack***
 - può interagire con il sistema e far decifrare un messaggio a sua scelta
- ***Chosen Text Attack***
 - è a conoscenza di coppie chiaro-cifrato

2.2 Cifrari con shift

Viene fatto uno shift delle lettere dell'alfabeto.

Cifrari con shift
 **Chiave K**
 $E_K(x) = (x+K) \bmod 26$
 $D_K(y) = (y-K) \bmod 26$
 $K \in \{1, \dots, 25\}$

Sono possibili solamente 25 chiavi, è vulnerabile ad un attacco di forza bruta.

2.3 Cifrari monoalfabetici

Viene fatta una sostituzione con un alfabeto cifrante. Sono possibili $26!$ chiavi (tutte le possibili combinazioni per l'alfabeto cifrante), ovvero 2^{88} .

2.3.1 Crittoanalisi

Potrei pensare di essere al sicuro perché abbiamo un ampio spazio delle chiavi (in realtà ad oggi si considera sicuro 2^{128}), ma questo schema si può rompere con una semplice tecnica di crittoanalisi: **analisi delle frequenze**.

L'assunzione è che un simbolo molto frequente nell'alfabeto di origine lo sarà anche in quello cifrato, shiftato di qualche lettera.

Si possono fare dei ragionamenti di crittoanalisi anche sulle caratteristiche della lingua utilizzata per risalire al testo in chiaro (ad esempio, in inglese dopo la t spesso c'è l' h ...).

2.4 Cifrario Affine

Sono un caso particolare dei cifrari a sostituzione monoalfabetica. La sostituzione è data da una funzione detta *affine*:

$$c_i = E(p_i) = (k_1 p_i + k_2) \bmod 26$$

→ la chiave è quindi data da **due costanti**.

La *decrittazione* avviene invece secondo la formula:

$$p_i = D(c_i) = (c_i - k_2) \cdot k_1^{-1}$$

con k_1^{-1} inteso come l'inverso modulo 26 di k_1 , ovvero quel numero x che soddisfa l'equazione:

$$(k_1 \cdot x) \bmod 26 = 1$$

Affinché questo sia possibile è necessario che k_1 e 26 siano primi tra loro.

È un altro modo per scrivere la mappatura da un alfabeto in chiaro ad uno cifrante; dato che deve essere rispettata la condizione di invertibilità, si ottiene un sottoinsieme di tutti i possibili $26!$ alfabeti.

2.4.1 Crittoanalisi

Si riconduce ad un cifrario di sostituzione monoalfabetica, è vulnerabile ad una semplice analisi delle frequenze.

2.5 Cifrario di Playfair

Si ragiona su coppie di caratteri invece che su singoli caratteri: l'idea è che in questo modo si ha uno spazio di $26 \cdot 26$, è ancora possibile fare analisi delle frequenze ma è necessario un testo più ampio.

M	T	Z	C	L
H	A	U	J	E
K	F	G	N	R
V	W	X	B	D
Q	O	S	Y	P

testo in chiaro: DO MA NI
testo cifrato: WP TH BN

Si costruisce un rettangolo 5×5 , mettendo per convenzione nella stessa cella le lettere i e j ; per cifrare viene tracciato un rettangolo tra la coppia di lettere, e la cifratura corrisponde ai vertici opposti.

Le lettere ripetute vanno separate da una lettera di riempimento (es. *mamma* → *mamxma*)

Nel caso di lettere sulla stessa riga o colonna:

- stessa riga → si sostituisce con lettere a destra
- stessa colonna → si sostituisce con lettere sottostanti

2.5.1 Crittoanalisi

È più sicuro rispetto alla cifratura monoalfabetica ($26 \cdot 26 = 676$ *digrammi*); è tuttavia sempre possibile fare un'analisi delle frequenze dei digrammi.

2.6 Cifrario di Hill

Permette di sostituire m lettere in chiaro con m lettere cifrate, usando m equazioni lineari.

Es. per $m=2$, (x_1, x_2) (y_1, y_2) :

$$\begin{aligned} Y_1 &= 11x_1 + 3x_2 \bmod 26 \\ Y_2 &= 8x_1 + 7x_2 \bmod 26 \end{aligned} \quad (y_1, y_2) = (x_1, x_2) \begin{pmatrix} 11 & 3 \\ 8 & 7 \end{pmatrix}$$

La chiave del cifrario è la matrice, i cui coefficienti vengono scelti in modo arbitrario.

Esempio

$$K = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \quad K^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

Il testo in chiaro è: *ciao*

$$\begin{pmatrix} 2,8 \end{pmatrix} \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \quad Y = (22+24, 16+56) = (20, 20)$$

$$\begin{pmatrix} 0,14 \end{pmatrix} \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \quad Y = (0+42, 0+98) = (16, 20)$$

Il testo cifrato è: *UUQU*

Le fasi di cifratura e decifratura, in generale, sono:

$$y = E_k(x) = xK$$

$$x = D_k(y) = yK^{-1}$$

2.6.1 Crittoanalisi

È vulnerabile ad un attacco di tipo *known plaintext*: conoscendo la coppia chiaro-cifrato, tramite un calcolo matematico, è possibile calcolare la chiave.

L'avversario conosce $m=2$

$$\begin{pmatrix} 5 & 17 \\ 4 & 3 \end{pmatrix} K = \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix}$$

$$MK = C$$

$$M^{-1}MK = M^{-1}C \quad \begin{pmatrix} 5 & 17 \\ 4 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 23 & 17 \\ 4 & 21 \end{pmatrix}$$

$$K = M^{-1}C$$

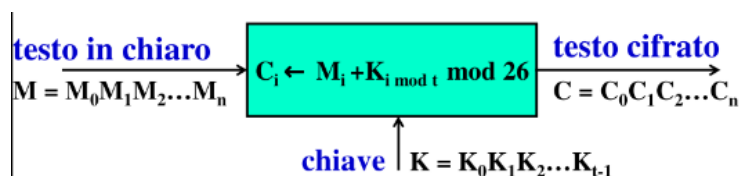
$$\begin{pmatrix} 23 & 17 \\ 4 & 21 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$$

2.7 Cifrario di Vigenère

Testo in chiaro:	CODICE MOLTO SICURO	Chiave:	REBUS
	CODIC EMOLT OSICU RO	testo in chiaro	
	REBUS REBUS REBUS RE	chiave	
	TSECU VQPFL FWJWM IS	testo cifrato	

Viene applicato un cifrario di shift per ogni singolo caratteri con la parola chiave; si ottiene che si hanno uguali caratteri cifrati che corrispondono a lettere in chiaro diverse e viceversa.

Matematicamente si può rappresentare in questo modo:



Sono possibili 26^t chiavi, con t lunghezza della chiave.

2.7.1 Crittoanalisi

È vulnerabile ad un attacco di tipo *known ciphertext*. Sfrutta dei metodi statistici per:

- determinare la lunghezza della chiave
 - indice di coincidenza
- determinare i caratteri della chiave
 - indice mutuo di coincidenza

Indice di coincidenza

Indice di coincidenza di una stringa $x_1x_2\dots x_n$

$IC(x_1x_2\dots x_n)$ = probabilità che due caratteri,

presi a caso in $x_1x_2\dots x_n$, siano uguali

$$= \frac{\sum_{i=0}^{25} \binom{f_i}{2}}{\binom{n}{2}} = \frac{\sum_{i=0}^{25} f_i(f_i - 1)}{n(n-1)}$$

f_i = numero occorrenze carattere i
nella stringa $x_1x_2\dots x_n$

f_0 = occorrenza della a
 f_1 = occorrenza della b
...



Stelvio Cimato DI Università degli Studi di Milano. Sede di

L'indice di coincidenza è utile perché permette di capire qual è la lingua utilizzata; si fa un calcolo con n che tende ad infinito, e ciascuna lingua ha un suo valore per l'indice di coincidenza.

Si ok ma...come funziona=?

- si prova calcolare l'IC ponendo la lunghezza della chiave $t = 1$

Se $t = 1$ allora $IC(C_0C_1\dots C_n) = IC(M_0M_1\dots M_n)$

$$IC(C_0C_1\dots C_n) \approx \begin{cases} 0.075 & \text{se } t=1 \\ 0.038 & \text{se } t \neq 1 \end{cases}$$

Non proprio!
Comunque lontano da 0.075

Figura 2.1: 0.038 = caratteri casuali, 0.075 = italiano

se ottengo un valore noto vuol dire che ho trovato la lunghezza della chiave, altrimenti proseguo ponendo $t = 2$

- calcolo l'IC ponendo $t = 2$

Se $t = 2$ allora $IC(C_0C_2...) = IC(M_0M_2...)$
 $IC(C_1C_3...) = IC(M_1M_3...)$

$$\begin{aligned} IC(C_0C_2...) &\approx \\ IC(C_1C_3...) &\approx \end{aligned} \begin{cases} 0.075 & \text{se } t=2 \\ 0.038 & \text{se } t \neq 2 \end{cases}$$

Comunque lontano da 0.075

- ...
- si continua iterativamente fino ad ottenere un valore noto

$$t = 5 ? \quad \begin{cases} IC(C_0C_5...) = 0.0710 \\ IC(C_1C_6...) = 0.0721 \\ IC(C_2C_7...) = 0.0805 \\ IC(C_3C_8...) = 0.0684 \\ IC(C_4C_9...) = 0.0759 \end{cases}$$

Tutti vicini a 0.075
 $t = 5$

Indice mutuo di coincidenza

Si usa per determinare quali sono i caratteri che compongono la chiave.

Indice mutuo di coincidenza di $x_1x_2...x_n$ e $y_1y_2...y_n$

$IMC(x_1x_2...x_n; y_1y_2...y_n)$ = probabilità che un carattere in $x_1x_2...x_n$, ed uno in $y_1y_2...y_n$, presi a caso, siano uguali

$$= \frac{\sum_{i=0}^{25} f_i \cdot f'_i}{n \cdot n'}$$

f_i = numero occorrenze carattere i in $x_1x_2...x_n$
 f'_i = numero occorrenze carattere i in $y_1y_2...y_n$

E questo come funziona?

Supponiamo quindi di conoscere la dimensione t della chiave,

➤ La chiave è $K = (k_0, k_1, \dots, k_{t-1})$, con ogni k_i che indica il numero da aggiungere per ottenere il testo cifrato


Si divide il ciphertext in t stringhe y_0, y_1, \dots, y_{t-1}

Si prendono due caratteri casuali di y_i e y_j :

➤ La probabilità che essi siano A è $p_{-k_i} p_{-k_j}$

➤ La probabilità che essi siano B è $p_{1-k_i} p_{1-k_j}$


- provo a porre $K_0 - K_1 = 0$

testo cifrato $C_0 C_1 \dots C_n$ 

Se $K_0 - K_1 = 0$ allora $\text{IMC}(C_0 C_1 \dots; C_1 C_{t+1} \dots) \approx 0.075$

$$\text{IMC}(C_0 C_1 \dots; C_1 C_{t+1} \dots) \begin{cases} \approx 0.075 & \text{se } K_0 - K_1 = 0 \\ \approx < 0.047 & \text{se } K_0 - K_1 \neq 0 \end{cases}$$


- provo a porre $K_0 - K_1 = 1$

testo cifrato $C_0 C_1 \dots C_n$ $Y_i \leftarrow C_i - 1 \bmod 26$ 

Se $K_0 - K_1 = 1$ allora $\text{IMC}(Y_0 Y_1 \dots; C_1 C_{t+1} \dots) \approx 0.075$

$$\text{IMC}(Y_0 Y_1 \dots; C_1 C_{t+1} \dots) \begin{cases} \approx 0.075 & \text{se } K_0 - K_1 = 1 \\ \approx < 0.047 & \text{se } K_0 - K_1 \neq 1 \end{cases}$$

- provo a porre $K_0 - K_1 = 2$

testo cifrato $C_0 C_1 \dots C_n$ $Y_i \leftarrow C_i - 2 \bmod 26$ 

Se $K_0 - K_1 = 2$ allora $\text{IMC}(Y_0 Y_1 \dots; C_1 C_{t+1} \dots) \approx 0.075$

$$\text{IMC}(Y_0 Y_1 \dots; C_1 C_{t+1} \dots) \begin{cases} \approx 0.075 & \text{se } K_0 - K_1 = 2 \\ \approx < 0.047 & \text{se } K_0 - K_1 \neq 2 \end{cases}$$

- ... si continua iterativamente fissando un sottotesto e modificando l'altro sottraendogli da 0 a 25; quando si trova un valore di IMC vicino a 0.075 (lingua italiana) significa che abbiamo trovato il corretto valore di shift $k_i - k_j$

A questo punto si possono scrivere $t-1$ equazioni in t incognite; si può riscrivere tutto in funzione di k_0 , provando tutti i possibili 26 valori e vedendo quale soddisfa il sistema.

$K_0 - K_1 = 5$
 $K_1 - K_2 = 6$
 $K_2 - K_3 = 9$
 \dots
 $K_{t-2} - K_{t-1} = 5$

$t-1$ equazioni in t incognite

Riesco ad esprimere tutti i K_i in funzione di K_0 !

Quanto vale K_0 ?

Provo tutti i possibili 26 valori !

Stelvio Cima 65

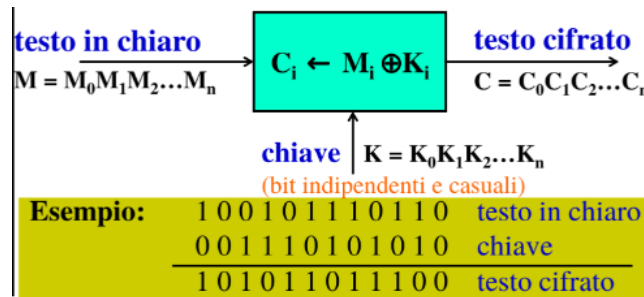
In sintesi:

- Determinare la lunghezza della chiave t
 - uso dell'indice di coincidenza
- Determinare il valore della chiave $K_0 K_1 K_2 \dots K_{t-1}$
 - calcolo delle differenze $K_0 - K_1, K_1 - K_2, \dots, K_{t-2} - K_{t-1}$
 - uso dell'indice mutuo di coincidenza
 - calcolo di K_0 : prova le 26 possibilità

2.8 One-Time-Pad

È definibile come il cifrario perfetto: viene presa una chiave lunga quanto il messaggio, i cui bit sono scelti in modo casuale.

Il messaggio risulta indecifrabile, perché il crittoanalista non può fare alcuna assunzione; il cifrario è *unconditionally secure*, perché a prescindere da tempo e risorse computazionali, non è possibile distinguere quale sia il messaggio originale (il massimo che si può ottenere è tutte le possibili frasi di senso compiuto, ma senza poter stabilire qual è quella originale).



Il problema è che è necessario avere la chiave lunga quanto il messaggio che ci si vuole scambiare.

Capitolo 3

Cifrari a blocchi

I cifrari a blocchi operano su blocchi di n bit di input per produrre blocchi di altrettanti n bit in output; la trasformazione deve essere reversibile e dipende dalla chiave utilizzata.

Mappaggi

Si può pensare di fare un mappaggio tra tutti i possibili blocchi in input e il corrispondente cifrato, che deve essere unico; facendo questo mappaggio il numero di trasformazioni possibili è 2^n

Testo chiaro	cifrato
00	11
01	10
10	00
11	01

Il problema è che per blocchi di piccole dimensioni equivale a fare una cifratura a sostituzione (vulnerabile ad analisi statistica), mentre per blocchi grandi diventa un problema la gestione della chiave: **per blocchi di n bit la chiave è di dimensione $n * 2^n$** (la chiave equivale alla tabella delle sostituzioni).

3.1 Cifratura di Feistel

L'idea è quella di approssimare un sistema ideale di cifratura, facendo uso di cifrature in sequenza per ottenerne una più complessa rispetto ad una singola trasformazione.

La chiave scelta permette di calcolare per ogni blocco di input uno di output; è una trasformazione robusta dato che è difficile da invertire se non si conosce la chiave.

3.1.1 Principi di Shannon

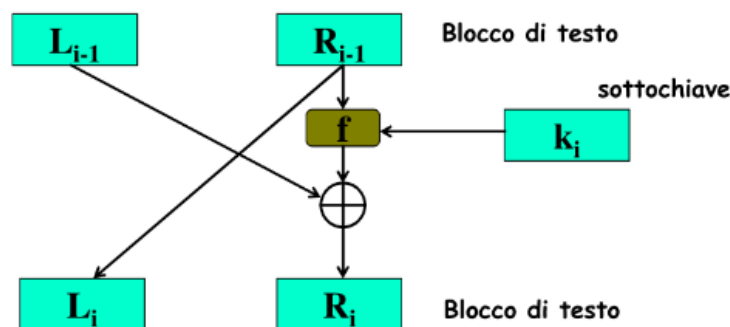
Feistel alterna **permutazioni** e **sostituzioni**; applicano i principi di Shannon per contrastare l'analisi statistica:

- **Diffusione:** ogni bit del cifrato dipende da più bit in chiaro; viene estesa la correlazione statistica, non c'è correlazione 1 : 1
- **Confusione:** si vuole aumentare la difficoltà nel capire la correlazione tra testo in chiaro/cifrato e la chiave; è reso possibile dal fatto che dalla chiave vengono generate più sottochiavi, per un cui un cambiamento di un singolo bit della chiave genera cambiamenti a cascata su tutte le sottochiavi e il cifrato

3.1.2 Struttura di Feistel

- Blocchi e chiave di grande dimensioni aumentano la sicurezza ma diminuiscono le prestazioni
- Tutte le fasi hanno la stessa struttura
- A partire dalla chiave vengono prodotte tante sottochiavi quanti sono i *round*
- La funzione di round deve essere non lineare (altrimenti sarebbe facilmente invertibile)

La struttura di Feistel è la seguente:



Il blocco di input viene diviso in due (*left* e *right*), e l'output viene ottenuto secondo la procedura mostrata in figura.

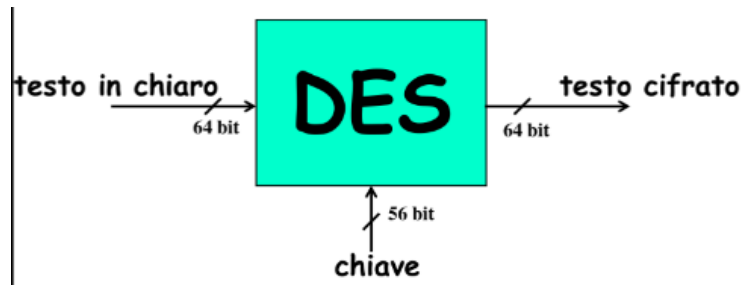
Questa operazione viene ripetuta per tutti i blocchi, per implementare i principi di Shannon.

È da notare come viene sfruttata la proprietà dello XOR per cui è possibile decifrare a prescindere della funzione f utilizzata.

- **Cifratura:** basta implementare un solo round, che verrà ripetuto su tutti i blocchi

- **Decifratura:** usa lo stesso algoritmo ma con le sottochiavi in ordine inverso

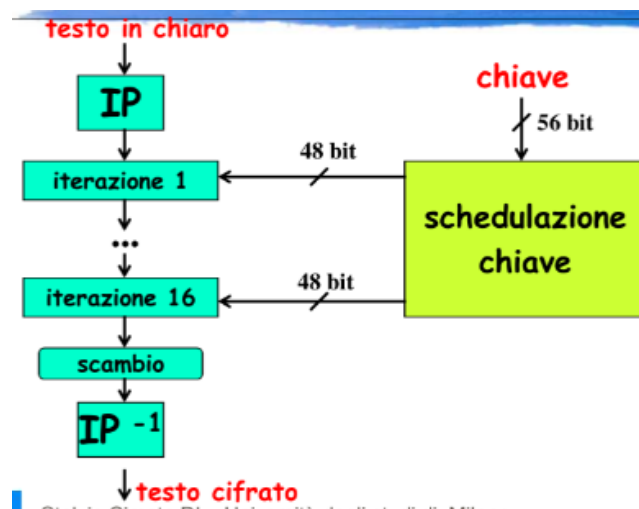
3.2 DES



Usa la struttura di Feistel, con:

- blocchi di 64 bit
- chiave di 64 bit, di cui 56 usati effettivamente dall'algoritmo
 - l'ultimo bit di ciascun byte viene usato come bit di parità (è lo XOR dei 7 precedenti)
 - spazio delle chiavi di 2^{56}

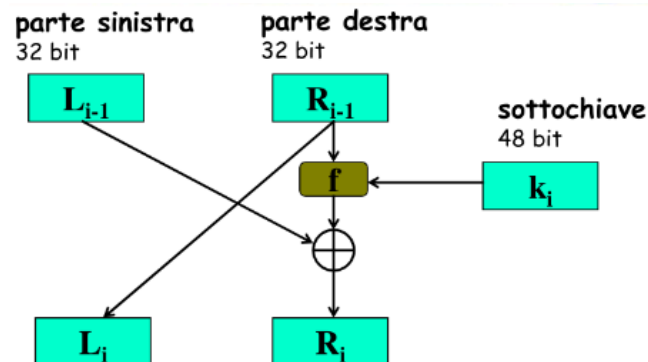
3.2.1 Struttura di DES



IP e IP^{-1}

Una permutazione iniziale ed inversa alla fine; usano tabelle fissate per fare permutazioni dei bit.

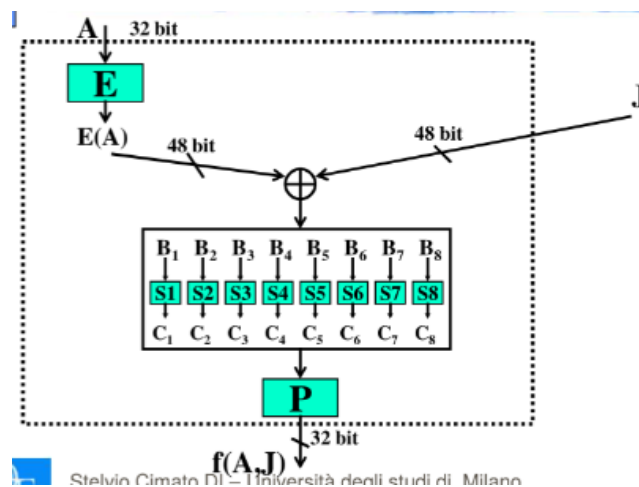
Singola iterazione



La funzione f

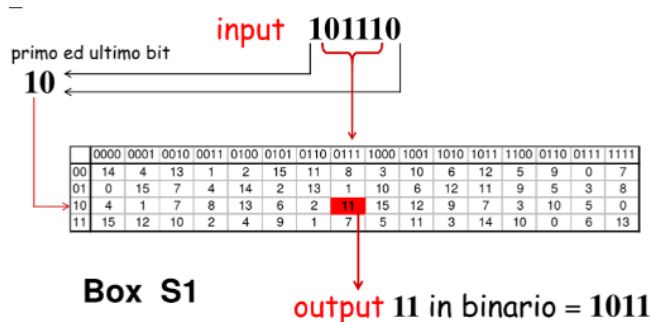
I 32 bit di input vengono espansi a 48, e poi vengono messi in XOR con la chiave.

I 48 bit risultanti vengono gestiti da 8 *S-Box* (6 bit ciascuna), ciascuna delle quali produce 4 bit per ottenerne 32; questi 32 bit vengono dati ad una funzione di permutazione e si ottiene il risultato finale.



S-Box

I 6 bit dati in input alle s-box fungono da indici di riga e di colonna per una tabella contenente tutti i possibili 16 valori dei 4 bit di output

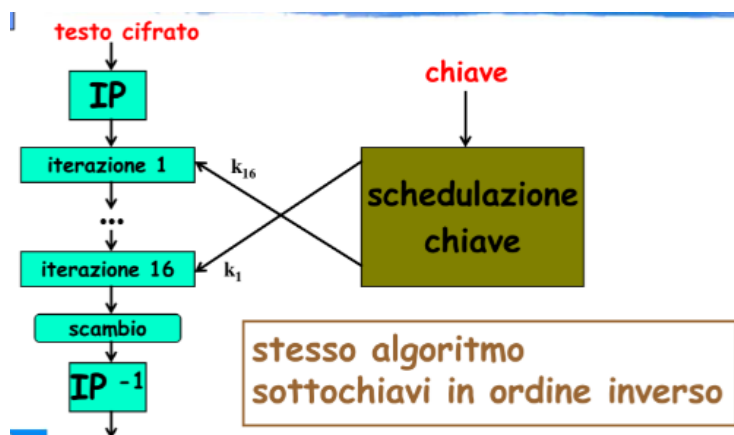


Le s-box hanno due proprietà importanti:

- cambiando un solo bit di input variano almeno due bit nell'output
- il numero di input per il quale il bit di output è 0 o 1 è circa lo stesso

3.2.2 Decifratura

Funziona allo stesso identico modo, ma usando l'ordine inverso delle sottochiavi.



3.2.3 Caratteristiche di DES

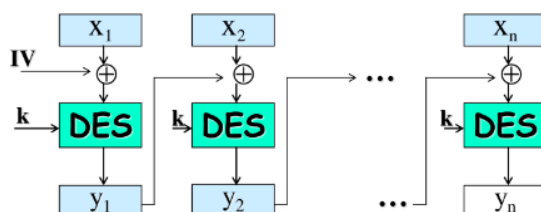
Cosiddetto **effetto valanga**: piccoli cambiamenti nel testo in chiaro provocano un grande cambiamento nel cifrato; lo stesso vale per la chiave e l'algoritmo di schedulazione.

Rende difficile per un attaccante fare delle analisi tra i bit input e di output.

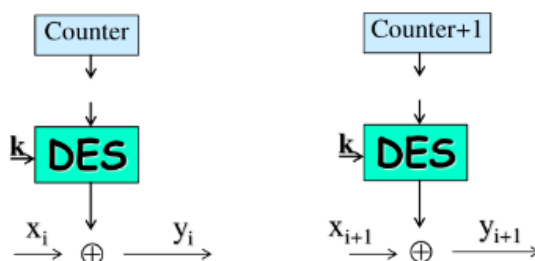
3.2.4 Modalità operative

Fino ad ora abbiamo visto come viene cifrato un singolo blocco; ora vediamo le modalità con cui cifrare più blocchi. Sono valide per tutti i cifrari simmetrici.

- **Electronic Codebook Chaining (ECB):** ciascun blocco di 64 bit viene cifrato in maniera indipendente; ha il vantaggio di evitare la propagazione degli errori ma ha il problema di essere deterministico (a stesso blocco e chiave, corrisponde sempre lo stesso output); non è sicuro se usato con messaggi lunghi
- **Cipher Block Chaining (CBC):** l'input si ottiene facendo lo XOR con il precedente blocco cifrato; per il primo blocco si usa un *initialization vector* da scegliere in modo casuale; c'è propagazione degli errori ma la cifratura non è più deterministica



- **Cipher Feedback (CFB):** viene usato un registro a scorrimento che permette di poter usare la lunghezza del blocco a piacere; usa sempre un *iv* per rompere il determinismo, ha propagazione degli errori; l'idea è quella di avvicinarsi ad un cifrario a flusso dato che si può scegliere la lunghezza del blocco a piacere
- **Output Feedback (OFB):** ha il vantaggio di non propagare gli errori di trasmissione dei bit (usato ad esempio per trasmissione con i satelliti); ha lo svantaggio di essere più vulnerabile ad una modifica del flusso
- **Counter (CTR):** si usa un contatore delle dimensioni del blocco in chiaro; per ogni blocco successivo il contatore viene incrementato



Ha il vantaggio di essere sicuro ed efficiente

3.2.5 Crittoanalisi

Avendo una chiave di 56 bit DES è vulnerabile ad un attacco di brute force.

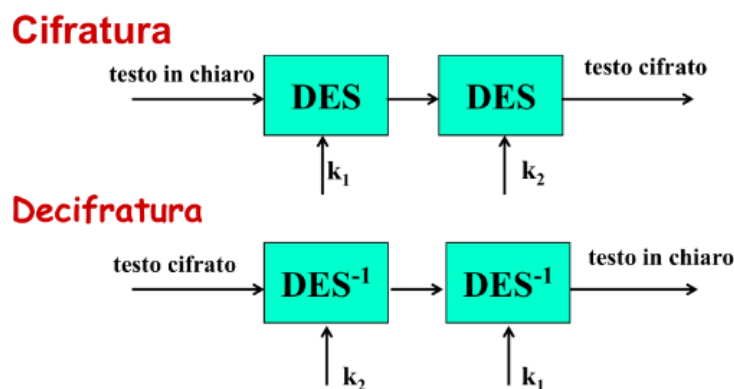
- La modalità ECB, essendo deterministica, non è resistente al CPA
- Le altre modalità sono tutte resistenti a CPA, dato che l'*iv* introduce un fattore di casualità
- Rimangono comunque degli schemi vulnerabili al CCA

Alcune tecniche di crittoanalisi (applicabili a tutti i cifrari a blocchi) sono:

- crittoanalisi differenziale → CPA, anche se necessita di 2^{47} testi in chiaro (poco applicabile nella realtà)
- crittoanalisi lineare → KPA, poco applicabile perché necessita di 2^{43} testi noti

3.3 DES doppio

Consiste nel cifrare due volte, applicando DES con due chiavi distinte.



È dimostrato che l'insieme delle permutazioni delle chiavi DES non è chiuso per composizione, ovvero che non sempre esiste una chiave $k_3 \equiv k_1 \cdot k_2$ (nel senso che dà in output lo stesso cifrato della doppia cifratura).

3.3.1 Attacco *meet in the middle*

È un attacco di tipo *known plaintext*.

Data una coppia nota (m, c) vengono eseguite tutte le cifrature per tutti i possibili 2^{56} valori di k_1 , e memorizzate in una tabella.

Vengono eseguite tutte le possibili decifrazioni per i 2^{56} possibili valori di k_2 , e si cerca una corrispondenza nella tabella.

Anche se si ha uno spazio delle chiavi di 2^{112} , la complessità computazionale rimane di 2^{56} ; lo stesso vale in termini di spazio per salvare le righe della tabella.

3.4 DES-X

È una tipologia di DES che usa una tecnica detta *key whitening* per migliorare la sicurezza rispetto al DES classico.

In questo caso, il cifrario non viene rafforzato contro attacchi analitici (analisi differenziale e lineare), ma solo contro gli attacchi a forza bruta. Vengono applicati due XOR prima e dopo l'esecuzione dell'algoritmo DES con due chiavi aggiuntive (una per XOR): $DES - X(M) = k_2 \oplus DES(M \oplus k_1)$

3.5 Blowfish

È un cifrario che utilizza diverse tecniche tra cui la rete di Feistel e s-box dipendenti dalla chiave, cosa che lo rende uno degli algoritmi più sicuri in circolazione.

I blocchi hanno una dimensione di 64 bit, mentre la chiave va dai 32 ai 448 bit. L'algoritmo consiste in due fasi:

1. **Espansione della chiave:** la chiave viene espansa in 18 sottochiavi a 32 bit e vengono generate 4 s-box dipendenti da essa
2. **Cifratura:** sono previsti 16 round dove in ognuno viene usata una sottochiave; al termine dei round vengono usate le ultime due chiavi rimaste; la decifrazione funziona allo stesso modo ma usando le sottochiavi in ordine inverso

Questo algoritmo è compatto e veloce; è inoltre resistente agli attacchi di forza bruta dato che è possibile arrivare ad uno spazio delle chiavi di 2^{448}

3.6 AES

Non è un cifrario di Feistel, ma opera in parallelo sull'intero blocco di input.

- blocchi e chiave da 128 bit
- 10 round
- vengono schedate 44 sottochiavi a 32 bit

L'unità di base di AES è il byte; viene rappresentato sotto forma di una matrice 4×4 , che prende il nome di *stato*; tutte le operazioni prendono e restituiscono un byte, definendo un campo finito su valori fino a 2^8 .

Ogni round è una composizione parallela di 4 operazioni, che prendono in input la matrice di stato:

- **SubBytes**
- **ShiftRows**
- **MixColumns**
- **AddRound Key**

Fanno eccezione una *AddRound Key* prima dei round, e l'ultimo dove manca la *MixColumns*.

SubBytes

Viene fatta la sostituzione dei byte con un altri, mediante un s-box tabellare, costruita in modo da resistere alla crittoanalisi.

ShiftRows

Viene fatto uno shift delle righe:

- la seconda shiftata di 1
- la terza shiftata di 2
- la quarta shiftata di 3

In questo modo con questa semplice permutazione i 4 byte di una colonna vengono riposizionati su colonne differenti.

MixColumns

Una colonna viene moltiplicata per un polinomio fissato, viene fatta una combinazione lineare.

AddRoundKey

Viene fatto lo XOR tra una colonna e la sottochiave di round.

3.7 Considerazioni sulla sicurezza

Ciò che viene fatto con i cifrari a blocchi è mettere il risultato in XOR con il successivo *plaintext*; è un modo per approssimare il cifrario perfetto dove si ha una chiave casuale lunga quanto il messaggio (sicurezza perfetta).

Usare i cifrari a blocchi in questo modo consente di ottenere un flusso di bit che si avvicina ad essere casuale; si sta approssimando OTP con 64 bit (DES) e 128 bit (AES).

Come operazione viene usato lo XOR perché non permette di trarre alcuna informazione osservando il cifrato.

Capitolo 4

Crittografia Asimmetrica

Una funzione botola è una funzione facile da computare in una direzione, ma difficile da computare in direzione opposta. RSA è un esempio di implementazione di funzione di questo tipo.

4.1 RSA

È un algoritmo di crittografia asimmetrico utilizzabile per criptare, decriptare, effettuare autenticazione tramite firma digitale e scambiare chiavi da usare in seguito per sistemi a cifratura simmetrica.

È basato sull'elevata complessità computazionale della fattorizzazione in numeri primi, che ne garantisce la sicurezza.

Generazione delle chiavi

La generazione delle chiavi per ciascun utente segue i seguenti passi:

1. scelti due primi p e q casuali da almeno 2048 bit
2. viene calcolato $n = pq$ e $\phi(n) = (p - 1)(q - 1)$
3. viene scelto un esponente pubblico e che sia coprimo con $\phi(n)$ (e minore di n)
4. viene calcolato l'esponente privato d tale che il prodotto con l'esponente pubblico sia congruo a 1 modulo $\phi(n)$
$$\rightarrow ed \equiv 1 \text{ mod } \phi(n)$$
 - chiave pubblica $k_u = (e, n)$
 - chiave privata $k_p = (d, p, q)$

Per la generazione dei parametri:

- per trovare p e q vengono generati dei numeri dispari casuali, e poi viene fatto un test di primalità
- per trovare l'esponente privato si utilizza l'algoritmo di Euclide esteso

Cifratura e decifratura

- **Cifratura:** $c = m^e \bmod n$
- **Decifratura:** $m = c^d \bmod n$

Calcolo dell'esponente

L'operazione di calcolo della potenza modulare $x^y \bmod z$ può essere implementata in diversi modi:

- *naive*: si moltiplica x per sé stesso per y volte facendo il modulo ad ogni passo, per evitare cifre grandi; poco efficiente siccome la complessità è asintotica a y
- *left to right*: siano $y_0 y_1 \dots y_n$ i bit che formano y
 - definiamo $y = y_0 + 2(y_1 + 2(y_2 \dots + 2(y_{n-1} + 2y_n)))$
 - da cui $\Rightarrow x^y = x^{y_0} + 2(x^{y_1} + 2(x^{y_2} \dots + 2(x^{y_{n-1}} + 2 \cdot x^{y_n})))$
- *right to left*: siano $y_0 y_1 \dots y_n$ i bit che formano y
 - definiamo $y = y_0 \cdot 2^0 + y_1 \cdot 2^1 + \dots + y_n \cdot 2^n$
 - da cui $\Rightarrow x^y = x^{y_0 \cdot 2^0} + x^{y_1 \cdot 2^1} + \dots + x^{y_n \cdot 2^n}$

Con gli ultimi due metodi la complessità passa da essere esponenziale a lineare rispetto al numero dei bit di y .

4.2 Crittosistema di El-Gamal

4.2.1 Concetto di logaritmo discreto

- Definiamo un gruppo finito G con operazione \oplus
- Definiamo H il sottogruppo generato da a , ovvero a^i con $i < 0$
- Definiamo $a \in G$ e $b \in H$
- L'unico intero x tale che $a^x = a \oplus a \oplus \dots \oplus a = b$ viene chiamato logaritmo discreto di b in base a

Possiamo definire anche come $a^x = b \bmod N$

\Rightarrow se N è primo, diventa difficile invertire il calcolo

4.2.2 Generatori

- definiamo $Z_p = \{0, \dots, p-1\}$
- definiamo Z_p^* i numeri di Z_p che sono coprimi con p
- g è un generatore di Z_p^* se elevandolo a tutte gli elementi di Z_p si ottengono tutti gli elementi che compongono Z_p stesso

$g = 2$ è un
generatore
di Z_{11}^*

$$\left\{ \begin{array}{ll} 2^{10} = 1024 = 1 \bmod 11 \\ 2^1 & = 2 \bmod 11 \\ 2^8 = 256 & = 3 \bmod 11 \\ 2^2 & = 4 \bmod 11 \\ 2^4 = 16 & = 5 \bmod 11 \\ 2^9 = 512 & = 6 \bmod 11 \\ 2^7 = 128 & = 7 \bmod 11 \\ 2^3 & = 8 \bmod 11 \\ 2^6 = 64 & = 9 \bmod 11 \\ 2^5 = 32 & = 10 \bmod 11 \end{array} \right.$$

4.2.3 Generazione delle chiavi

Il crittosistema di El-Gamal si basa sul concetto di logaritmo discreto; il processo per la generazione delle chiavi è il seguente:

- ogni utente genera un numero primo p molto grande e calcola:
 - un generatore g di Z_p^*
 - un α casuale compreso tra 1 e $p-1$
- la chiave pubblica è (p, g, β) , con $\beta = g^\alpha \bmod p$
- la chiave privata è α

4.2.4 Cifratura

- codificare il messaggio come un intero m in Z_p
 - scegliere un intero k compreso tra 1 e $p - 1$
 - calcolare $y_1 = g^k \bmod p$
 - calcolare $y_2 = m\beta \bmod p$
- \Rightarrow il messaggio cifrato corrisponde alla coppia (y_1, y_2)

4.2.5 Decifratura

- calcolare $z = y_1^\alpha \bmod p$
- calcolare $m = z^{-1} \cdot y_2 \bmod p$, con z^{-1} inverso modulare di z

4.3 Crittosistemi su curve ellittiche

La sicurezza dei sistemi crittografici a chiave pubblica dipende dal problema matematico su cui si basano; attualmente sono considerati sicuri sistemi basati su:

- fattorizzazione dei numeri primi
- logaritmi discreti
- curve ellittiche

Nel caso delle curve ellittiche, le operazioni basate su aritmetica modulare sono sostituite da operazioni su curve ellittiche; l'operazione di addizione forma un gruppo, e tale gruppo viene usato per la costruzione di un crittosistema. Questi sistemi permettono di avere la stessa complessità ma con una lunghezza della chiave minore.

El-Gamal applicato a curve ellittiche

Data una curva $E_p(a, b)$ un generatore G , e un punto sulla curva z , il problema è calcolare l'intero x tale che $x \cdot G = z$

Capitolo 5

Funzioni di hash

Una funzione di hash prende un messaggio di lunghezza arbitraria e restituisce una stringa unica di lunghezza fissata; l'idea è che il valore di hash è una rappresentazione non ambigua e non falsificabile di un messaggio.

Una funzione di hash comprime i messaggi e deve essere facile da computare. Vengono usate per:

- firme digitali
- integrità dei dati
- certificazione del tempo

Proprietà di sicurezza

- **One-Way:** dato y è computazionalmente difficile trovare M tale che $y = h(M)$
- **Sicurezza debole:** dato M è computazionalmente difficile trovare un altro M' tale che $h(M) = h(M')$
- **Sicurezza forte (collision resistance):** è computazionalmente difficile trovare due messaggi con lo stesso valore di hash

Attacco del compleanno

Questo attacco prende il nome dal *paradosso del compleanno*, appartenente alla teoria del calcolo probabilistico.

La probabilità che almeno due persone in un gruppo compiano gli anni lo stesso giorno è largamente superiore a quanto potrebbe dire l'intuito: infatti già in un gruppo di 23 persone la probabilità è circa 0,51 (51%); con 30 persone essa supera 0,70 (70%), con 50 persone tocca addirittura 0,97 (97%), anche se per arrivare all'evento certo occorre considerare un gruppo di almeno 366 persone

Lo scopo dell'attacco è, data una funzione di hashing h , trovare due valori x_1 e x_2 tali che $h(x_1) = h(x_2)$, ovvero una collisione.

È dimostrato che, in media, il numero di messaggi da genere è $2^{\frac{n}{2}}$, dove n è la lunghezza in bit del *digest* di h .

\Rightarrow dato che la soglia di sicurezza è di 2^{80} , la dimensione minima di un digest dovrebbe essere di almeno 160 bit.

Questo attacco viene usato per alterare le **firme digitali**, dato che lavorano applicando una cifratura sui *digest* delle funzioni di hash:

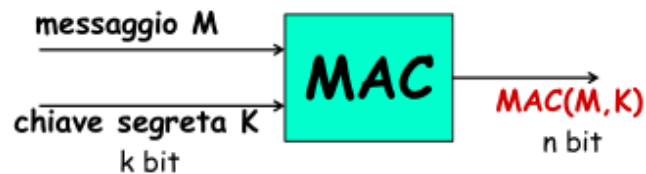
- prendo delle copie del documento originale, e faccio piccole modifiche insignificanti
- lo stesso viene fatto per il documento fraudolento, fino a trovare una corrispondenza nei valori di hash (collisione)

\Rightarrow viene fatto firmare il documento originale, allegando però quello fraudolento (che passerà per vero dato che hanno lo stesso digest)

Capitolo 6

MAC

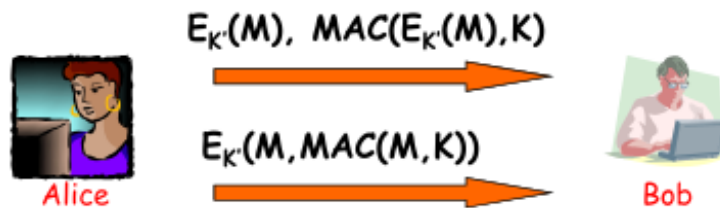
Con *Message Authentication Code* si intende un blocco di dati che preso in input un messaggio ed una chiave segreta, restituisce un codice MAC; il destinatario, alla ricezione del messaggio e del MAC, farà la stessa procedura per vedere se i codici MAC coincidono.



Viene usato per due scopi:

- autenticità del messaggio
- integrità del messaggio

MAC fornisce solo autenticità e integrità; se si vuole anche **condifenzialità**, si può cifrare prima o dopo con un'altra chiave (diversa da quella del MAC).



Proprietà

- **Easy computation:** dato un messaggio M e la chiave k , $MAC(M, k)$ è facile da calcolare
- **Compression:** output di lunghezza fissata
- **Computation-resistance:** data una o più coppie $(m, f_k(m))$ non deve essere possibile calcolarne altre; questa proprietà implica la *key-non-recovery*, ma non è vero il viceversa
- **Key-non-recovery:** data una o più coppie $(m, f_k(m))$ deve essere impossibile ricavare la chiave k

Attacchi

Si possono distinguere in base a:

- Scopo dell'attacco:
 - *Total break:* determinare la chiave
 - *Selective forgery:* dato m , determinare y tale che $y = MAC(m, k)$
 - *Existential forgery:* determinare una coppia (y, m) tale che $y = MAC(m, k)$
- Tipo di attacco:
 - *Known message* (conosce una lista di messaggi e relativi MAC)
 - *Chosen message* (sceglie i messaggi e può calcolare i relativi MAC)
 - *Adaptive chosen message* (come il precedente ma le scelte dipendono dalle risposte precedenti)

È ovviamente possibile fare anche attacchi di forza bruta, motivo per cui è consigliabile avere una chiave di almeno 128 bit.

6.1 HMAC

È una funzione che permette di calcolare un MAC tramite una funzione di hash qualsiasi e una chiave segreta; l'indipendenza dalla funzione di hash precisa è dovuta alla necessità di doverla sostituire nel caso in cui non sia più sicura.

- Data una funzione di hash h , un messaggio m , e una chiave k , hmac opera utilizzando blocchi di b byte
 - nel caso la chiave sia più lunga di b byte, le si applica la funzione di hash in modo da ottenerla lunga b byte
 - nel caso la chiave sia più corta, si aggiunge del padding
- vengono definite due stringhe:
 - *ipad* : byte 0x36 ripetuto $\frac{b}{8}$ volte
 - *opad* : byte 0x5c ripetuto $\frac{b}{8}$ volte
- la funzione prende m e k producendo il MAC secondo la formula

$$h(k \oplus opad \parallel h(k \oplus ipad \parallel m))$$

In generale, HMAC è sicuro se la funzione di hash usata è sicura.

Capitolo 7

Digital Signature Standard

DSS è un algoritmo di firma digitale basato sull'algoritmo di cifratura asimmetrica DSA (basato sul logaritmo discreto); il suo funzionamento è:

- scelta di un numero primo q da 160 bit
- scelta di un numero primo p di 1024 bit che sia multiplo di 64; inoltre, deve vale $p = qz + 1$ per un qualsiasi intero z
- viene scelto un h casuale tale che $1 < h < p - 1$
- $\alpha = h^z \bmod p$
- si genera la chiave privata s tale che $0 < s < q$
- si calcola la chiave pubblica $\beta = \alpha^z \bmod p$

I parametri p, q, α sono pubblici; il procedimento per il **calcolo della firma** è:

- generazione di un numero casuale r compreso tra 0 e q
- calcolo $y = (\alpha^r \bmod p) \bmod q$
- calcolo $\delta = (r^{-1} \cdot (h(m) + s \cdot y)) \bmod q$, con:
 - $h(m)$ funzione di hash applicata al messaggio
 - r^{-1} inverso modulare di r
- la firma è data dalla coppia (y, δ)

Per la verifica della firma:

- si calcola $e'' = y\delta^{-1} \bmod q$
- si calcola $e' = (h(m) \cdot \delta^{-1}) \bmod q$
- si calcola $v = ((\alpha^{e'} \cdot \beta^{e''}) \bmod p) \bmod q$
- la firma è verificata se $v = y$

Capitolo 8

Certificati

Un certificato è un documento elettronico che attesta **l'associazione univoca tra una chiave pubblica e l'identità di un soggetto** (persona, compagnia, computer ...), il quale dichiara di usarla in procedura di cifratura asimmetrica e/o autenticazione tramite firma digitale.

Il certificato contiene informazioni sulla chiave, sull'identità del proprietario e la firma digitale di un'autorità emittente che ha verificato il contenuto del certificato. I certificati si rivelano utili quando la chiave pubblica viene usata su larga scala.

Proprietà del certificato:

- chiunque deve poter leggere e determinare nome e chiave pubblica
- chiunque deve poter verificare l'autenticità
- solo l'autorità di certificazione (CA) può creare e aggiornare i certificati

Un certificato può contenere altre informazioni, come ad esempio:

- periodo di validità
- id della chiave
- stato della chiave
- informazioni addizionale su chiave e utente

Dato che i certificati possono essere revocati, si deve verificare questa evenienza consultando una lista dei certificati revocati (CRL), che deve essere aggiornata dalla CA. Una revoca può essere richiesta per diversi motivi:

- compromissione della chiave privata
- informazioni non più valide
- compromissione dell'algoritmo usato

,

Capitolo 9

Secret Sharing

Con il termine *secret sharing* si intendono i metodi per **distribuire un segreto tra un gruppo di partecipanti**, a ciascuno dei quali viene assegnato una parte di tale segreto; il segreto può essere ricomposto solo quando un numero sufficiente di parti viene combinato insieme.

In uno schema di condivisione segreta c'è un *dealer* e n *player*; il dealer fornisce una parte del segreto ai player, e solo quando determinate condizioni sono soddisfatte i player sono in grado di ricostruire il segreto.

9.1 Schema (n, n)

Sono richiesti n player su n partecipanti per ricostruire il segreto (quindi tutti); il processo di distribuzione funziona le modo seguente:

- viene scelto un numero primo p
- si sceglie un numero segreto s con $s < p$
- si scelgono casualmente $n - 1$ valori minori di p , ciascuno dei quale sarà uno share a_1, a_2, \dots, a_{n-1}
- lo share a_n viene calcolato come $a_n = (s - a_1 - a_2 - \dots - a_{n-1}) \bmod p$
- viene distribuito a ciascuno utente uno share a_i ; segue che per ricostituire s servono tutti gli share

9.2 Schema (k, n)

In questo caso il segreto viene diviso in n parti, ma ne bastano k per ricostruirlo; il funzionamento è il seguente:

- bisogna scegliere un primo p e un segreto s tale che $s < p$

- si generano $k - 1$ valori a_1, \dots, a_{k-1} che compongono lo share
- l' i -esimo share si ottiene come:

$$y_i = (s + a_1 \cdot i + a_2 \cdot i^2 + \dots + a_{k-1} \cdot i^{k-1}) \bmod p$$

con i che assume valori da 1 a n , corrisponde al numero dello share

La ricostruzione del segreto avviene risolvendo un sistema di k equazioni in k incognite (s e $k - 1$ valori di a).