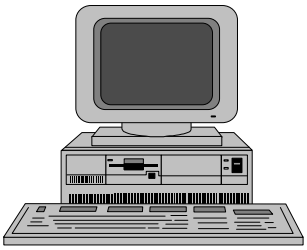
Insegnamento di

Sistemi ed Algoritmi per la Protezione dei Dati 1

(*Corso di Laurea Specialistica in Ingegneria Elettronica*)



Introduzione alla crittologia
Algoritmi crittografici

Rodolfo Zunino

Edizione 2004

Scuola di Elettronica
DIBE – Università degli Studi di Genova

Indice

1. Introduzione pag. 4

1.1 La crittologia pag. 4

1.2 Il sistema crittografico pag. 6

2. Elementi di crittologia pag. 9

2.1 Crittografia pre-computer pag. 9

2.2 Tecniche elementari di crittografia pag. 9

2.2.1 Sostituzione pag. 9

2.2.2 Cifrari polialfabetici pag. 10

2.2.3 Trasposizione pag. 12

2.2.4 La macchina Enigma pag. 13

2.3 Crittografia post-computer pag. 13

2.3.1 Perfetta segretezza e perfetta autenticità pag. 13

2.3.2 Principi di confusione e di diffusione pag. 18

2.4 Cifratura a blocchi e cifratura a stringa pag. 18

3. 1 sistemi crittografici a chiave segreta pag. 21

3.1 Struttura e principi generali pag. 21

3.2 Le problematiche aperte dei sistemi a chiave segreta pag. 23

3.3 Il crittosistema a chiave segreta DES pag. 24

3.3.1 L'algoritmo pag. 26

3.3.2 Caratteristiche degli S-boxes pag. 28

3.3.3 Dimensione della chiave segreta e sicurezza dei sistema pag. 29

4. 1 sistemi crittografici a chiave pubblica pag. 31

4.1 Introduzione e schema di principio pag. 31

4.2 La matematica per la crittografia pag. 32

4.2.1 L'aritmetica modulo n pag. 32

4.2.2 Elementi di teoria dei numeri pag. 35

4.2.2.1 La Euler totient function pag. 35

4.2.2.2 Proprietà fondamentali pag. 35

4.2.2.3 Il teorema di Fermat pag. 36

4.2.3 La ricerca di numeri primi "grandi" random pag. 37

4.3 I sistemi a distribuzione di chiave pubblica pag. 37

4.4 Crittosistemi a chiave pubblica pag. 38

4.4.1 Segretezza e problema della distribuzione della chiave pag. 39

4.5 Il crittosistema a chiave pubblica RSA pag. 41

4.5.1 Le funzioni unidirezionali e le funzioni unidirezionali pag. 41

4.5.2 L'algoritmo pag. 42

4.5.2.1 Un metodo per cifrare e decifrare efficientemente pag. 44

4.5.3 RSA: sicurezza e applicazioni pag. 45

5. I cifrari a stringa pag. 47

5.1 Struttura e principi generali pag. 47

5.2 Sequenze pseudorandom e generatori pseudorandom di bits pag. 48

5.3 Tecniche per la generazione di sequenze pseudorandom pag. 49

5.3.1 Linear congruence generators (LCGs) pag. 49

5.3.2 Feedback shift-registers (FSRs) pag. 49

5.3.2.1 Non Linear Feedback shift-registers (NLFSRs) pag. 50

5.3.2.2 Linear Feedback shift-registers (LIFSRs) pag. 51

Bibliografia pag. 53

1
Introduzione

1.1 La crittologia

La **crittologia,** come indica il nome stesso che significa "parola-nascosta", è l'area di ricerca che si occupa delle comunicazioni segrete. Essa pone la propria attenzione allo studio di tecniche che possano garantire lo scambio di informazioni riservate in maniera **sicura** su un canale di comunicazione pubblico, in modo tale che il messaggio inviato da un dato mittente possa raggiungere il destinatario designato senza correre il pericolo di essere intercettato e/o alterato da quella che si potrebbe definire "una terza parte incomoda".

Lo sviluppo della crittologia ha ricevuto un forte impulso soprattutto con l'evoluzione delle forme di comunicazione. Da un lato, l'avvento della comunicazione elettronica via computer, che ha rimpiazzato in diversi ambiti la comunicazione cartacea, ha reso necessari l'individuazione e l'impiego di nuove tecniche che garantissero protezione a dati "preziosi" o segreti, in sostituzione dei vecchi sistemi (casseforti, corrieri armati, veicoli blindati) rivelatisi inadeguati sia perché inapplicabili sia perché antieconomici; dall'altro, l'utilizzo di tale forma di comunicazione da parte di settori che fino ad oggi avevano seguito altre vie, ha fatto crescere la concentrazione delle energie sulla ricerca di soluzioni sempre più affidabili.

Un tempo la crittologia era oggetto d'interesse principalmente per le comunicazioni militari e diplomatiche, mentre oggi grandi quantità di dati che richiedono sicurezza e riservatezza, quali invio di numeri telefonici, numeri di carte di credito, ordini di acquisto (essere certi che l'ordine arriva da una certa persona e che la quantità di beni acquistati sia veramente quella indicata), transazioni finanziare, registrazioni legali, ecc. viaggiano su canali pubblici (e quindi alla portata di tutti) tramite posta elettronica (e-mail) e necessitano dunque dì essere protetti.

La crittologia si può suddividere in due settori: i protocolli e la crittografia.

I **protocolli** riguardano i "comportamenti" ossia il modo di usare la crittografia, è una serie di passi che coinvolge due o più parti e che serve a portare a termine un compito; ad esempio un attacco al protocollo può essere quello di corrompere la persona che è a conoscenza delle parola chiave necessaria all'apertura del sistema.

La **crittografia** è volta a trovare metodi che possano operare trasformazioni dei dati, tali da renderne impossibile l'utilizzo da parte di un "nemico" che ne sia entrato in possesso indebitamente. Colui che si occupa di studiare tali metodi è il **crittografo** egli opera su di un **messaggio originale o testo in chiaro M (plaintext)** trasformandolo in un **testo cifrato o crittogramma C (ciphertext or cryptogram).** L'intero processo di trasformazione dal messaggio originale al crittogramma prende il nome di **cifratura (enciphering process)** (Fig.1.1), il quale è governato da un parametro segreto che ne garantisce la sicurezza e che lo caratterizza e che viene chiamato chiave (key). A sua volta la chiave potrà essere privata (chiave simmetrica), quindi la trasmissione dovrà avvenire su un canale sicuro, oppure pubblica (chiave asimmetrica) dove la trasmissione può avvenire su un canale insicuro.

Una volta individuata la chiave di un sistema di cifratura è possibile facilmente risalire all'informazione originale. Vedremo più avanti che la chiave ha un'importanza fondamentale e ad essa sono strettamente connessi alcuni problemi fondamentali della crittografia.

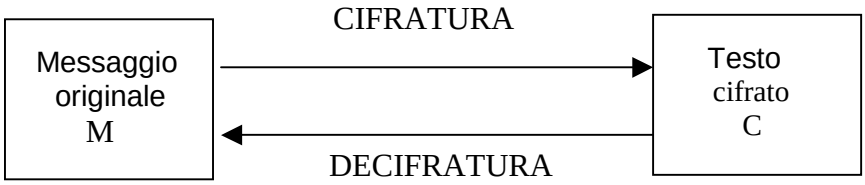


Fig.1.1: Schematizzazione delle operazioni crittologiche

ESEMPIO (codice di Cesare):

Supponiamo di avere a disposizione due alfabeti

1) (A,B,C,D,E….U,V,Z) alfabeto tradizionale;

2) (C,D,E,F,G, ... Z,A,B) alfabeto ottenuto da quello precedente sostituendo a ciascuna lettera in posizione i-sima quella corrispondente in posizione i+2-sima.
Dato un messaggio originale scritto con l'alfabeto 1):
vado a scuola
è possibile ottenere il corrispondente crittogramma
acfq c uezqnc
semplicemente riscrivendo il messaggio con l'alfabeto 2).

In questo caso la chiave è rappresentata dall'ampiezza dello scalamento dell'alfabeto 2) rispetto all'alfabeto 1); infatti, se avessimo scelto di sostituire ad ogni lettera in posizione i-sima quella corrispondente in posizione i+4-sima, anziché i+2-sima, il meccanismo per ottenere il crittogramma sarebbe rimasto invariato (semplice sostituzione di lettere), ma il risultato sarebbe stato sicuramente diverso. La chiave può quindi assumere 26 possibili valori (da 1 a 26) tanti quante le lettere dell'alfabeto.

La **crittanalisi** lavora in maniera diametralmente opposta alla crittografia. Il **crittanalista** è colui che intercettando di nascosto, sul canale di comunicazione, il crittogramma inviato da un mittente ad un destinatario, cerca di forzare il sistema di cifratura per risalire al messaggio originale o poterlo modificare prima che giunga a destinazione. Le strategie che il crittanalista utilizza sono diverse, ferma restando, comunque, un'assunzione nota come **assunzione di Kerchoffs** secondo la quale **il crittanalista nemico conosce per intero il meccanismo di cifratura, eccezion fatta per il valore della chiave segreta, quindi la sicurezza di un sistema informativo deve essere data solamente da essa.**

In generale si distinguono i seguenti tipi di approcci che il crittanalista può utilizzare:

Ciphertext only attack: è un attacco crittanalitico in cui il crittanalista ha a sua disposizione esclusivamente il testo cifrato. In tal caso il crittanalista si affida alla conoscenza che ha del sistema generale ed eventualmente ad informazioni sul messaggio del nemico, quali proprietà statistiche del linguaggio in uso (ad esempio quanto è ricorrente in percentuale una certa lettera, oppure se esistono parole che sono più probabili, come la parola "Caro" all'inizio di una missiva);

Known plaintext attack: è un attacco crittanalitico in cui il crittanalista possiede una sostanziosa quantità di messaggi originali e corrispondenti testi cifrati. In tal caso il crittanalista conosce a priori molti dettagli del messaggio di partenza;

Chosen plaintext attack: è un attacco crittanalitico nel quale il crittanalista ha la possibilità di proporre un'illimitata quantità di messaggi originali di sua scelta ed esaminare i corrispondenti crittogrammi che ne derivano. Talvolta, anziché proporre dei plaintext, il crittanalista può scegliere di proporre dei crittogrammi la cui risposta egli è in grado di decifrare servendosi dell'attuale chiave e in tal caso allora si parla di **chosen ciphertext attack.**

Tenendo presenti queste definizioni, appare dunque chiaro che, da un punto di vista ingegneristico, la crittanalisi può essere vista come un **problema d'identificazione del sistema.** Il Known plaintext attack e il chosen plaintext attack corrispondono rispettivamente al problema d'identificazione **passivo ed attivo.**

La crittografia e la crittanalisi non lavorano sempre in contrapposizione. La loro attività può essere coordinata allo scopo di ottenere sistemi crittografici sempre più potenti e sicuri; un crittanalista "amico" (anziché nemico, come lo abbiamo considerato fino a questo momento) può aiutare a rivelare punti di forza o insospettate debolezze di un cifrario appena progettato e fornire utili indicazioni per modificare il modello originario o ripensarlo "ex-novo".

Nella costruzione di sistemi crittografici si possono distinguere quattro approcci diversi; essi si differenziano nelle loro assunzioni circa le capacità e le opportunità del crittanalista, nella definizione di successo crittanalitico e nella nozione di sicurezza.
1. Information-theoretic approach.
Nell'information-theoretic approach si assume che il crittanalista abbia un illimitato tempo di calcolo e un'illimitata potenza di calcolo. La crittanalisi è intesa come il processo per determinare il messaggio, o la particolare chiave, avendo a disposizione solo il crittogramma e conoscendo le probabilità a priori dei vari messaggi e delle varie chiavi. Il sistema segreto è considerato violato quando esiste una soluzione "unica" per il crittogramma, cioè un messaggio con probabilità praticamente unitaria e tutti gli altri con probabilità praticamente nulla. Questo è l'approccio seguito da Shannon nella sua trattazione [ref 5]. L'assunzione dell'infinita potenza di calcolo del nemico implica che in questo tipo di modello la nozione di sicurezza è indipendente dalla complessità dei metodi di cifratura e decifratura. Il sistema è definito perfettamente sicuro se il testo in chiaro e il testo cifrato sono statisticamente indipendenti.

2. System-theoretic approach.
L'obiettivo del system-theoretic approach è assicurarsi che ogni nuovo crittosistema crei al crittanalista un nuovo, sconosciuto e difficile problema da risolvere. "Forzare" un sistema di questo tipo deve risultare un problema assai meno affascinante rispetto alla crittanalisi di un sistema basato su alcuni famosi problemi come la fattorizzazione o il logaritmo discreto; lo scopo del progettista è quello di fare in modo che nessuno dei principi fondamentali di crittanalisi (come la sostituzione, l'analisi statistica, ecc ...) sia applicabile al suo sistema. Senza dubbio, questa è la metodologia pratica di progettazione più largamente usata oggi.

3. Complexity-theoretic approach.
Nel complexity-theoretic approach tutti i calcoli sono parametrizzati da un parametro di sicurezza, generalmente la lunghezza della chiave, e viene effettuata un'analisi asintotica. Vengono considerati computazionalmente realizzabili soltanto quegli algoritmi che hanno un tempo di "running" polinomiale nella dimensione dell'input. L'obiettivo del progettista è quello di basare il sistema su di un problema computazionalmente irrealizzabile o, per lo meno, di renderlo equivalente ad esso. E' il caso, come vedremo, dell'algoritmo RSA (par.6.2) che si basa su alcuni "famosi" "problemi da risolvere quali il logaritmo discreto e l'inversione.

4. Randomized stream ciphers.
Il progettista può, anziché puntare ad assicurare che il processo crittanalitico richieda uno sforzo computazionale irrealizzabile, mirare a garantire che il problema crittanalitico abbia una dimensione irrealizzabile. L'obiettivo è accrescere il numero di dati che il crittanalista deve analizzare nel processo crittanalitico, pur mantenendo piccola la chiave segreta. Questo può essere fatto facendo uso di una grande stringa di dati casuali pubblicamente accessibile (da cui il nome di Randomized stream ciphers) nei processi di cifratura e decifratura. La chiave specifica quali parti del "grande randomizzatore" devono essere usate, mentre il nemico, non conoscendo tale chiave segreta, è costretto a fare una ricerca su tutti i dati casuali. La sicurezza di sistemi basati su "randomized stream cipher"può essere espressa attraverso il numero medio di dati che il crittanalista deve esaminare prima che le sue possibilità di determinare la chiave o il messaggio originale crescano al di là delle semplici previsioni.

1.2. Il sistema crittografico

Un sistema crittografico può essere rappresentato in generale come in fig. 1.2; esistono tre parti principali: un **mittente (sender), un ricevente (receiver) ed un crittanalista (cryptanalyst**, a volte indicato come **eavesdropper** cioè colui che origlia**).** Mittente e ricevente comunicano tra loro tramite un canale di comunicazione **insicuro** che è controllato dal crittanalista.

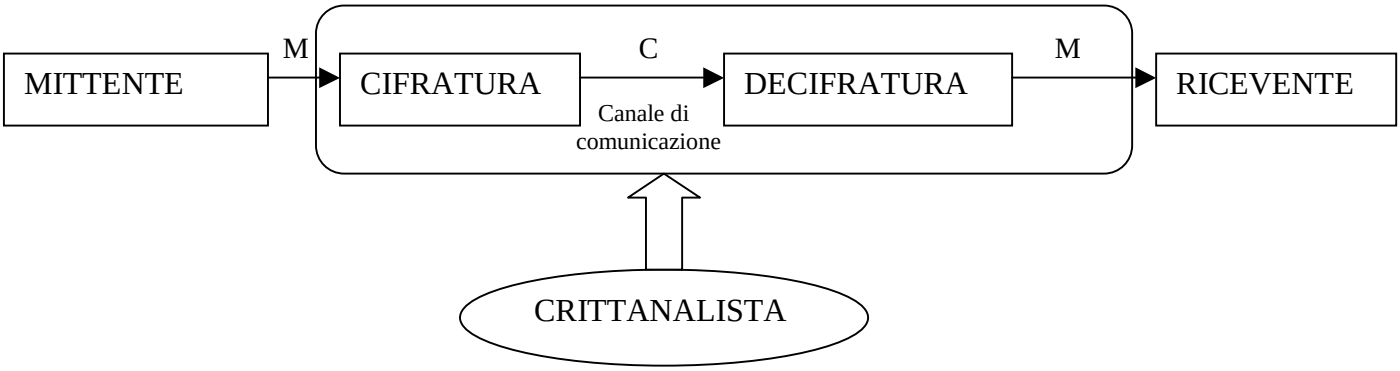


Fig. 1.2: **schema di principio di un sistema crittanalitico**

Il mittente genera il messaggio non cifrato M che deve essere comunicato al ricevente designato; prima di essere inviato sul canale, esso viene **cifrato** (blocco cifratura in figura) per prevenire che il crittanalista venga a conoscenza del suo contenuto e dà vita al crittogramma C, che costituisce la reale informazione che viaggia sul canale. Quando il legittimo ricevente ottiene C, lo **decifra** (blocco decifratura in figura) e recupera il messaggio originale.

Il canale di comunicazione **è insicuro** poiché a disposizione di chiunque voglia utilizzarlo; per questa sua caratteristica, esso viene identificato anche con il termine **canale pubblico.** In realtà è importante evidenziare che un canale è considerato pubblico se la sua sicurezza è inadeguata per le necessità dei suoi utilizzatori. Un canale come quello telefonico può, quindi, essere considerato privato da alcuni utilizzatori e pubblico da altri. Il canale, in base all'uso che ne viene fatto, può essere soggetto alla minaccia di **intercettazione (eavesdropping) o di inserimento (injection) illegittimi** d'informazione o di entrambe. Nella comunicazione telefonica, il rischio d'inserimento è massimo poiché la parte chiamata non può stabilire qual'è il telefono chiamante; l'intercettazione, invece, che richiede l'uso di un intercettatore di chiamate telefoniche, è tecnicamente molto difficoltosa e legalmente pericolosa.

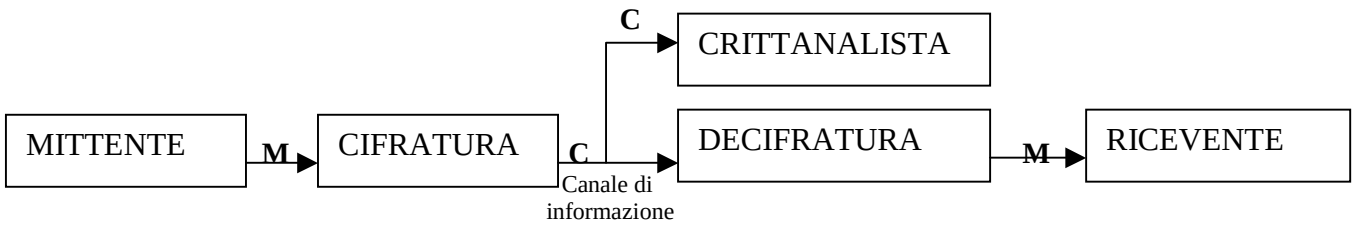


Fig. 1.3: flusso d'informazione in un privacy system

Al contrario, nella comunicazione via radio, l'intercettazione è passiva e non comporta nessun pericolo legale, mentre l'inserimento costringe il trasmettitore illegittimo a scoprirsi.

In figura 1.2, dove è rappresentato solo lo schema di massima di un sistema crittografico, viene indicata l'azione di un crittanalista sul sistema, ma volutamente non è precisato il punto in cui essa avviene. Tenendo presente, però, quanto è stato detto sopra circa l'eavesdropping e l'injection è possibile ora dare una descrizione più dettagliata. Le figg. 1.3 e 1.4 rappresentano due diverse possibili configurazioni di sistemi crittografici, in cui il crittanalista gioca un ruolo diverso.

La fig. 1.3 rappresenta un cosiddetto **privacy sistem;** si tratta di una classe di sistemi crittografici, che, come dice il termine stesso, sono rivolti a risolvere quello che in crittografia è noto come **problema della segretezza o privacy problem.** Essi sono progettati per prevenire l'estrazione di informazione, da parte del crittanalista, di messaggi inviati su un canale pubblico allo scopo di assicurare al mittente che essi sono stati letti **esclusivamente** dal destinatario designato. Il crittanalista, in tal caso, è pensato essere in grado soltanto di intercettare di nascosto il crittogramma C che viaggia sul canale, senza avere la possibilità (o la volontà) di modificarlo.

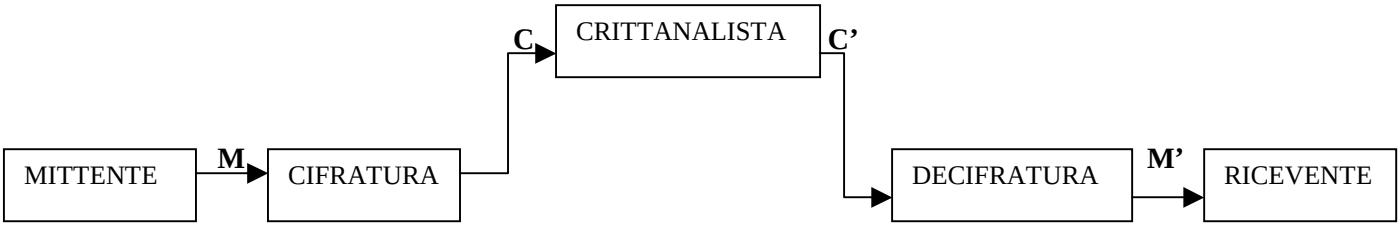


Fig 1.4: flusso d'informazione in un authentication system

La fig. 1.4, invece, rappresenta un **authentication system** volto a risolvere l'altro gran problema della crittografia: il **problema dell'autenticità' o authentication problem.** Questa classe di sistemi mira a prevenire un non autorizzato inserimento di messaggi sul canale o una modifica del messaggio originale da parte del crittanalista, garantendo al ricevitore di un messaggio la legittimità del suo mittente. In questo caso, infatti, il crittanalista ha la possibilità di prelevare dal canale il crittogramma C non solo per

ricavare da esso il messaggio originale M, ma anche per ottenere le informazioni sufficienti a cifrare a sua volta un inautentico messaggio M' da inviare come accettabile crittogramma C' al destinatario.

La distinzione tra questi due tipi di sistemi è importante perché, **in generale, un sistema crittografico progettato** per **garantire** la segretezza **può non assicurare l'autenticità'.** In realtà questo è vero per i sistemi a chiave segreta; per i sistemi a chiave pubblica, come vedremo, la situazione è un po' diversa poiché la loro struttura è in grado di fornire allo stesso tempo entrambe queste caratteristiche.

Un ultimo elemento fondamentale di un sistema crittografico è la **chiave segreta.** Come già evidenziato, la cifratura e la decifratura sono rese possibili da una quantità segreta detta appunto chiave di cui è in possesso sia mittente sia ricevente; le modalità con cui questa chiave viene scambiata tra mittente e ricevente ha generato due diversi tipi di sistemi crittografici, i cosiddetti **sistemi a chiave segreta,** in cui le chiavi usate da sender e receiver sono uguali, **e i sistemi a chiave pubblica**, in cui le chiavi usate da sender e receiver sono diverse.

Tali sistemi hanno caratteristiche e prestazioni differenti che analizzeremo più da vicino nei prossimi capitoli. Ciò che è importante rilevare qui, è che **la chiave gioca un ruolo centrale in un sistema crittografico perché da essa dipende la sicurezza del sistema.** Una volta scoperta la chiave, il meccanismo di cifratura può essere facilmente identificato e di conseguenza viene vanificato il suo scopo.

Per meglio comprendere questo concetto, si può paragonare il sistema crittografico ad una cassaforte con combinazione resettabile. La struttura della cassaforte è a disposizione di chiunque desideri aprirla, ma la combinazione è tenuta segreta; anche se uno conosce le operazioni da fare, ma non sa come "combinarle" non riesce a violare il sistema. Inoltre la combinazione può essere cambiata nel caso si sospetti che sia finita in possesso di qualcuno non autorizzato ad averla; analogamente, in un sistema crittografico, la cifratura (o la decifratura) sono costituiti da un set di istruzioni, un pezzo di hardware o un programma di computer che è capace di cifrare il messaggio originale (o decifrare il testo cifrato) in modi diversi, uno dei quali è scelto dalla chiave specifica.

Ci sono due modi fondamentalmente diversi in cui un sistema può essere sicuro.
In alcuni sistemi, la quantità di informazione a disposizione del crittanalista è insufficiente a determinare le trasformazioni di cifratura e decifratura, indipendentemente dal tempo illimitato e dalla potenza di calcolo che il crittanalista può utilizzare. Un sistema di questo tipo si dice **incondizionatamente sicuro.** La sicurezza incondizionata deriva dall'esistenza di soluzioni multiple, tutte significative, per il crittogramma.

In altri casi, invece, il materiale intercettato contiene sufficiente informazione per fornire un'unica soluzione al problema crittanalitico, ma non c'e'nessuna garanzia che questa soluzione possa essere ottenuta da un crittanalista con limitate risorse computazionali. Sistemi di questo tipo, in cui la soluzione sarebbe ottenibile, ma con costi computazionali (sia in termini di memoria usata sia in termini di runtime) troppo elevati per essere realizzabili, si dicono **computazionalmente sicuri.**

2
Elementi di crittologia

2.1 Crittografia pre-computer:
Nei prossimi paragrafi verranno trattate le tecniche di crittografia utilizzate prima dell'avvento del computer, quindi fino alla seconda guerra mondiale. Si è scelto di fare questa differenziazione perché il tipo di impostazione di tali tecniche e la velocità nell'evoluzione e nella creazione di nuove e sofisticate è nettamente diversa, infatti la grande potenza di calcolo dei computer ha mutato radicalmente il modo di procedere nel mondo della crittologia.

2.2 Tecniche elementari di crittografia

Esistono diversi tipi di trasformazioni che i sistemi crittografici possono applicare al messaggio originale per ottenere il crittogramma. La varietà e la complessità di tali trasformazioni sono andate via via evolvendosi nel tempo, di pari passo con la necessità di ottenere sistemi sempre più affidabili e sicuri e con l'esigenza di soddisfare bisogni diversi dettati dal tipo di applicazione cui erano destinate.

D'altra parte, i criteri in base ai quali è possibile dare una valutazione di un sistema crittografico sono davvero molti e tengono conto, per esempio, del grado di segretezza, della dimensione della chiave, dell'espansione del messaggio e della propagazione degli errori. Risulterebbe, quindi, lungo e, per certi aspetti, anche inutile fare una classificazione precisa e dettagliata.

E' possibile, tuttavia, individuare alcune tecniche di crittografia che sono molto semplici e che fanno uso di metodi elementari facilmente applicabili. La loro semplicità non va scambiata, però, come sinonimo di inaffidabilità, poiché i principi che ne stanno alla base sono molto efficaci e, opportunamente adattati, hanno dimostrato di ottenere ottimi risultati in diverse situazioni.

Queste tecniche sono indicate come tecniche di base e vengono impiegate o autonomamente così come sono o variamente combinate come fondamento di procedimenti crittografici più complessi. Qui di seguito le analizziamo in dettaglio e supponiamo, per la semplicità e per la chiarezza della spiegazione, che il messaggio originale cui vanno applicate sia rappresentato da un testo letterario.

2.2.1 Sostituzione

E' la più semplice tecnica di cifratura, consiste nel sostituire a ciascuna lettera del testo una lettera, una cifra, o un segno convenzionale che le corrisponde regolarmente. Il procedimento più semplice (quello del cifrario di Cesare) consiste nello scalare di una quantità costante le lettere dell'alfabeto: ad esempio, se tale quantità è fissata pari a tre, si ha che A diviene D, B diviene E, C diviene F, ecc... Siccome, però, la chiave di tale cifratura (rappresentata dalla quantità costante di cui si trasla) è facile da scoprire, generalmente si preferisce utilizzare come chiave un alfabeto incoerente o permutato (ad esempio B WEKQFMVYALUCONP1-ISIDXTRGZJ) le cui lettere vengono sostituite a quelle del normale alfabeto, questo metodo è il cifrario di sostituzione monoalfabetico.

Utilizzando l'alfabeto indicato: A è rimpiazzata da B, B è rimpiazzata da W, C è rimpiazzata da E, ecc. Inoltre, se le divisioni tra parole vengono preservate, un messaggio come

OGNI GIORNO HA IL SUO MATTINO

viene trasformato in

NMOY MYNSON VB YU IXN CBDDYON

Cambiando l'alfabeto permutato si ottengono crittogrammi diversi per lo stesso messaggio originale (si veda esempio par.1.1). Cifrari a semplice sostituzione su alfabeti piccoli, come l'alfabeto romano o l'ASCII, offrono piccola protezione per i relativi messaggi di testo in chiaro. 1 crittogrammi ottenuti dalla sostituzione, infatti, possono essere risolti facendo delle tabelle di frequenza delle lettere, di coppie di lettere (digrammi) e di triple di lettere (trigrammi).

In queste tabelle, risultano evidenti le frequenze alte di lettere come E e T, oppure le frequenze basse di Q e Z, o ancora le frequenti associazioni di vocali con consonanti; esse forniscono, dunque, un valido strumento per identificare le corrispondenze tra le lettere del testo in chiaro e del testo cifrato.

D'altro canto, risultati analoghi si possono ottenere passando a setaccio il testo cifrato per evidenziare delle cosiddette parole campione (pattern words) che si pensa possano essere presenti nel testo in chiaro. Ad esempio, in testi scritti in lingua inglese, parole frequenti quali "EVERY", "THAT", e "LOOK" possono essere evidenziate a causa delle loro lettere ripetute. Riassumendo, per rompere un cifrario monoalfabetico bisogna:

- sapere la lingua del messaggio
- avere un cipher-text lungo
- avere la tabella delle frequenze
- confrontare tale tabella con quella naturale

Facendo crescere la dimensione dell'alfabeto, queste tecniche diventano molto costose da applicare. Ad esempio, se anziché utilizzare il codice ASCII con i suoi 128 caratteri a 7 bit, si utilizzasse un set di parole tutte a 32 bit, la compilazione di una tabella di frequenza potrebbe richiedere uno sforzo eccessivo: essa sarebbe lunga 2^{32} parole, cioè circa quattro bilioni di parole, così come l'alfabeto permutato (cioè la chiave). 1 cifrari a sostituzione, dunque, su "alfabeti" grandi non permettono tutte le possibili permutazioni, allo scopo di limitare la chiave ad una più ragionevole dimensione. Il sistema crittografico DES, che vedremo nel par. 6. 1, rappresenta un buon esempio di tale filosofia.

2.2.2 Cifrari polialfabetici

Si tratta di cifrari a sostituzione nei quali, per cifrare il messaggio, vengono usati periodicamente parecchi differenti alfabeti allo scopo di contrastare i metodi usati con i cifrari monoalfabetici (basati sull'uso delle tabelle delle frequenze); adottando tale tecnica si riesce ad "attenuare" l'andamento degli istogrammi relativi alle analisi in frequenza. Per esempio, utilizzando due alfabeti si procede come segue:

A B C……Z
L'_1 L'_2 L'_3……L'_{26}
L''_1 L''_2 L''_3……L''_{26}

Ossia si mette in corrispondenza l'alfabeto di partenza con i due alfabeti ottenuti con la stessa tecnica dei cifrari monoalfabetici. Adesso, volendo criptare la parola "mattino" si ottiene:

M A T T I N O
L'_{13} L''_1 L'_{20} L''_{20} L'_9 L''_{14} L'_{15}

E’ immediato notare come un conteggio di frequenza sia per lettere singole che per coppie di lettere si rivela, in questo caso, molto più inefficace e produce scarsi risultati, usando più alfabeti l’efficacia del cifrario aumenta.

Proprio a partire da questa osservazione Vigenère ha inventato il cifrario omonimo, il quale non è altro che un’evoluzione del precedente. Vigenère ha pensato di usare il massimo numero possibile di alfabeti, ossia ventisei, tanti quanti le lettere dell’alfabeto, in modo da ridurre ad un andamento costante l’istogramma delle frequenze.
Il modo di procedere è il seguente:

1) crea una tabella dove è presente l’alfabeto di partenza ed i corrispondenti ventisei alfabeti, ottenuti traslando l’alfabeto di partenza di un posto, poi due, ecc. fino a ventisei:

	a	b	c……z
1	B	C	D……...A
2	C	D	E……B
.	.		.
.	.		.
.	.		.
26	A	B	C……Z

2) fissa una chiave e codifica ogni lettera del messaggio usando l’alfabeto che inizia per la lettera della chiave corrispondente; per esempio, scegliendo come chiave la parola “cane” e volendo codificare la frase “non vedo non sento non parlo”:

```
C A N E C A N E C A N E C A N E C A N E C A N
n o n v e d o n o n s e n t o n o n p a r l o
```

- codifica la prima lettera (la “n”) usando l’alfabeto che inizia per C e scegliendo la quattordicesima lettera di esso (perché n è appunto la quattordicesima lettera dell’alfabeto)
- La seconda lettera (la “o”) viene codificata usando l’alfabeto che inizia per A e prendendo la ventesima lettera
- e così via.

Quindi la lettera della chiave considerata corrisponde alla riga della tabella, cioè ad un determinato alfabeto, mentre il plain-text (o il messaggio in chiaro) rappresenta la colonna della tabella.

E' possibile, tuttavia, seguire un approccio diverso per la crittanalisi, noto come approccio di Kasiski (scoperto in verità 50 anni prima di Kasiski dall’inglese Babbage, e tenuto segreto per motivi strategici e politici). Esso consiste nel determinare il periodo, a partire dalla loro prima comparsa, di gruppi ripetuti di lettere di tre o più cifre del testo cifrato, che occorrono in corrispondenza di trigrammi frequenti del testo in chiaro, quali THE o ING per la lingua inglese, che si presentano due volte nella stessa frase.

Le ipotesi di base che devono essere rispettate per poter “rompere” il cifrario di Vigenère sono sempre le solite due, ossia che bisogna conoscere la lingua del messaggio ed il messaggio stesso deve essere sufficientemente lungo.
Si può notare come il linguaggio naturale sia ridondante e sfruttando tale caratteristica si può impostare un metodo per rompere il cifrario; osservando l’esempio precedente si vede che la parola “non”, è ripetuta tre volte e per la seconda e la terza c’è una periodicità: compaiono a multipli della lunghezza della chiave (compaiono all’ottavo ed al sedicesimo carattere mentre la parola chiave è lunga quattro):

```
C A N E C A N E C A N E C A N E C A N E C A N
n o n v e d o n o n s e n t o n o n p a r l o
              8                 16
```

Il crittanalista, allora, conta il numero di caratteri tra i gruppi ripetuti, ottenendo i periodi, ed il fattor comune su tutti i periodi dà la (presunta) lunghezza "n" della chiave. Adesso su ogni n-sima lettera di testo cifrato si effettua un calcolo di frequenza (attacco "monoalfabetico"), proprio perché ogni n lettere del cipher-text otterrò una codifica che usa un solo alfabeto. Per chiarire meglio, se si osserva l'esempio precedente si nota che ogni quattro lettere, cioè la lunghezza della chiave, la codifica viene effettuata usando l'alfabeto che comincia per "E", quindi prendendo solo le lettere a multipli di quattro mi ritrovo un cipher-text ottenuto da un cifrario monoalfabetico, di conseguenza farò un analisi delle frequenze.
Dovrò infine fare tanti attacchi monoalfabetici quanto è lunga la chiave (nell'esempio dovrò fare quattro attacchi).

L'algoritmo per rompere il cifrario di Vigeneire allora è:

1) Cercare ripetizioni nel cipher-text
2) Enumerare i periodi (distanza in lettere nel cipher text dalle ripetizioni)
3) Cercare il fattor comune (comune a tutti i periodi)
4) Riordinare il cipher-text secondo la lunghezza della chiave L_K
5) Fare L_K (lunghezza chiave) attacchi "monoalfabetici"

Esaminando il cifrario di Vigeneire si può notare che se la chiave fosse lunga quanto il messaggio (nell'esempio dovrebbe essere lunga 23) e fosse assolutamente casuale sarebbe un algoritmo di cifratura perfetto, ossia indistruttibile; comunque questo argomento si approfondirà successivamente.

2.2.3 Trasposizione

Nella trasposizione, viene cambiata la posizione delle lettere all'interno del testo in chiaro, piuttosto che venire cambiate le lettere dell'alfabeto. Ad esempio, se il messaggio del paragrafo precedente viene spezzato in gruppi di 5 caratteri (includendo gli spazi) e le lettere in ciascun gruppo vengono riordinate in accordo ad una permutazione: 1-2, 2-5, 3-1, 4-4, 5-3 (cioè il terzo carattere di ogni gruppo è scritto per primo, il primo carattere è scritto per secondo, ecc.) il crittogramma diventa

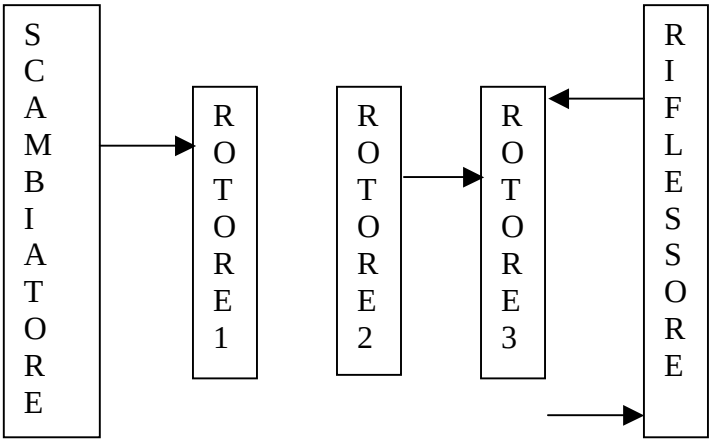
1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4
O	G	N	I	_	G	I	O	R	N	O	_	H	A	_	I	L	_	S	U	O	_	M	A	T	T	I	N	O
N	O	_	I	G	O	G	O	R	I	H	O	_	A	_	_	I	U	S	L	M	O	T	A	_	N	T	T	O
3	1	5	4	2	3	1	5	4	2	3	1	5	4	2	3	1	5	4	2	3	1	5	4	2	3	1	5	4

Nel caso della trasposizione, le tabelle di frequenza di coppie e triple di lettere rivelano la disgiunzione di comuni coppie di lettere, come ad esempio T e H della parola inglese THE, consentendo di ricostruire il testo in chiaro con quelle permutazioni che le ricongiungono.

In questo modo, è possibile recuperare dal crittogramma la chiave usata per trasformare il testo in chiaro e, quindi, il testo in chiaro stesso.

2.2.4 La macchina enigma

E' stata la prima macchina "automatica" di cifratura, infatti era una macchina da scrivere in cui si digitava il testo in chiaro e veniva visualizzato in codice. L'interno di tale macchina si può schematizzare così:



Lo scambiatore traspone da uno a dieci posti la lettera digitata (codifica per trasposizione), i rotori contengono ciascuno un alfabeto e l'uscita del precedente è l'ingresso del successivo; il riflessore serve per la visualizzazione del cipher-text.
Ogni rotore ha ventisei posizioni possibili (tante quante le lettere dell'alfabeto) e dopo un certo numero di tasti digitati uno di essi effettua una rotazione (ogni rotore esegue una codifica di sostituzione monoalfabetica).
Quindi la lunghezza totale della chiave sarà data da:
26^3 x N x 6
dove 26 sono le posizioni possibili di ciascun rotore, 3 è il numero di rotori, N è il contributo alla lunghezza della chiave dovuto allo scambiatore e 6 ?

Il punto debole di tale macchina sta nel fatto che i cifrari monoalfabetici sono fissi, infatti i fili che collegano i rotori tra loro sono fisicamente saldati nella macchina.

2.3 Crittografia post-computer:

2.3.1 Perfetta segretezza e perfetta autenticità

Shannon, nella sua teoria sulla sicurezza teorica e sulla sicurezza pratica dei sistemi crittografici [ref.5], ha definito la perfetta segretezza (perfect secrecy) di un sistema crittografico come la capacità del sistema di garantire che il messaggio originale M e il crittogramma C siano statisticamente indipendenti; questo significa, in termini pratici, che un crittanalista nemico ha le stesse possibilità di stimare il messaggio originale M, sia che conosca C sia che non lo conosca, poiché C non consente di ricavare alcuna informazione utile riguardo M.

Bisogna intanto definire la quantità di informazione che una sorgente può erogare, essa prende il nome di entropia ed è definita come il valor medio del logaritmo cambiato di segno della corrispondente distribuzione di probabilità; ad esempio, se il messaggio trasmesso può assumere soltanto due valori: "uomo" (U) o "donna" (D), l'entropia del messaggio M sarà:

$$H(M)= - [p(U)\log_2(U)+p(D)\log_2(D)]$$

Allora se U e D sono equiprobabili H(M)=1 altrimenti H(M)<1.

In generale, date due grandezze random X e Y la quantità $H(X/Y)=\Sigma x \Sigma y\ P(X=x, Y=y)\ [-\log P(X=x, Y=y)]$ rappresenta *l'incertezza circa X data la conoscenza di Y* e le sommatorie sono fatte sopra tutti i possibili valori di x e y.

Dall'entropia è possibile ricavare il "rate" di un linguaggio:

$$r = H(M)/L \leq 1 \quad \text{con L pari alla lunghezza del messaggio}$$

Per esempio nel caso della lingua inglese il rate corrisponde a 1.3 bit/byte.
Sempre a Shannon si deve la formula della ridondanza D definita come:

$$D = \log_2 A - r \quad \text{dove A è il numero di simboli dell'alfabeto considerato}$$

Quindi a seconda della lingua del messaggio e dell'alfabeto usato si avrà un valore di ridondanza diverso. Tale ridondanza è presente in ogni linguaggio naturale ed è anche grazie ad essa che un crittogramma può venire decifrato dal crittanalista (es: tabella delle frequenze dei simboli dell'alfabeto); per questo motivo bisogna cercare di ridurre la ridondanza dei simboli dell'alfabeto usato. Per eliminare la ridondanza, in modo da rendere equiprobabili i simboli dell'alfabeto si usa la codifica di Huffmann o codifica ottima; tale codifica associa pochi bit ai simboli più probabili e tanti bit a quelli più improbabili.
Teorema fondamentale della crittografia:
In un sistema crittografico sicuro la chiave deve essere casuale e lunga quanto il messaggio.

Nella sua teoria Shannon assume che ogni chiave k (e quindi ogni trasformazione) abbia associata una probabilità a priori $P_K(k)$ che rappresenta la probabilità di scegliere proprio quella chiave. Analogamente, assume che ogni messaggio M abbia associata una probabilità a priori $P_M(M)$ determinata da un sottostante processo stocastico. Queste probabilità per le varie chiavi e i vari messaggi rappresentano la conoscenza a priori che il crittanalista può avere della situazione.

Nel momento, però, in cui egli intercetta il crittogramma, può calcolare le probabilità a posteriori $P_{M/C}(M/C)$ dei vari messaggi e le probabilità a posteriori $P_{k/C}(k/C)$ delle diverse chiavi che possono avere prodotto quel crittogramma. L'insieme di tali probabilità a posteriori costituisce la conoscenza del crittanalista circa la chiave e il messaggio dopo l'intercettazione.

Alla luce di tutto ciò, la perfetta segretezza può essere definita, dunque, come la capacità del sistema crittografico di garantire che la probabilità a posteriori che un dato crittogramma rappresenti un dato messaggio sia identicamente uguale alla probabilità a priori di quel messaggio (questo è appunto lo scopo della crittografia, ossia che il cipher text C appaia come rumore). Formalmente, si deve avere quindi che:

$$P_{M/C}(M/C)=P_M(M) \qquad \text{per qualunque } M,C \qquad (3.4)$$

La (3.4) rappresenta proprio la formulazione matematica di quella *indipendenza statistica* tra il messaggio *M* e il crittogramma C cui si faceva accenno all'inizio del paragrafo. Shannon ha mostrato che è possibile ottenere la perfetta segretezza nei sistemi per i quali è valida la relazione

$$H(k) \geq H(M) \qquad (3.5)$$

la disuguaglianza (3.5) si può leggere in questo modo:

in un sistema si ha perfetta segretezza se l'incertezza della chiave segreta k è grande almeno quanto l'incertezza del messaggio originale M.

La dimostrazione è la seguente. Le equazioni (3.2) e (3.3) che definiscono le trasformazioni dei sistemi a chiave segreta, possono essere scritte, equìvalentemente, in termini di incertezza come:

$$H(C/M,k)=0 \qquad (3.6)$$

$$H(M/C,k)=0 \qquad (3.7)$$

rispettivamente, poiché per esempio *H(C/M,k)* è zero se e solo se M e *k* insieme determinano unicamente C. Analogamente la definizione (3.4) di Shannon di perfetta segretezza può essere riscritta come:

$$H(M/C)=H(M) \qquad (3.8)$$

poichè questa uguaglianza esiste se e solo se *M è* statisticamente indipendente da C e in tal caso la conoscenza di C non porta alcuna informazione. Per ogni sistema a chiave segreta si ha che:

$$\begin{aligned} H(M/C) &\leq H(M,k/C) \qquad (3.9)\\ &= H(k/C)+H(M/C,k)\\ &= H(k/C)\\ &= H(k). \end{aligned}$$

Vediamo di illustrare, ad uno ad uno, tutti i passaggi. La prima disuguaglianza delle (3.9) si può spiegare facendo riferimento ad una proprietà elementare dell'entropia che stabilisce che, date tre generiche variabili random x, y, z,

$$H(x,y) = H(x)+H(y/x) \qquad (3.10a)$$
$$= H(y)+H(x/y) \qquad (3.10b)$$

e similmente

$$H(x,y/z) = H(x/z)+H(y/x,z) \qquad (3.11a)$$
$$= H(y/z)+H(x/y,z) \qquad (3.11b)$$

Dalla (3.11a) si ricava facilmente che:

$H(x,y/z)=H(x,y/z)-H(y/x,z)\leq H(x,y/z)$

che altro non è che la riscrittura del primo passaggio delle (3.9). La (3.11b) è stata, invece, utilizzata nella seconda uguaglianza delle (3.9). Si può facilmente verificare, cambiando soltanto il nome delle variabili rappresentate.

Nella terza uguaglianza delle (3.9) è stata applicata la (3.7) ed, infine, nell'ultimo passaggio delle (3.9) si è tenuto conto del fatto che la rimozione di una data conoscenza (nel nostro caso la conoscenza di C) può soltanto accrescere l'incertezza.

Possiamo concludere, dunque, che *se il sistema garantisce la perfetta segretezza*, allora segue dalla (3.8) e dalla (3.9) che

$$H(k) \geq H(M)$$

che è proprio la condizione (3.5) che volevasi dimostrare.

Shannon ha dimostrato altresì che, sotto opportune ipotesi, è possibile formulare diversamente la condizione (3.5), legandola alla lunghezza della chiave segreta. Se le N cifre della chiave sono scelte da un alfabeto di dimensione L_k (cioè possono avere valori compresi tra 0 e L_{k-1} si potranno avere al più L_k^N valori diversi di chiave e, nel caso di una scelta completamente random, la probabilità di avere una data chiave sarà L_k^{-N} . In questo caso, si avrà che:

$$H(k) < \log(L_k^N)=N \log L_k \qquad (3.12)$$

con l'uguaglianza se e solo se la chiave è completamente random.

Analogamente, se P sono le cifre del testo in chiaro M, scelte da un alfabeto di dimensione L_M , si avrà che:

$$H(M) < P \log L_M \qquad (3.13)$$

con uguaglianza, anche in questo caso, se il testo in chiaro è completamente random.

Se $L_M=L_k$ (come nel *one-time pad)* e se il testo in chiaro è completamente random, la condizione di Shannon (3.5) per la perfetta segretezza si può riscrivere, tenendo conto della (3.12) e (3.13) (presa con il segno uguale), nella forma:

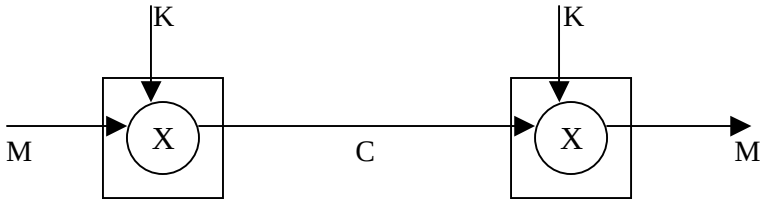
$$N \geq P \qquad (3.14)$$

dove N rappresenta la lunghezza della chiave e P quella del messaggio originale.
Si tratta di una formulazione sicuramente più immediata che fornisce anche indicazioni di ordine pratico per chi deve progettare un sistema a chiave segreta sicuro. Essa evidenzia il fatto fondamentale che se la chiave è più breve del messaggio da cifrare, un avversario abbastanza abile può ricavare dal crittogramma informazioni utili sul messaggio. Questa fuga d'informazione avviene a prescindere da quanto complicato sia il processo di cifratura.

Viceversa il messaggio può essere protetto completamente ed incondizionatamente (perfetta segretezza) da qualunque spia in sistemi in cui la chiave è lunga quanto il messaggio, è del tutto casuale e viene usata una volta sola.

Per un esame più dettagliato di tutte le considerazioni fatte da Shannon, rimandiamo direttamente alla lettura del suo articolo [ref 5] o del [ref 18]; qui vorremmo evidenziare semplicemente che la validità della (3.14) è confermata praticamente *dall'unico sistema incondizionatamente sicuro* in uso tutt'oggi: il one time pad; si tratta di un sistema nel quale il messaggio originale è combinato con una chiave totalmente random della stessa lunghezza. Usualmente il messaggio originale è rappresentato da una stringa binaria di n-bit, che è sommata modulo 2 (equivalente ad un'operazione di XOR) con una chiave binaria random della stessa lunghezza.Come suggerisce il nome, la chiave non è mai riutilizzata.

Macchina crittografica perfetta (one time pad):



Dove:
$\otimes$ è l'operazione di exor
K è la chiave casuale
M è il messaggio non casuale
C è il cipher text casuale
Il cipher text è il risultato dell'operazione:
$C= M \otimes K$

Per ottenere il messaggio M da C conoscendo la chiave giusta K:
$M'= C \otimes K= M \otimes (K \otimes K) = M$

Condizioni:
K casuale
K lunga almeno quanto il messaggio M

Per quanto riguarda lo studio dell'autenticità, G.J.Simmons ha sviluppato una teoria che sotto diversi risvolti è del tutto analoga alla teoria della segretezza di Shannon. Non ci soffermiamo sui particolari delle considerazioni fatte da Simmons, per cui rimandiamo a [ref.2] ma riportiamo la definizione che egli dà di perfetta autenticità.

Simmons tiene conto del fatto che un crittanalista nemico può scegliere di inviare il suo falso crittogramma al destinatario senza aver prima intercettato il crittogramma autentico che era stato inviato dal mittente *(impersonation attack)* con una probabilità di riuscire ad ingannare il decifratore pari a PI oppure può decidere di attendere di vedere il crittogramma originale e solo successivamente inviare quello falso *(substitution attack)* con una probabilità di riuscita pari a PS ; definisce, quindi, *PD=max(PI,PS)* la probabilita'd'inganno (deception probability), ossia la probabilità di successo che ha un crittanalista nemico nell'ingannare chi effettua la decifratura.

Un sistema garantisce la perfetta autenticita' quando:

$$\log PD \geq -I(C, k) \qquad (3.15)$$

dove I(C,k) è una quantità derivata dalla teoria dell'informazione, è detta mutua informazione tra il crittogramma C e la chiave k ed è definita come

$$I(C, k) = H(C) - H(C/k) \qquad (3.16)$$

con H(*) entropia che abbiamo indicato subito sopra.

Garantire la perfetta autenticità significa avere la migliore protezione possibile rispetto all'inganno, cioè fare in modo che la probabilità d'inganno sia la più bassa possibile; questo, come si può facilmente vedere dalla (3.15), si ottiene soltanto quando I(C,k) è grande, cioè soltanto quando il crittogramma fornisce molta informazione circa la chiave.
Occorre notare che la teoria circa la sicurezza dei sistemi che mirano a garantire l'autenticita`e`molto meno ben sviluppata di quella relativa ai sistemi che si occupano della segretezza; in particolare, non si conosce in generale sotto quali condizioni possano esistere sistemi che garantiscono la perfetta autenticita`, anche se alcuni casi particolari sono stati realizzati.

Osservazioni:
Da Shannon si deduce che l'unico elemento di segretezza è la chiave K quindi una misura del grado di sicurezza di un algoritmo crittografico potrebbe essere la lunghezza di essa. In realtà bisogna considerare che se dentro l'algoritmo c'e' una falla questo discorso crolla, quindi una chiave troppo lunga potrebbe addirittura mascherare tali falle.
Un ulteriore problema è dato dal fatto che non è possibile dimostrare se un algoritmo è sicuro; notare che un algoritmo è sicuro se per romperlo bisogna provare tutte le chiavi possibili (brute force attack).

La lunghezza della chiave ed il tempo necessario a provare tutte le chiavi possibili sono dipendenti dalla potenza del computer utilizzato, comunque ci sono dei limiti termodinamici invalicabili:
K*T è l'energia che serve a flippare un bit se si adotta un dispositivo con efficienza 1 (non dissipativo)
Dove:
$K = 1.38*10^{-16}$ erg/°K
T_0=3.2 °K è la temperatura dell'universo

Quindi:
$Eb=K*T_0=4.4*10^{-16}$ erg quantità minima di energia per flippare un bit

$P_{sole}=1.21*10^{41}$ erg/anno

Si deduce che in un anno usando tutta l'energia del sole si fa un brute force attack di Nb=187 bit.

D'altra parte bisogna anche tenere conto che le informazioni non sono tutte uguali, esse hanno una loro vita ed un loro valore; di conseguenza gli aspetti critici di una chiave saranno:
- lunghezza
- tempo di vita
- valore dell'informazione da proteggere

2.3.2 Principi di confusione e di diffusione

Le tecniche finora illustrate vengono normalmente impiegate per realizzare praticamente due principi indicati come:
- confusione
- diffusione

Per confusione s'intende l'uso di trasformazioni di cifratura che rendano il più complicato possibile determinare come la statistica del testo cifrato dipenda da quella del testo in chiaro. In particolare, nei cifrari a trasposizione la statistica della singola lettera del testo cifrato è la stessa del testo in chiaro, mentre le statistiche di ordine più elevato del testo in chiaro sono alterate in maniera confusa, Nei cifrari a sostituzione, invece, la statistica della singola lettera del testo cifrato è la stessa del testo in chiaro.

Per diffusione s'intende la possibilità di fare in modo che una singola cifra di testo in chiaro influenzi diverse cifre di testo cifrato in modo da nascondere la struttura statistica del testo in chiaro. Un'estensione di questa idea è fare in modo che sia una sola cifra della chiave ad influenzare diverse cifre del testo cifrato in modo da rendere vano un attacco frammentario sulla chiave.

Non sembra, dunque, che si riescano ad ottenere prestazioni particolarmente brillanti; in realtà, risultati notevolmente migliori, in termini di confusione e diffusione, si possono raggiungere utilizzando insieme le due diverse tecniche: intervallando per parecchie volte trasposizioni e sostituzioni si ottengono cifrari molto robusti. Questo principio viene utilizzato nei cosiddetti cifrari a prodotto (product ciphers) che sono implementati, appunto, come successione di diversi cifrari ognuno dei quali aggiunge il suo modesto contributo all'intera quantità di diffusione e confusione.

Uno degli strumenti fondamentali in tali tecniche è la **hash function** ossia una funzione che prende in ingresso un testo e fornisce in uscita un numero. La caratteristica di tale funzione è che se il testo in ingresso differisce anche minimamente da un altro, il numero prodotto è totalmente diverso; inoltre tale funzione deve essere “collision free”, ossia non deve accadere che a input diversi ci siano output uguali.

2.4 Cifratura a blocchi e cifratura a stringa

Nel paragrafo precedente, illustrando alcune tecniche elementari di crittografia, abbiamo visto alcuni esempi di *cosa* bisogna fare sul messaggio originale per ottenere delle semplici trasformazioni dei dati.

In questo paragrafo, invece, vorremmo analizzare *come* occorre trattare il messaggio originale per poter applicare una qualunque trasformazione crittografica (non necessariamente una di quelle sopra citate, che rappresentano un ristretto, seppur significativo campionario). Fino a questo momento si è parlato del messaggio originale in maniera generica, senza mai entrare, però, in merito alle caratteristiche che esso possiede.

Il messaggio originale, che deve essere cifrato, consiste di una sequenza di simboli discreti, ognuno scelto da un insieme finito. Questi **simboli** possono essere lettere di un linguaggio, parole di un linguaggio, livelli di ampiezza di un segnale quantizzato, bits, ecc... e ad essi si fa riferimento con il termine generico di caratteri senza tenere conto della loro dimensione; I'insieme di tutti i simboli di uno stesso tipo è detto **alfabeto.**

Il messaggio originale può essere di arbitraria lunghezza (cioè una sequenza di lunghezza variabile di caratteri) e come tale non può essere trattato tutto in una volta da un componente di calcolo di fissata dimensione.

Occorre, dunque, scegliere *come* operare sul messaggio originale per poter effettuare la trasformazione crittografica. Le strategie che si possono utilizzare sono due:

Suddividere il messaggio originale in blocchi di caratteri ed applicare a ciascuno di essi la trasformazione crittografica; in tal caso si parla di **cifrario a blocchi (block cipher)** oppure effettuare la trasformazione crittografica su ciascun carattere singolarmente e trattare, dunque, il messaggio originale come fosse una stringa di caratteri; si parla, in tal caso, di **cifrario a stringa (stream cipher).**

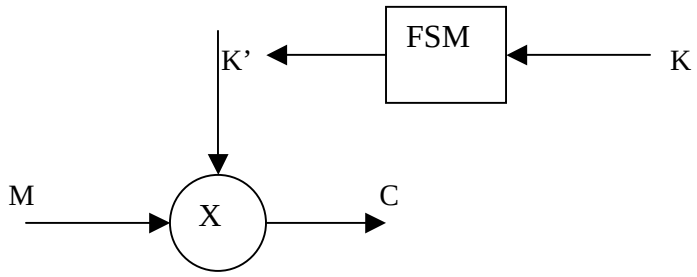
Un **cifrario a blocchi** divide il testo in chiaro in blocchi usualmente di una dimensione fissata e opera su ogni blocco indipendentemente. Un particolare blocco di testo in chiaro viene mappato nello stesso blocco di testo cifrato ogni volta che appare nel testo. I cifrari a blocchi sono, quindi, cifrari a semplice sostituzione e devono avere larghi alfabeti per contrastare l'analisi di frequenza. 1 blocchi a 64 bits del DES realizzano un alfabeto di 2^{64} caratteri.

Le proprietà che si desiderano in un cifrario a blocchi sono completamente l'opposto di quelle desiderate in un codice a correzione sistematica d'errore. Nessun bit del testo in chiaro dovrebbe mai apparire direttamente nel testo cifrato; piuttosto ogni bit del testo cifrato dovrebbe essere una funzione di tutti i bits del testo in chiaro e della chiave.

Il cifrario dovrebbe essere progettato in modo tale che cambiando anche un solo bit del testo in chiaro, o della chiave, cambino approssimativamente il 50 % dei bìts del testo cifrato. Questa proprietà è nota sotto il nome di *propagazione dell'errore (error propagation)* ed è utile nei problemi di autenticità, poiché rende improbabile che un nemico possa alterare il messaggio in una maniera che non possa essere scoperta, a meno che non sia in possesso della chiave.

I **cifrari a stringa,** invece, **non** trattano i caratteri **indipendentemente.** Ogni carattere che viene accettato come input viene cifrato in un carattere di output in un modo che **dipende sullo stato interno** del componente che effettua la cifratura.

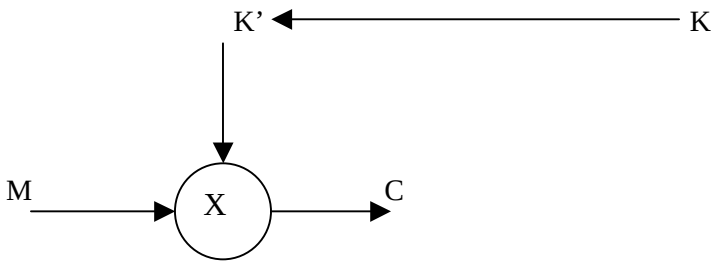
Dopo che ogni carattere è stato cifrato, il componente cambia stato seguendo sempre una stessa regola. Uno stesso carattere di testo in chiaro può, dunque, non essere cifrato nello stesso carattere di testo cifrato.



K':stream di bit generati dalla FSM a partire da K

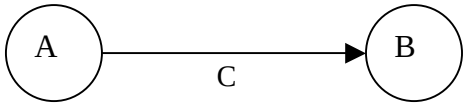
K' si ripeterà con un certo periodo (se non si ripetesse si otterrebbe una one time pad), però lo stream deve sembrare casuale.

Il più semplice degli stream cipher è:



In questo caso il periodo di K' coinciderà con la lunghezza della chiave K.

Una tecnica attaccabile solo dal punto di vista del protocollo si chiama **cipher text im cipher text**, in cui A comunica con B attraverso un cipher-text C, ottenuto facendo l'exor tra il plain-text P e la chiave K, conosciuta sia da A che da B; il plain-text P si ricava facendo l'exor tra il cipher-text C e la chiave K:



$C = P \oplus K$

$P = C \oplus K = P \oplus K \oplus K = P$

A o B generano un'altra chiave K', attraverso l'exor tra il cipher–text C ed un testo insignificante D:

$K' = C \oplus D$

Tale chiave non viene nascosta essendo una chiave "tranello", nata appositamente per decifrare il cipher text in un testo diverso dal plain-text, infatti:

$P' = K' \oplus C = C \oplus D \oplus C = D$

3

I sistemi crittografici a chiave segreta

3.1 Struttura e principi generali

I sistemi crittografici a chiave segreta rappresentano la prima generazione di sistemi progettati per trovare soluzione ai problemi fondamentali della crittografia e vengono indicati anche con il termine di sistemi convenzionali.

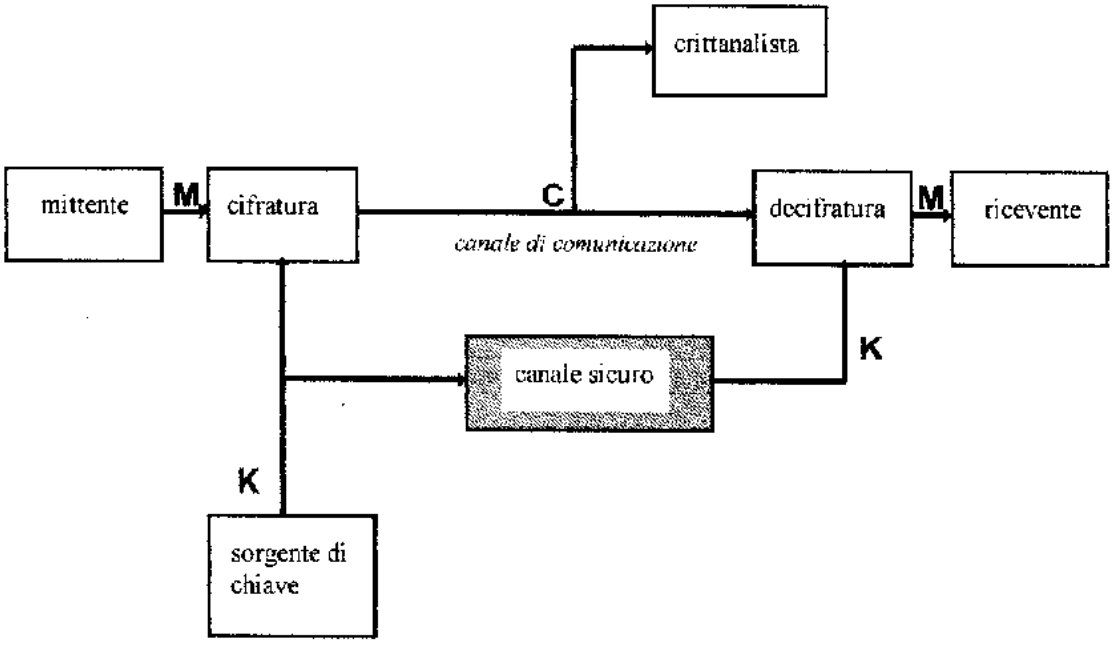


Fig.3.1 : Sistema crittografico a chiave segreta

La loro struttura, illustrata in fig. 3.1, è identica, in linea di principio a quella già analizzata nel paragrafo precedente, ma è caratterizzata da un elemento fondamentale che li contraddistingue da altri sistemi; si tratta della presenza, accanto al canale pubblico insicuro, di un canale sicuro utilizzato per comunicare al ricevente designato, protetta dagli occhi indiscreti del crittanalista nemico, la chiave segreta k, dopo che è stata generata dalla sorgente di chiave,.Come abbiamo visto, infatti, le procedure di cifratura e decifratura sono di dominio pubblico, ma sono controllate e adattate ai singoli utenti dalla chiave, in modo che un avversario che abbia intercettato il crittogramma e conosca il metodo generale di cifratura, ma non la chiave, non possa dedurre niente di utile riguardo al messaggio originale. Si pone, dunque, la necessita`di mantenere segreta la chiave e di trasmetterla segretamente solo tra mittente e ricevente che si sono precedentemente accordati. Proprio per il fatto che nei sistemi a chiave segreta sia chi effettua la cifratura sia chi effettua la decifratura utilizza la stessa chiave segreta , i sistemi a chiave segreta sono anche chiamati crittosistemi a una chiave o crittosistemi simmetrici.

Nella pratica il canale sicuro può presentarsi sotto diverse forme: può essere ad esempio un corriere fidato che recapita direttamente al destinatario la chiave segreta, oppure può essere un canale di comunicazione privato, ecc... Esso presenta comunque caratteristiche ben diverse dal canale pubblico; in sintesi potremmo dire che:

canale sicuro : è molto costoso ;
è utilizzabile solo in certi momenti ;
ha un set-up difficile.

canale pubblico : è disponibile per chiunque ;
è poco costoso ;
consente l' intercettazione.

Da un punto di vista matematico, i sistemi crittografici a chiave segreta possono essere rappresentati come una famiglia di trasformazioni invertibili $S_{k\in\{K\}}$ a singolo parametro che realizzano un mappaggio da uno spazio {M} di messaggi originali ad uno spazio {C} di messaggi cifrati. Il parametro *k* rappresenta la grandezza che fino a questo momento abbiamo indicato come chiave ed è scelto da un insieme finito {K} detto spazio delle chiavi (keyspace). Per ogni scelta di *k* , si ha una diversa coppia di trasformazioni invertibili. Scriviamo la trasformazione di cifratura come

$$C=S_k(M) \qquad (3.2)$$

per enfatizzare che il critogramma C è funzione del solo messaggio originale, dove la particolare funzione è determinata dal valore della chiave segreta *k*; parimenti la trasformazione di decifratura è data da

$$M=S_k^{-1}(C)=S_k^{-1}(S_k(M)) \qquad (3.3)$$

con significato analogo. La possibilità di avere coppie diverse di trasformazioni S_k e S_k^{-1} per valori diversi della chiave *k*, garantisce la possibilità di comunicare segretamente a coppie diverse di utilizzatori, senza correre il rischio di interferire reciprocamente.E` importante notare che *M,C,* e *k* sono quantità random; la statistica del messaggio originale *M* è determinata dalla sorgente del messaggio (mittente), mentre la statistica della chiave *k* è determinata da chi effettua la cifratura. In ogni caso si assume che *M* e *k* sono statisticamente indipendenti.

I sistemi crittografici a chiave segreta vengono impiegati sia per risolvere il problema della segretezza che quello dell'autenticità. Per questa classe di sistemi, però, alcuni ricercatori sono riusciti a formulare delle condizioni ben precise per ottenere i migliori risultati in termini di segretezza ed autenticità.

3.2 Le problematiche aperte dei sistemi a chiave segreta

I sistemi a chiave segreta , a causa della loro struttura , mostrano difficoltà a trovare una soluzione accettabile a due problemi fondamentali in crittografia e questo ne ha limitato fortemente l'impiego. Il primo problema è quello che va sotto il nome di problema della distribuzione delle chiavi (key distribution problem).

Garantire un'appropriata distribuzione delle chiavi ai mittenti e ai riceventi di messaggi cifrati è indubbiamente uno dei maggiori problemi che si incontrano nel costruire sistemi di comunicazione crittografici. Le chiavi devono essere prodotte e distribuite non una volta soltanto, ma costantemente. In alcuni sistemi esse devono essere cambiate con il passare del tempo, o con la quantità del traffico e in tutti i sistemi, poi, devono essere cambiate quando esiste il sospetto di compromissioni. Frequenti cambi di chiavi limitano la quantità di dati compromessi, nel caso un "nemico" fosse entrato in possesso della chiave. Inoltre, le chiavi devono essere assegnate ai nuovi utilizzatori del sistema e ritirate ai vecchi che non le utilizzano più.

Nei sistemi a chiave segreta, la necessità per il mittente e il ricevente di far uso di un canale fisicamente sicuro, quale può essere un corriere o la posta registrata per la distribuzione della chiave, pone dei seri vincoli alla semplicità di comunicazione. In pratica, prima di qualunque comunicazione privata è necessario effettuare un'altra transazione per la trasmissione della chiave: questo determina costi e ritardi che spesso non sono accettabili e d'altra parte costringe ad avere una preparazione a priori per la comunicazione crittografica; in affari, capita sovente che ci possa essere una conversazione privata tra due persone senza che prima vi sia stato alcun accordo, ed è irrealistico pensare di dover attendere, per avere i primi contatti iniziali, che le chiavi segrete siano trasmesse attraverso mezzi fisici.

D'altro canto, è praticamente impossibile pensare di realizzare una pre-distribuzione delle chiavi a tutti gli utilizzatori del sistema, soprattutto quando si tratta di grosse reti di comunicazione. In una rete in cui si hanno *n* utilizzatori, infatti, il numero di possibili connessioni è pari a $(n^2-n)/2$, il che significa che in un sistema con un milione di utilizzatori si hanno almeno 500 bilioni di connessioni con costi di distribuzione proibitivi.

Nel campo militare è più facile porre rimedio a questo problema, in quanto la catena di comando aiuta a limitare il numero di possibili coppie di connessioni tra utilizzatori (un colonnello in una divisione non ha necessità di comunicare privatamente con un sergente di un'altra divisione), mentre in sistemi con grande accesso casuale, diventa praticamente impossibile predire a priori quali coppie di utilizzatori desiderino comunicare privatamente in un dato giorno.

La seconda difficoltà che ha limitato l'applicazione della crittografia convenzionale, è la sua incapacità a trattare il problema della controversia (problem of dispute). Esso consiste nel fornire a chi riceve il messaggio la prova legale dell'identità del mittente.

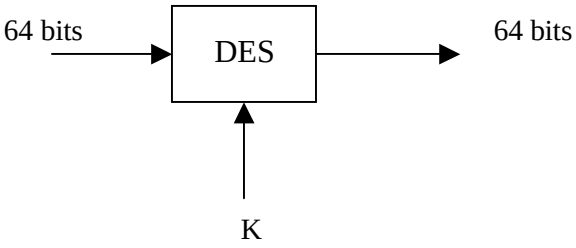
In un sistema convenzionale, il ricevente autentica ogni messaggio che riceve dal mittente decifrandolo con la chiave che essi hanno in comune. Tuttavia, proprio per il fatto che la chiave è comune ad entrambi, il ricevente ha la possibilità di produrre di sua iniziativa un crittogramma che potrebbe benissimo essere stato prodotto dal mittente, senza alcuna possibilità di distinzione. In questo modo, non si può provare che il mittente abbia attualmente mandato un messaggio controverso.

I sistemi a chiave segreta progettati per garantire l'autenticità sono in grado di prevenire l'azione di una terza parte nemica rivolta a falsificare il messaggio originale, ma non possono risolvere controversie tra mittente e ricevente circa il fatto se un dato messaggio è stato realmente inviato oppure no.

Nella pratica commerciale corrente, la validità di contratti ed accordi è garantita da firme manoscritte. Un contratto firmato serve come prova di un accordo e può essere presentata in tribunale dal suo possessore, nel caso fosse necessario; anche per le comunicazioni via computer, quindi, occorrerebbe utilizzare questo metodo ma l'uso di firme richiede la trasmissione e la memorizzazione di documenti scritti che costituiscono un grosso ostacolo ad un più generalizzato uso delle comunicazioni elettroniche in affari.

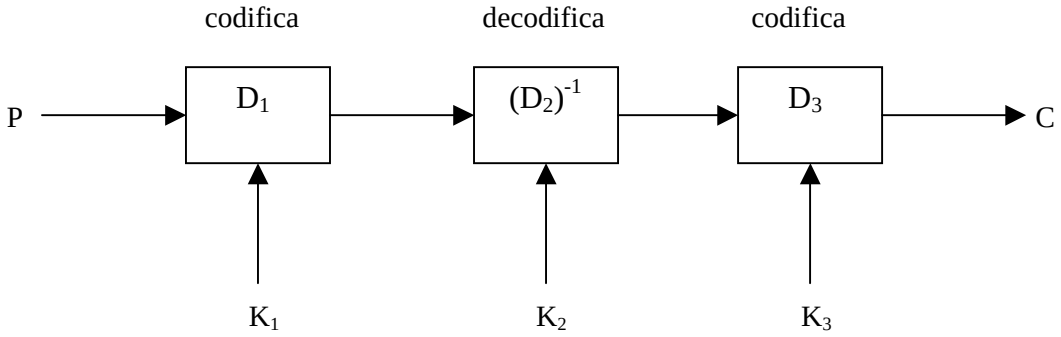
Le attuali comunicazioni in chiave privata di tutto il mondo sono regolate dal DES(Data Encryption Standard), il quale si basa su una combinazione di trasposizioni e scambi più l'uso delle hash function.

Schematicamente si può rappresentare così:



Il DES trasforma un blocco di 64 bits di testo in chiaro, P, in un blocco di 64 bits di testo cifrato, C; questa trasformazione è governata da una chiave K a 56 bits (esiste una versione militare del DES che usa una chiave di 112 bits).

Per esempio le comunicazioni satellitari sono tutte codificate con il triplo DES:



Notare che se $K_1=K_2$ si ottiene il DES classico, questo per non perdere la compatibilità. Del DES se ne parlerà in maniera più approfondita successivamente.

Recentemente è stata fatta una gara per l'algoritmo che sostituirà il DES, ed è stato deciso l'AES (Advanced Encryption Standard).
Riassumendo, i pregi della crittografia a chiave privata sono:

- "sicurezza" (basata sulla complessità dell'algoritmo)
- esistenza dell'implementazione hardware
- velocità (sia nella codifica che nella decodifica)

3.3 Il crittosistema a chiave segreta DES (Data Encryption Standard)

L'algoritmo DES è stato messo a punto nel 1974 dall' IBM in risposta al pressante invito che il National Bureau Standard (NBS) aveva rivolto ai progettisti per ideare algoritmi che potessero essere usati come standard, sia da enti privati che governativi, nella cifratura di dati.

Il DES è un cifrario a blocchi ossia un sistema crittografico che divide il testo in chiaro in blocchi separati e opera su ognuno di essi, indipendentemente, per produrre una sequenza di blocchi di testo cifrato. Esso trasforma un blocco di 64 bits di testo in chiaro, che indichiamo con P, in un blocco di 64 bits di testo cifrato C. Questa trasformazione è governata da una chiave *k* a 56 bits ed è invertibile cosicchè:

$$C=S_k(P) \qquad (6.1)$$

$$P=S_k^{-1}(C) \qquad (6.2)$$

dove S_k è la trasformazione di cifratura quando viene usata la chiave *K*.

Ci sono $2^{56}=10^{19}$ possibili valori di chiave e 2^{64} possibili valori di P (e di C), tutti in generale consentiti. Questo fatto, unitamente alla condizione che la dimensione di P=dimensione di C=64, potrebbe far pensare che il DES sia un semplice cifrario a sostituzione (substitution cipher), ma in realtà non è così.

Come abbiamo in parte accennato nel par. 2.3 un cifrario a sostituzione è il piu`generale possibile tra i cifrari a blocchi e può trasformare ogni possibile blocco di testo in chiaro in ogni possibile blocco di testo cifrato; se i blocchi sono lunghi n bits, ci sono 2^n blocchi distinti e $2^n!$ modi in cui la sostituzione può avvenire.

La peculiarità dei cifrari a sostituzione è che ciascuno di questi modi è individuato da un valore diverso della chiave e di conseguenza saranno necessari $\log_2(2^n!)$ bits di chiave, o, approssimativamente, $nx2^n$ per contemplare tutti i casi possibili. Questo numero è,però, proibitivamente grande anche per piccoli valori di *n*: es. per n=16 richiede circa un milione di bits; quindi, nel caso del DES, con blocchi di 64 bits, era davvero improponibile.

Il DES s'inserisce, invece, nella categoria dei cosidetti cifrari a prodotto (product ciphers) (si veda anche par. 2.3). Si tratta di sistemi (l'idea è di Shannon [ref. 5]) in cui la trasformazione dei dati è pensata come prodotto della forma $B_1MB_2M....B_n$, dove *M* è una trasformazione di mescolanza (mixing transformation) e B_i è una semplice trasformazione crittografica.

Il vantaggio è quello di poter utilizzare un numero di bits di chiave molto minore di quello necessario nei cifrari a sostituzione e, nel contempo, di ottenere buone trasformazioni non lineari dei dati che garantiscono che la statistica del testo cifrato sia difficilmente riconducibile a quella del testo in chiaro. Inoltre, la circuiteria elettronica ad alta velocità permette che un “product system” possa essere implementato economicamente come una singola coppia *BM*. I dati vengono cifrati in un numero di “cicli” (iterazioni) ognuno dei quali realizza una singola coppia B_iM e utilizza lo stesso hardware.

Il DES, come in generale tutti i sistemi IBM, impiega 16 cicli di successive cifrature, ognuna delle quali utilizza come trasformazione di mescolanza M una trasposizione di bits, e come semplice trasformazione crittografica B_i una sostituzione su gruppi di 4 bit del blocco di testo in chiaro. Lo schema di principio è indicato in Fig. 6.1 dove S indicano i blocchi di sostituzione e T quelli di trasposizione. Vedremo, nel prossimo paragrafo che in realtà il DES, nella sua implementazione, complica un po’ questo schema di base.

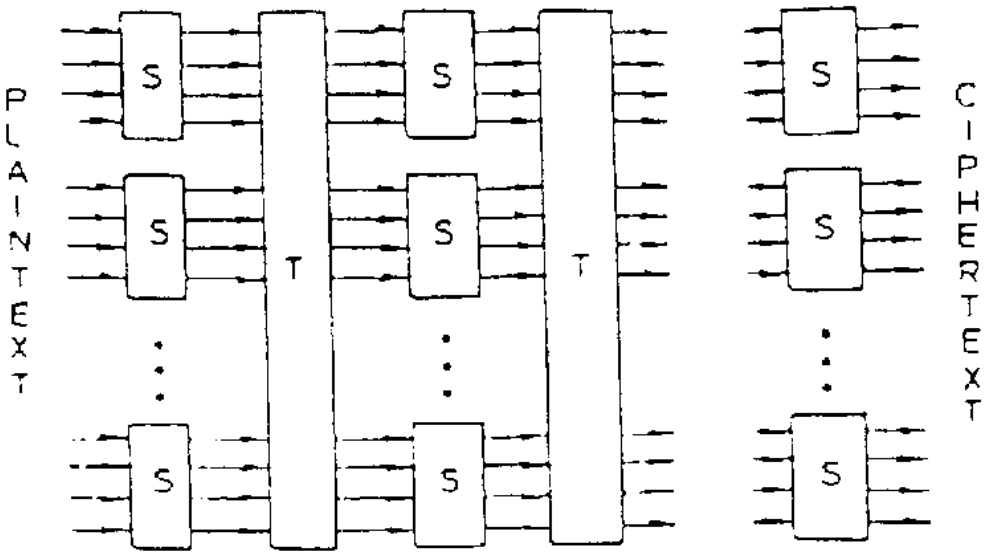


Fig. 6.1 :Schema di principio di un product cipher realizzato con alternanza di sostituzioni (S) e trasposizioni (T)

3.3.1 L' algoritmo

L' algoritmo adottato per il DES è basato sempre su blocchi (boxes) S e T come illustrato nel paragrafo precedente, ma usa una struttura un po' più complicata di quella indicata. In Fig. 6.2a si può osservare che dopo una fissata permutazione iniziale (IP) il blocco a 64 bit di testo in chiaro viene rotto in due metà sinistra e destra, $L0$ e $R0$ rispettivamente, ciascuna lunga 32 bits.

Queste due metà passano, quindi, attraverso 16 iterazioni di una trasformazione dipendente dalla chiave (Fig. 6.2b) per produrre un blocco di "preoutput" a 64 bits. Questo blocco di preoutput viene sottoposto ad una permutazione finale per produrre un blocco di testo cifrato a 64 bits.

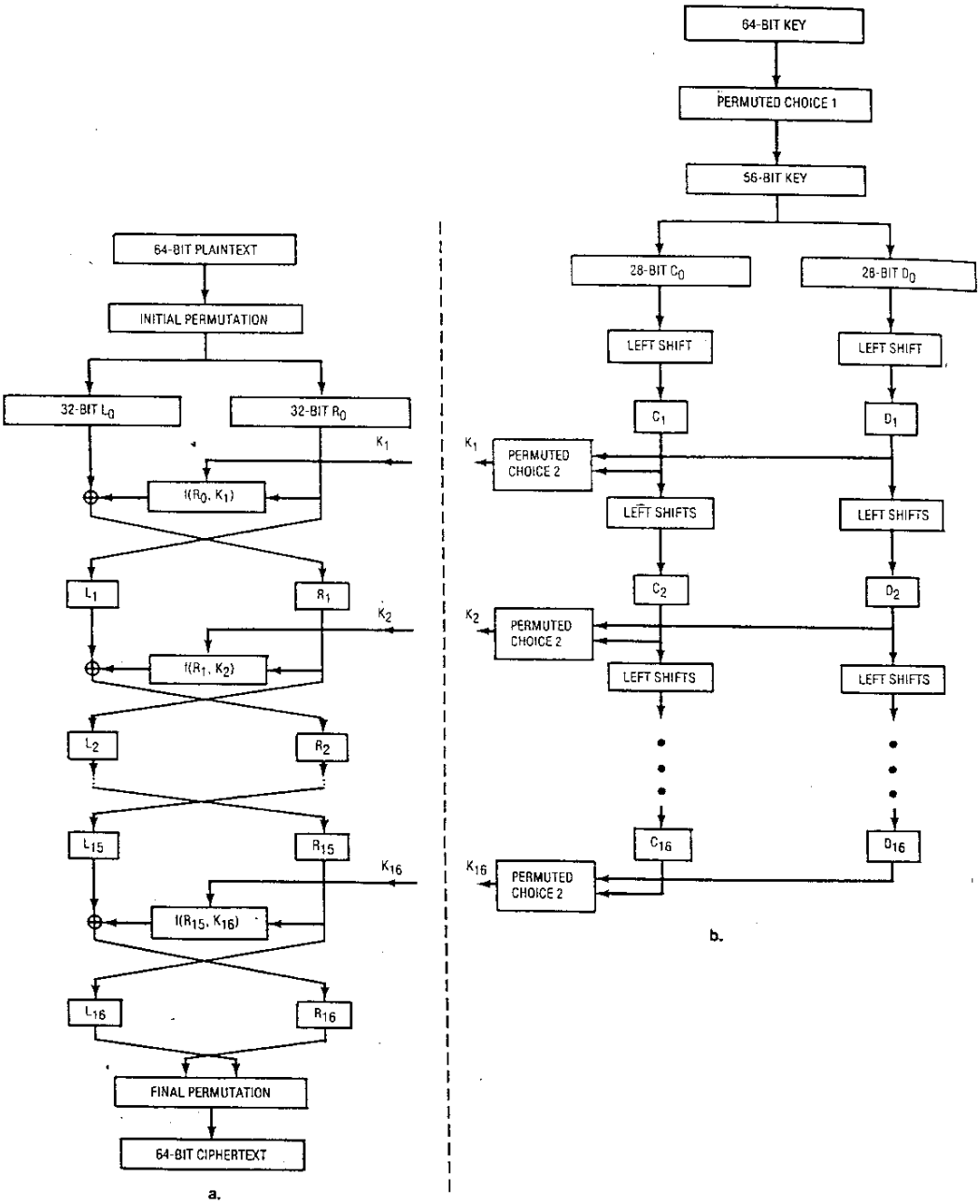


Fig. 6.2 :Diagrammi di flusso per gli algoritmi di cifratura (a)
e di schedulazione della chiave (b).

Le permutazioni iniziale e finale potrebbero anche essere eliminate dallo standard poiché non hanno alcun valore crittografico e occupano circa il 20% del tempo di cifratura in un' implementazione software ; tuttavia, esse sono contemplate nel progetto originale e quindi le terremo presenti anche noi.
Il loop di base che viene iterato per 16 volte è della forma :

$$Li=Ri-1 \qquad (6.3)$$

$$Ri = Li-1 \oplus f(R_{i-1},k_i) \qquad (6.4)$$

i=1,2,...,16 dove k_i un insieme di 48 bits di chiave scelti dalla chiave in accordo con un algoritmo di schedulazione della chiave (Fig.6.2b) ; L_i e R_i denotano rispettivamente la metà sinistra e destra del blocco a 64 bit dopo la i-sima iterazione ; $\oplus$ denota l' operazione di OR esclusivo e f è una funzione con un output a 32 bit. E' importante notare che non è necessario invertire la funzione f durante la decifratura, poiché $L_{i-1}R_{i-1}$ può essere calcolato da L_iR_i come :

$$Ri-1=Li \qquad (6.5)$$

$$Li-1=Ri \oplus f(Li,ki) \qquad (6.6)$$

anche se la funzione $f(.,k)$ è molti-a-uno. Poichè tutte le altre operazioni sono lineari in aritmetica binaria, la funzione $f(R,k)$ è la base di tutta la sicurezza.

L' algoritmo di schedulazione della chiave è rappresentato dal diagramma a blocchi di Fig. 6.2b, mentre la funzione $f(R,K)$ è rappresentata in Fig. 6.3.

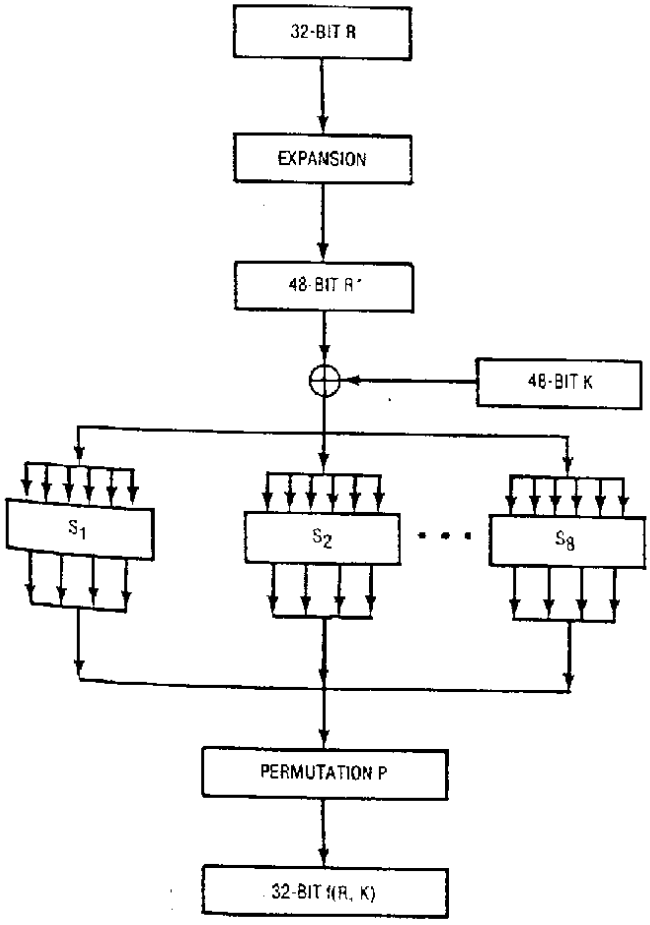


Fig. 6.3 :Calcolo di f(R,k)

Con riferimento alla Fig. 6.2b, la chiave a 64 bits dapprima subisce un'iniziale permutazione (permuted choice1) che scarta gli 8 bits di chiave numerati 8, 16, 24, 32, 40, 48, 56, e 64 fornendo come risultato una chiave a 56 bits. Gli otto bits scartati sono riservati per la parità, mentre i 56 bits restanti sono caricati in due shift-registers a 28 bits chiamati *C* e *D*. I contenuti di questi registri alla *i* -sima iterazione sono denotati C_i e D_i .

- C_i e D_i sono ottenuti da C_{i-1} e D_{i-1} shiftando entrambi i registri di una o due posizioni verso sinistra. Il numero di shifts a sinistra (uno o due) ad ogni iterazione è una parte fissa dell'algoritmo che viene chiamata lista degli shifts (o shift schedule).
- k_i è ottenuta da C_i e D_i scegliendo 48 dei 56 bits disponibili. Questa scelta (permuted choice 2) è fissata ed è la stessa per tutte le iterazioni.

Con riferimento alla Fig. 6.3, *f*(*R*,*k*) è calcolata come segue. La quantità *R* a 32 bits viene espansa in un *R'* a 48 bits ripetendo i bits numerati 1, 4, 5, 8, 9, 12, ..., 24, 27, 28, 32 (il primo e l'ultimo bit di successivi gruppi di 4 bits). I bits ripetuti sono affiancati ai gruppi adiacenti di 4 bits per formare 8 nuovi gruppi da 6 bits ciascuno; per cui si ha, se

$$R=r_1r_2r_3....r_{31}r_{32}$$

allora

$$R'=r_{32}r_1r_2r_3r_4r_5,r_5r_6r_7r_8r_9r_{10},..., r_{28}r_{29}r_{30}r_{31}r_{32}r_1.$$

Le due quantità a 48 bits *R'* e *k* sono, quindi, messe in XOR e spezzate in 8 gruppi a 6 bits, che costituiscono gli ingressi a 8 differenti blocchi S (S-boxes: S_1, S_2, ... , S_8) con 6 ingressi e 4 uscite. I 32 bits di uscita di tutti gli S-boxes sono permutati da P e quindi offerti come *f*(*R*,*k*).

3.3.2 Caratteristiche degli S-boxes

E`importante soffermarsi sulle caratteristiche che devono avere gli S-boxes perchè essi giocano il ruolo principale nell'algoritmo, essendo i responsabili della trasformazione crittografica in senso proprio.
E`cruciale, per garantire l'inviolabilità del sistema, che gli S-boxes siano non affini, cioè operino una trasformazione non affine. Una trasformazione affine e`una trasformazione dei dati di tipo lineare+costante, ossia presi per esempio due bits *x* e *y*, essa opera un mappaggio dall'uno all'altro con l'operazione:

$$y=Ax\oplus b \qquad (6.7)$$

dove l'aritmetica è fatta modulo 2.
La necessità di avere una trasformazione non affine, cioè che non sia del tipo (6.7), è motivata dal fatto che, come già evidenziato, tutte le operazioni contemplate nell'algoritmo (tranne appunto il mappaggio degli S-boxes) sono lineari (la trasposizione è lineare e la composizione di mappaggi affini sarebbe a sua volta affine) e di conseguenza l'uso di S-boxes affini può rendere l'intero algoritmo affine nella forma:

$$c=Ap\oplus Bk\oplus b \qquad (6.8)$$

dove *A*, *B*, *b* sono fissati, *k* è la chiave a 56 bits, *c* è il crittogramma e *p* il testo in chiaro.
In tal caso, la conoscenza di anche una sola coppia *p*-*c* potrebbe consentire di calcolare *k* come:

$$k= B^{-1}(c-Ap-b) \qquad (6.9)$$

dove B^{-1} è la pseudoinversa di B, e violare la sicurezza del sistema.
Si è visto [ref. 1] che è possibile ottenere S-boxes a *q* bits non affini scegliendo opportunamente il valore di *q*; da studi effettuati dall'IBM è emerso che tutti i mappaggi invertibili 2-bit a 2-bit (quindi con *q*=2) sono affini, mentre solo il 3% dei mappaggi invertibili 3-bit a 3-bit è affine e quindi il valore *q*=3 potrebbe già andare bene; tuttavia,l'IBM considera come primo valore adeguato a fornire sufficienti margini di sicurezza *q*=4.

Un'altra proprietà che hanno gli S-boxes è che cambiando un qualunque singolo ingresso cambiano almeno due uscite. Se un solo bit del testo in chiaro o della chiave viene cambiato, immediatamente si verifica una

valanga di cambiamenti nel testo cifrato che non permette di fare considerazioni di tipo statistico; tale proprietà, nota, come abbiamo visto nel par. 2.3, con il nome di *propagazione dell'errore* consente di evitare i cosidetti *attacchi a insiemi di chiavi (key cluster attack)*. In un attacco di questo tipo, il crittanalista cerca, per prima cosa, una chiave che sia vicina alla chiave corretta e quindi prova tutte le chiavi che sono in un intorno di essa (un cluster di chiavi) finchè trova quella giusta. Questo metodo è molto più rapido di una ricerca esaustiva.

Tuttavia, la propagazione dell'errore è necessaria ma non sufficiente per evitare "key cluster attacks". L'espansione *E* e la permutazione *P* di Fig. 6.3 sono accoppiate in modo tale che le quattro uscite di ogni S-box, dopo essere passate attraverso *P* ed *E*, si presentano come ingressi a 6 diversi S-boxes e questo incorraggia una rapida propagazione dell'errore.

3.3.3 Dimensione della chiave segreta e sicurezza del sistema

Come già evidenziato in altri paragrafi, la scelta della dimensione della chiave, nei sistemi crittografici a chiave segreta, rappresenta uno dei nodi cruciali da risolvere. Si tratta, infatti, di conciliare due diverse esigenze che sono fondamentali per il sistema, ma che sono orientate in direzioni opposte.

Da un lato vi è la necessità di contenere la dimensione della chiave per renderne possibile fisicamente l'implementazione, dall'altro, invece, vi è la necessità di ingrandirla il più possibile per avere un sempre maggior numero di combinazioni di chiave utilizzabili e,di conseguenza rendere più difficoltosa l'identificazione della chiave correttamente utilizzata. La scelta operata nel caso del DES, con 56 bits di chiave, è stata oggetto di grande discussione negli ambienti crittografici e ha riscosso, alternativamente, critiche e consensi.

L' obiezione di fondo nasce dal fatto che una chiave a 56 bits sembrerebbe insufficiente a garantire la sicurezza del sistem qualora esso venga sottoposto ad una *crittanalisi esaustiva* cioè ad un attacco crittanalitico *"forza bruta"* nel quale il crittanalista in possesso di una coppia (*p*,*c*) testo in chiaro-testo cifrato conosciuta, decifra il crittogramma *c* utilizzando, una dopo l'altra, tutte le $2^{56}=10^{19}$ possibili chiavi, finchè trova quella che gli permette di ricavare esattamente il testo in chiaro *p* di cui è in possesso.

Sondare così a fondo lo spazio delle chiavi potrebbe apparire una cosa irrealizzabile poichè, essendo $2^{56}=10^{19}$ le possibili chiavi e supponendo di impiegare anche soltanto un microsecondo per provare se ciascuna di esse è quella giusta, sarebbero necessari 10^{11} secondi, cioè circa 10^{6} giorni per una ricerca esaustiva.

Dal momento, però, che lo standard è stato scelto per essere implementato su un singolo chip LSI, si potrebbe costruire una macchina con un milione di questi chips che ricercano in parallelo e in tal caso basterebbe una sola giornata per effettuare una ricerca esaustiva e mezza giornata per una ricerca media.

I ricercatori Diffie e Hellman della Stanford University hanno fornito una dettagliata giustificazione circa la realizzabilità di una macchina di questo tipo e hanno stimato il suo costo intorno ai 20 milioni di dollari e il costo medio per soluzione intorno ai 5000 dollari; sebbene si tratti di una grossa somma, tuttavia oggi essa potrebbe essere facilmente investita da una qualche grande organizzazione.

Inoltre, occorre tenere presente il progressivo decrescente costo che è richiesto dalle operazioni di calcolo: in 10 anni i 20 milioni di dollari del costo della macchina potrebbero scendere a 200.000 dollari e i 5000 dollari del costo della soluzione potrebbero diventare soltanto 50 dollari. Sia l'investimento iniziale che il costo per soluzione potrebbero essere dunque troppo piccoli per offrire un adeguato livello di sicurezza.

Da più parti il DES è stato giudicato, di conseguenza, soprattutto in passato, un sistema non sufficientemente sicuro; a dispetto, però delle numerose approfondite analisi cui e`stato sottoposto dai diversi ricercatori, esso non ha mostrato alcuna debolezza ad un tipo di attacco crittanalitico migliore della pura e semplice ricerca esaustiva e, d'altra parte, anche nel caso della ricerca esaustiva, recentemente, le conclusioni cui si era arrivati in precedenza sono state ribattute e riviste e corrette (confronta [ref. 8]).

In definitiva, il consenso generale dei ricercatori crittografici e`, oggi, che il DES costituisce un estremamente buon cifrario con una sfortunatamente piccola chiave.

Si è pensato di apportare dei correttivi allo schema base, anche se questo richiederà una notevole spesa. Lo standard, infatti, deve essere incluso nei terminali, nei disk drives, nelle unità a nastro sotto forma di hardware ed è necessario, dunque, modificare gli apparati per accettare un nuovo standard. Se i vincoli di progetto lo permetteranno e la sicurezza sarà adeguata, si potrà cercare di minimizzare i costi di cambiamento usando moduli *plug-in*.

Lo standard revisionato avrà probabilmente una chiave più larga: si passerà dagli attuali 56 bits di chiave a 128 o 256; questo comporterà un margine di sicurezza di tutta tranquillita`. Infatti, l'uso di una chiave a 128 bits fara`crescere il costo stimato per l'attacco forza bruta da 5000 dollari a 2x1025 dollari e nessun progresso tecnologico potrà ricondurlo in un range ragionevole.

Attualmente, per accrescere la sicurezza del DES, è possibile effettuare cifrature multiple con chiavi differenti, cioè realizzare un cifrario a prodotto con il DES come cifrari componenti. Questa possibilità potrebbe essere tenuta in considerazione da inserire definitivamente, in futuro, nello standard ma appare assai meno probabile, rispetto all'aumento della dimensione della chiave, perchè comunque rappresenta una tecnica imprecisa.

4

I sistemi crittografici a chiave pubblica

4.1 Introduzione e schema di principio

Nei sistemi crittografici a chiave segreta, i problemi maggiori nascono dalla necessità di avere un canale sicuro che consenta lo scambio della chiave segreta tra mittente e ricevente, indispensabile alle operazioni di cifratura e decifratura.

Diffie e Hellmann [ref.6] e, indipendentemente, Merkle [ref.7] hanno suggerito che in realtà è possibile fare a meno di tale canale sicuro per la distribuzione della chiave e che **è possibile comunicare in maniera sicura anche sul canale di comunicazione insicuro senza alcun pre-arrangiamento**.

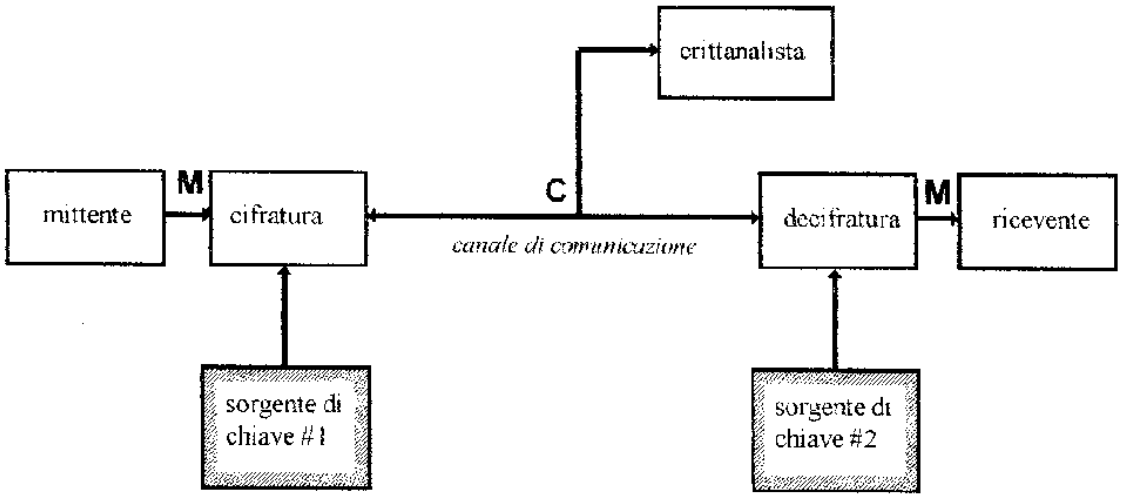


Fig.4.1 : Schema di principio di un sistema crittografico a chiave pubblica

Sistemi che permettono questo, cioè di creare una connessione sicura comunicando solo su un canale pubblico e usando solo tecniche pubblicamente conosciute, sono chiamati **sistemi a chiave pubblica**, (in contrasto con i sistemi convenzionali) e la loro struttura è indicata in Fig.4.1.

Si può notare che è permessa tra mittente e ricevente una comunicazione bidirezionale (indicata dalla doppia freccia) e il crittanalista ha un ruolo puramente passivo, ossia quello di semplice e indesiderato ascoltatore; inoltre, è scomparso il canale sicuro e la chiave non è più una soltanto , bensì due differenziate per mittente e ricevente; vedremo in seguito, addentrandoci più nei dettagli di tali sistemi che in realtà, per ogni utilizzatore del sistema, si ha una coppia di chiavi : una "segreta"e l'altra "pubblica"; l'utente tiene al sicuro la propria chiave segreta e rende disponibile la chiave pubblica.

Esempio:
A e B devono comunicare segretamente ed il canale che vogliono usare è il servizio postale.
Seguendo i passi successivi A e B riescono a comunicare segretamente anche usando un canale pubblico:

1) A mette nella cassetta da spedire il messaggio, la chiude con un lucchetto avente chiave Ka e la spedisce a B
2) B riceve la cassetta, aggiunge un suo lucchetto avente chiave Kb e la spedisce ad A
3) A riceve la cassetta, apre il lucchetto con la chiave Ka e la spedisce a B
4) B riceve la cassetta e apre il lucchetto con la chiave Kb

Questo è un protocollo, non è un algoritmo di crittografia; l'algoritmo è l'equivalente del lucchetto.
Protocollo e algoritmo sono quindi indipendenti.

Riguardando i passi dell'esempio precedente dal punto di vista crittografico:

1) Il messaggio M viene cifrato con A: Ea(M)
2) Il messaggio M viene cifrato con B: Eb(Ea(M))
3) Il messaggio M viene decifrato con A: Da(Eb(Ea(M)))
4) Il messaggio M viene decifrato con B: Db(Da(Eb(Ea(M))))

Se ne deduce per la crittografia a chiave pubblica serve un'operazione matematica che sia associativa e commutativa.

4.2 La matematica per la crittografia

La crittografia, come abbiamo visto, si occupa di sviluppare tecniche per la *trasformazione dei dati* allo scopo di garantire che informazioni riservate possano essere trasmesse dal mittente al legittimo destinatario senza il rischio di essere intercettate da un nemico. Tali trasformazioni dei dati sono state realizzate nel corso dei secoli (la crittografia è molto antica, risale addirittura a Cesare) con modalità diverse e facendo riferimento a discipline diverse. Oggi, con l'avvento delle moderne tecnologie e con il fatto che la trasmissione dell'informazione avviene sempre maggiormente per via informatica, le trasformazioni crittografiche dei dati sono affidate, per lo più, all'elaborazione dei calcolatori e, come tali, fanno riferimento a principi matematici. Gli algoritmi che implementano i diversi tipi di sistemi crittografici si basano spesso su modelli matematici ben definiti con regole precise e assiomi inconfutabili.

In questo paragrafo, vorremmo illustrare alcune nozioni matematiche fondamentali che vengono normalmente impiegate in crittografia e che saranno utili per comprendere più a fondo i sistemi che analizzeremo in seguito.

4.2.1 L'aritmetica modulo n

L'aritmetica modulo n è un sistema di calcolo che opera su un insieme finito Z di n cifre intere, definito come: Z={0,1, ... n-1}. Su tale insieme si possono effettuare le operazioni dell'aritmetica tradizionale e in particolare le quattro fondamentali: addizione, sottrazione, moltiplicazione, divisione.

La peculiarità sta nel fatto che i termini e il risultato di ciascuna operazione non possono assumere un valore qualunque, bensì uno tra quelli contemplati nell'insieme Z che sono gli unici valori ammessi. Nell'aritmetica tradizionale l'insieme dei numeri utilizzabili per il calcolo è infinito, comprende cioè numeri dall'insieme {0,...,n-1} e il calcolo di una certa quantità è rappresentabile, dunque, con un qualsivoglia numero grande a piacere.

Nel caso, invece, dell'aritmetica modulo n, essendo l'insieme Z finito, il conteggio di una certa quantità si ottiene operando ciclicamente su tale insieme: una volta che il conteggio è arrivato alla massima cifra

rappresentabile n-1, ricomincia dalla prima e così via. Si parla, infatti, per l'aritmetica modulo n, di **struttura ad anello.** Per capire meglio la differenza di calcolo delle due aritmetiche facciamo un esempio.

ESEMPIO:
Supponiamo di voler effettuare l'operazione di somma 3+4.

In aritmetica tradizionale, questo significa contare dapprima 1,2,3 } 3 unità
e poi continuare con 4,5,6,7 } 4 unità;

Il risultato dell'operazione è, dunque rappresentato dal numero 7 e si può scrivere:

3+4=7

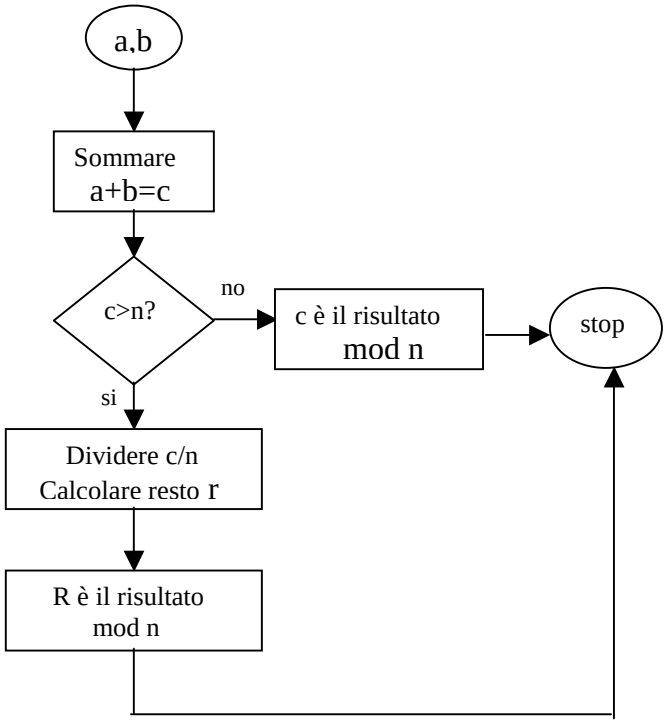
Volendo effettuare la stessa operazione in aritmetica modulo 5, in cui n=5 e Z={0,1,2,3,4}, il risultato che si ottiene è ben diverso. Infatti, anche in questo caso il meccanismo di conteggio in successione rimane lo stesso, ma esso non procede indefinitamente, bensì torna su sé stesso.
In aritmetica modulo 5 le cifre 5,6,7 non hanno ragione di essere; il conteggio parte da 0 ma si ferma a 4 e poi ricomincia da 0.
Il numero totale delle unità contate, alla fine, deve comunque essere quello indicato nell'operazione.
Si conterà, dunque dapprima 1,2,3 } 3 unità
e poi 4,0,1,2 } 4 unità.
Il risultato dell'operazione, in aritmetica modulo 5, non è più 7, ma 2.
Possiamo allora scrivere che:

3+4=2 *(mod* 5)

dove la notazione *(mod* 5) ricorda che il calcolo è stato fatto in aritmetica modulo 5.

Detto questo, che spiega la differenza concettuale tra le due aritmetiche, è possibile facilmente calcolare risultati di operazioni in aritmetica modulo *n* dopo averle eseguite prima in aritmetica tradizionale.

Questo si ottiene dividendo il risultato dell'operazione in aritmetica tradizionale per *n* e tenendo come risultato valido per l'operazione modulo *n il* resto di tale divisione. Il procedimento da seguire può essere schematizzato mediante un diagramma di flusso come quello di Fig. 2. la dove a, b, c rappresentano generici numeri e dove viene considerato il caso dell'operazione di somma (ovviamente il discorso non cambia per le altre operazioni). Accanto ad esso, in Fig.2. 1b è rappresentato l'esempio numerico cui si faceva accenno qui sopra.
Grazie alla sua proprietà di rimanere sempre e comunque all'interno di un range prefissato, che è quello compreso tra 0 *e n-1,* l'aritmetica modulo *n* consente di operare con valori anche molto grandi senza il timore che i risultati crescano in maniera incontrollata.



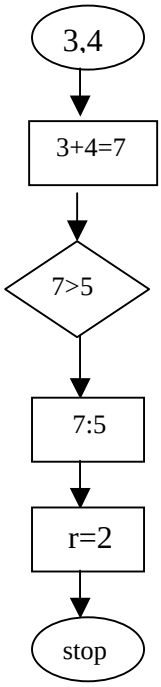


Fig. 2.1: **(a) Diagramma di flusso di un'operazione di somma modulo n. a,b,c,r,n sono numeri generici. (b) Esempio numerico di somma modulo 5. Qui a=3, b=4, n=5.**

Volendo calcolare a^x mod n ci sono due modi possibili, prendiamo per esempio x=8 (ed a< n):

a^8 mod n = (a a a a a a a a) mod n implica 7 moltiplicazioni e 1 operazione di modulo

oppure

a^8 mod n = $((a^2)^2)^2$ mod n = $[(a^2 \text{ mod } n)^2 \text{ mod } n]^2$ mod n implica 3 moltiplicazioni e 3 operazioni di modulo

Questo esempio è stato fatto per evidenziare come la stessa operazione se effettuata in modi diversi possa avere pesi computazionali diversi, infatti l'operazione di modulo può essere effettuata attraverso sottrazioni e shift (computazionalmente "leggeri") quindi il secondo modo di procedere è preferibile al primo per ciò che riguarda il peso computazionale (l'operazione di moltiplicazione in un DSP può richiedere dai 50 ai 100 colpi di clock, lo shift richiede 1 colpo di clock).
Per ricavare il numero di moltiplicazioni necessarie per realizzare il calcolo di a^x *mod n* basta scrivere *x* in forma binaria e poi scomporre a^x nel prodotto di più *a* con esponente multiplo di due, esempio:

a^{25} mod n =a^{11001} mod n =(a a^8 a^{16}) mod n = (a $((a^2)^2)^2$ $(((a^2)^2)^2)^2$) mod n

Questo procedimento è possibile implementarlo per esempio con il linguaggio C:

```
long qe2(long x, long y, long n)
{
        long s,t,u;
        s=1;
        t=x;
        u=y;
        while(u)
        {
```

```
            if((u&1)!=0)        /* u&1 fa l'and bit a bit di u con 1, quindi estrae il bit meno significativo di
u */
                    s=(s*t)%n;
            u=u>>1;  /* shift a destra di 1, equivale a dividere u per 2 */
            t=(t*t)%n;
        }
        return s;
}
```

4.2.2. Elementi di teoria dei numeri

Definiamo in questo paragrafo alcune grandezze e alcune proprietà della teoria dei numeri che sono alla base di alcuni algoritmi crittografici.

Nella trattazione che segue denotiamo con **gcd(i,n)** il massimo comune divisore dei numeri interi i e n (non entrambi 0) (la sigla gcd deriva dall'espressione inglese "greatest common divisor"): per esempio, gcd(12, 18)=6.

4.2.2.1 La Euler Totient Functìon

Una quantità spesso utilizzata in crittografia è la *Euler totient function φi(n),* dove *n* e' un intero positivo, ed è definita come il numero di interi positivi i minori di *n* che sono primi rispetto a *n,* tali cioè che gcd *(i,n)=1* (eccetto φ(1)=1 per definizione).

Per esempio, φ(6)=2 poiché per 1 ≤ i < 6 soltanto i =1 e i =5 danno gcd(i,6)=1.

Si nota immediatamente che se p è un numero primo, allora si ha che

$$\varphi(p) = p-1 \qquad (2.1)$$

e,inoltre, per le proprietà elementari della funzione φ, se *n=p *q* e se p e *q* sono due distinti numeri primi, allora

$$\varphi(pq) = \varphi(p)*\varphi(q) \qquad (2.2)$$
$$= (p-1)*(q-1)$$
$$= n-(p+q)+1.$$

Per esempio, per n=6, p=2 e q=3 si ha φ(6)=φ(2x3)= 1 x2=2.

Riportiamo qui di seguito gli enunciati di due teoremi fondamentali della teoria dei numeri senza però addentrarci nella loro dimostrazione che, per i nostri scopi, è del tutto inutile e avrebbe il solo risultato di appesantire la trattazione; per un approfondimento dell'argomento rimandiamo, tuttavia, alla consultazione dì un qualunque manuale di teoria dei numeri.

4.2.2.2 Proprietà fondamentali
Teorema 1 (Teorema di Eulero)

Per tutti gli interi positivi x e n con x < n e tali che gcd(x, n)= 1 si ha che

$$x^{\varphi(n)} = 1 \text{ (modulo n).} \qquad (2.3)$$

Per esempio: $5^2 = 1$ (modulo 6).

Teorema2
Se *e* e *m* sono due numeri che soddisfano $0 < e < m$ e tali che gcd(*m*, *e)=1,* allora esiste un unico *d* tale che $0 < d < m$ e

$$d*e=1 \text{ (modulo m).} \qquad (2.4)$$

4.2.2.3 Il teorema di Fermat

Il teorema di Fermat asserisce che per qualunque intero positivo b minore di un dato numero primo *p*

$$b^{p-1}=1 \text{ (modulo } p\text{)}$$

Per esempio, $2^4=1$ (modulo 5). Se uno desidera testare un dato numero *r* per vedere se è primo, può scegliere un qualunque intero positivo *b* minore di *r* e controllare se

$$b^{r-1}=1 \text{ (modulo } r\text{).}$$

Se la risposta è negativa si ha, allora, l'assoluta certezza di Fermat che *r* non è un numero primo; se, invece, la risposta è affermativa, si puo'cominciare a sospettare che *r* sia primo e si può dire che *r* è uno *pseudoprimo sulla base b*.

Si può dimostrare che se *r* non è un numero primo, può essere un pseudoprimo per meno della metà di tutte le possibili basi *b*. Quindi, se scegliamo ad esempio *t* basi *b* completamente a caso, la probabilità che *r* verifichi la condizione di Fermat, se non è primo, è minore di 2^{-t}.

Questo test per verificare se un numero è primo o no è stato messo a punto da Solovay e Strassen e perfezionato da Rabin [ref. 19] ed è oggi usato per trovare i due numeri primi necessari all'algoritmo RSA; un computer ad alta velocità impiega pochi secondi per determinare se un numero a 100 cifre è primo e impiega 1-2 minuti per trovare il primo numero primo da un dato punto in poi.

Rabin Miller test:
Input: numero P da testare
Output: dice se P è primo o no

0) b => max K 2^b = p-1 b è la massima potenza di 2 tale che 2^b divide p-1
m tale che p = m 2^b +1 esprime p come potenza di 2 a cui somma 1 in quanto p sarà certamente dispari

1) scegliere a < p random, j = 0, z=a^m mod p

2) if(z = = 1) or (z = = p-1) => ok p è un numero primo

3) if(j>0) and (z = = 1) => no p non è un numero primo

4) j=j+1;
if(j < b) and (z != p-1)
z = z^2 mod p
goto 3)
if(z = = p-1) => ok p è un numero primo

if(j = = b) and (z != p-1) => no p non è un numero primo

Il numero di tentativi da fare è dell'ordine di 6 test, in cui la probabilità di aver sbagliato è $< 1/2^{51}$

4.2.3 La ricerca di numeri primi "grandi "random"

La scelta di numeri primi "grandi" è una questione di notevole importanza in crittografia, al punto che un sistema a chiave pubblica, l'RSA (si veda par. 6.2) fonda su di essa la propria sicurezza. Quando si parla di numeri "grandi s'intende numeri sull'ordine delle 100 cifre decimali; i problemi che si pongono sono fondamentalmente due:

1) Come scegliere "a caso" due numeri di questo tipo che siano primi fra loro;

2) Come è possibile riconoscere che un numero è primo, senza fattorizzarlo (visto che trattandosi di numeri "grandi"la fattorizzazione sarebbe tutt'altro che semplice).

In merito al problema 1) un teorema dei numeri primi stabilisce che la frazione di numeri interi positivi minori di un qualunque altro intero *m* "grande" che sono primi è pari a $(\ln m)^{-1}$. Per esempio, la frazione di interi minori di 10^{100} (che è un numero a 100 cifre decimali) che sono primi è circa $(1n\ 10^{100})^{-1}=1/230$. Tale frazione, poi, si dimezza ancora e diventa pari a 1/115 se si scelgono soltanto interi dispari. Quindi, per cercare un numero primo "random" a 100 cifre, occorre generare, in media, 115 numeri dispari a 100 cifre prima di trovarlo.

Per quanto riguarda, invece, il problema 2) esiste un test,che si basa sul *Teorema di Fermat*, che permette di dimostrare che un numero è primo senza fattorizzarlo.

In merito a risolvere, in particolare, il problema della distribuzione della chiave, sono stati sviluppati nell' ambito dei sistemi a chiave pubblica, due diversi approcci definiti **sistemi a distribuzione di chiave pubblica (public key distribution systems)** e **crittosistemi a chiave pubblica (public key cryptosystems)**; i primi sono molto piu`legati alla realizzazione, mentre i secondi sono molto piu`potenti poiche`si prestano a risolvere anche problemi di autenticità.

4.3 I sistemi a distribuzione di chiave pubblica

Tali sistemi sono stati studiati con il preciso scopo di eliminare la necessita`di un canale sicuro per la distribuzione della chiave. L'obiettivo e`quello di far sì che due utilizzatori del sistema, A e B, possano scambiarsi in maniera sicura una chiave sopra un canale insicuro. Tale chiave viene quindi usata, in un secondo tempo, da entrambi gli utilizzatori all'interno di un crittosistema convenzionale per effettuare le operazioni di cifratura e decifratura.

Il meccanismo fondamentale che e`alla base di tali sistemi e` molto semplice: i due utenti che vogliono scambiarsi la chiave comunicano tra loro *avanti e indietro* (si spiega così la doppia freccia in figura 4.1) finche`arrivano ad una chiave in comune; questo scambio di informazioni puo`avvenire tranquillamente sotto lo sguardo indiscreto del crittanalista, a cui pero`deve risultare *computazionalmente irrealizzabile* ricavare la chiave dalle informazioni sottratte di nascosto.

Per meglio chiarire questo meccanismo, analizziamo qui lo schema proposto da Diffie-Hellman, che sebbene costituisca la piu`vecchia proposta per eliminare il trasferimento di chiavi segrete in crittografia, e`considerato, comunque, ancor oggi, uno dei piu`sicuri e piu`pratici schemi a distribuzione di chiave pubblica.

Esso si basa sulla apparente difficolta`di calcolare logaritmi discreti sopra un campo finito *GF(q)* [ref.11] con un numero primo *q* di elementi.La aritmetica cui si fa riferimento e`quella *modulo q* (vedi Par. 2.2).

Consideriamo la *funzione esponenziale discreta*:

$$Y=a^X \mod q \qquad 1\leq X\leq q-1, \tag{4.1}$$

dove *a ($1\leq a\leq q$)* e`un intero tale che *a*, $a^2$ , ... , $a^{q-1}$ sono uguali, in un certo ordine, a *1, 2, ... q-1*. Per esempio, con *q=7* , si puo`scegliere *a=3* poiche`*a=3*, $a^2=2$, $a^3=6$, $a^4=4$, $a^5=5$, e $a^6=1$. Tale quantita`*a* prende il nome di *elemento primitivo* del campo finito GF(q).
Allora *X* si puo`pensare come il *logaritmo discreto* riferito alla base *a, mod q* :

$$X=\log_a Y \mod q \qquad 1\leq Y\leq q-1, \tag{4.2}$$

Il calcolo (4.1) di Y da X e`facile, anche per valori molto grandi di q, attraverso il *metodo della quadratura e moltiplicazione (square-and-multiply)* (si veda par.6.2.2.1). Esso richiede al piu`2*$\log_2$q moltiplicazioni. Per esempio per calcolare $a^{53}=a^{32+16+4+1}$, uno puo`dapprima calcolare $a^2$, $a^4=(a^2)^2$, $a^8=(a^4)^2$ , $a^{16}=(a^8)^2$, *e* $a^{32}=(a^{16})^2$ che richiedono 5 moltiplicazioni. Quindi uno puo`moltiplicare a^{32},a^{16},a^4 e *a* insieme, che richiede 3 moltiplicazioni, per un totale di 8 moltiplicazioni (mod q).

Il calcolo (4.2), invece, di X da Y puo`essere molto piu`difficoltoso e per certi, opportunamente scelti, valori di *q* richiede sull"ordine di $q^{1/2}$ operazioni, pur usando il miglior algoritmo di calcolo. Proprio in virtù di questa difficolta`di calcolo, Diffie e Hellman suggeriscono un semplice modo nel quale il logaritmo discreto puo`essere utilizzato per creare chiavi segrete tra coppie di utilizzatori in una rete di comunicazione utilizzando solo messaggi pubblici. La difficoltà di calcolo, infatti, diventa la garanzia di sicurezza del sistema. Il sistema funziona cosí:

1) Si presume che tutti gli utenti del sistema siano a conoscenza di *a* e *q*.
2) Ogni utente, come per esempio *l'utente i*, sceglie a caso un intero X_i, tra 1 e *q*-1 che tiene segreto.Quindi calcola

$$Y_i=a^{Xi} \pmod q$$

e lo sistema in un file pubblico insieme al suo nome e al suo indirizzo, accessibile a tutti gli utenti.
3) Se l'utente *i* e l'utente *j* desiderano comunicare segretamente, l'utente *i* preleva Y_j dal file di *j*, e utilizza la sua quantita`segreta X_i per formare

$$K_{ij} = (Y_j)^{Xi} = (a^{Xj})^{Xi} = a^{Xj\,Xi} \pmod q \qquad (4.3)$$

In modo simile, utente *j* forma K_{ji} . Ma $K_{ij} = K_{ji}$, cosicche`gli utenti *i* e *j* possono usare $K_{ij}$ come chiave segreta in un crittosistema convenzionale. Se un crittanalista nemico e`in grado di risolvere il problema del logaritmo discreto, puo`prelevare $Y_i$ e $Y_j$ dal file pubblico, risolvere $Xi = \log_a Y_i$ , e quindi formare $K_{ij}$ allo stesso modo dell'utente *i*. Se, dunque, i *log modulo q* sono facili da calcolare, il sistema puo`essere facilmente violato. Per il momento non si hanno prove del contrario (cioe`che il sistema e`sicuro se i *log modulo q* sono complicati da calcolare), e tuttavia non si vede altro modo di calcolare K_{ij} da Y_i e Y_j senza prima aver ottenuto o Xi o Xj.

4.4 Crittosistemi a chiave pubblica

La ragione per cui la chiave deve essere protetta nei sistemi crittografici convenzionali e`che le funzioni di cifratura e decifratura sono inseparabili. Chiunque abbia accesso alla chiave per cifrare un messaggio, automaticamente e`anche in grado di decifrarlo.

Se la capacita`di effettuare la cifratura e la decifratura vengono distinte, allora non c'e`piu`alcuna ragione di mantenere segreta la chiave necessaria alla cifratura, poiche`essa non puo`poi essere usata per la decifratura.
Questo e`il principio su cui si basano i crittosistemi a chiave pubblica; essi sono progettati in modo tale che risulti facile generare una coppia random di chiavi inverse, rispettivamente *E* per la cifratura e *D* per la decifratura, sia facile operare con *E* e *D* , ma sia computazionalmente impossibile calcolare *D* da *E*.

Formalmente, dunque, un crittosistema a chiave pubblica e`costituito da una coppia di famiglie $\{E_k\}_{k\in\{K\}}$ e $\{D_k\}_{k\in\{K\}}$ di algoritmi che rappresentano trasformazioni invertibili

$$E_k : \{M\} \rightarrow \{M\} \qquad (4.4)$$

$$D_k : \{M\} \rightarrow \{M\} \qquad (4.5)$$

su uno spazio dei messaggi *{M}* finito, tali che soddisfano le seguenti condizioni (4.6) :

1) Per ogni $k \in \{K\}$, D_k e`l'inversa di E_k . Questo significa che per ogni *k* e per ogni *M*, $D_k\,E_k\,(M)=M$.
2) Per ogni $k \in \{K\}$ e $M \in \{M\}$, i valori $E_k(M)$ e $D_k(M)$ sono facili da calcolare.
3) Per quasi tutti i $k \in \{K\}$, non e` "computazionalmente"possibile derivare da E_k un qualunque algoritmo equivalente a D_k .
4) Per ogni $k \in \{K\}$ e`possibile generare la coppia inversa E_k e D_k da *K*.

La proprieta`3) permette ad un utilizzatore di rendere pubblica la chiave di cifratura $E_k$ senza compromettere la sicurezza della propria chiave di decifratura $D_k$ .Infatti, rivelando $E_k$ , egli rivela un metodo davvero *inefficiente* per calcolare $D_k(C)$ (dove *C* e`il crittogramma): tale metodo consiste nel testare tutti i possibili messaggi *M* fino a trovarne uno tale che sia verificata l'uguaglianza $E_k(M)=C$. Se, pero`, la proprieta`3) e`soddisfatta il numero di tali messaggi da testare puo`essere cosi`grande da rendere impraticabile questo approccio.

Una funzione E che ubbidisce alla 1) e alla 3) e`una cosidetta **funzione unidirezionale (one-way function)**. Nel capitolo 6, illustrando un esempio di sistema a chiave pubblica, analizzeremo piu`da vicino questa classe di funzioni (si veda par. [6.2.1]; qui

basti dire che esse sono chiamate *unidirezionali (one-way)* poiche`sono facili da calcolare in una direzione (trasformazione diretta), ma (apparentemente) molto difficoltose da calcolarsi nella direzione opposta (trasformazione inversa); sono, invece, chiamate funzioni *a trabocchetto (trap-door)* poiche`le funzioni inverse sono, in realta`, facili da calcolarsi a patto pero` che si conoscano alcune informazioni private (quelle che potremmo definire appunto i "trabocchetti").

Vediamo adesso, in base a queste loro caratteristiche, quale tipo di risposta sono in grado di dare i crittosistemi a chiave pubblica ai problemi lasciati aperti dai sistemi a chiave segreta.

4.4.1 Segretezza e problema della distribuzione della chiave

Un sistema del tipo illustrato qui di sopra garantisce un ottima segretezza e, al contempo, semplifica notevolmente il problema della distribuzione della chiave. Ogni utilizzatore genera una coppia di trasformazioni inverse, E e D, al suo terminale.Tiene, quindi, segreta la trasformazione di decifratura D, mentre rende pubblica la trasformazione di cifratura E inserendola in una **directory pubblica** con il proprio nome e indirizzo. Chiunque puo`cifrare messaggi e inviarli all'utilizzatore, ma nessuno puo`decifrare i messaggi destinati a lui. Per questa caratteristica, un crittosistema a chiave pubblica puo`essere visto come una **cifratura ad accesso multiplo (multiple access cipher)**.

Per meglio comprendere il meccanismo con cui due utenti del sistema possono scambiarsi un messaggio segreto, consideriamo il seguente

ESEMPIO

Supponiamo che **A** e **B** siano due utenti qualsiasi del sistema (convenzionalmente identificati dai crittografi con i nomi di **A**lice e **B**ob) e distinguiamo le loro procedure di cifratura e decifratura con i rispettivi pedici: $E_{\mathbf{A}}$, $D_{\mathbf{A}}$, $E_{\mathbf{B}}$, $D_{\mathbf{B}}$ (omettiamo in tal caso, per non confondere le idee, il pedice *K* utilizzato nella notazione precedente per evidenziare che ciascuna procedura e`diversa a seconda del diverso valore della chiave K). Le operazioni che devono essere eseguite affinche`B possa mandare ad A un messaggio segreto possono essere cosi`schematizzate:

Utente B

1) preleva dalla directory pubblica la procedura di cifratura E_A di A;
2) forma il crittogramma $E_A(M)$ del messaggio originale M da inviare;
3) invia, sul canale pubblico, ad A il messaggio cifrato $E_A(M)$.

Utente A

1) Riceve il messaggio cifrato $E_A(M)$ da B;
2) Lo decifra utilizzando la propria chiave segreta D_A;
   in tal modo, calcolando $D_A(E_A(M))=M$ ottiene il messaggio originale.

Come si puo`osservare, **non e`necessaria alcuna transazione privata** su di un canale sicuro (come invece avveniva nei sistemi a chiave segreta), per stabilire una comunicazione segreta. Il solo "setup"che viene richiesto e`che ogni utente che desideri ricevere comunicazioni riservate sistemi il suo algoritmo di cifratura nella directory pubblica.Questo significa indubbiamente una riduzione notevole di costi sia in termini di tempo che di denaro, oltre che ovviamente la possibilita`di estendere ad un numero molto elevato di utilizzatori la capacita`di comunicare segretamente gli uni con gli altri senza alun pre-accordo.

Un crittosistema a chiave pubblica puo`essere impiegato, d'altra parte, come "bootstrap"in uno schema di cifratura standard, con il fine, ancora una volta di eliminare il canale sicuro che e`molto costoso. Una volta stabilita una comunicazione affidabile, il primo messaggio trasmesso puo`essere la chiave da utilizzare nello schema standard per cifrare tutti i successivi messaggi; questo e`desiderabile, quando lo schema standard offre prestazioni, in termini di velocita`di cifratura, superiori a quelle del crittosistema a chiave pubblica.

2) Autenticita`e problema della controversia

Un crittosistema a chiave pubblica e`in grado non solo di garantire l'autenticita`del messaggio, ma offre anche la capacita` di "firmare"un messaggio e fornisce cosi` la possibilita` a chi `lo riceve di dimostrare che tale messaggio e`stato inviato proprio dal mittente che lo ha "firmato" (ossia l'ha scritto e mandato).

Sebbene la distinzione possa apparire sottile, tuttavia e`sostanziale; infatti, la seconda esigenza e`molto piu`onerosa della prìma. Nel problema della **semplice autenticita`** il ricevente deve poter verificare che il messaggio derivi dal mittente e non sia stato modificato o sostitiuto dal nemico; tale verifica ha uno scopo puramente personale, ed e`mirata a fugare dubbi circa la veridicita`delle informazioni trasmesse. Nel secondo caso **(problema della controversia)**, invece, si vuole ottenere una prova, legalmente valida, da poter esibire in tribunale in caso di controversia; il ricevente deve poter dimostrare ad un giudice che un dato messaggio e`stato inviato, cosi`com'e`, senza aver subito alcun tipo di alterazione, dal legittimo mittente (in qualche modo identificabile) anche se costui afferma il contrario.

I crittosistemi a chiave pubblica, fornendo dunque la possibilita`di realizzare **"firme elettroniche" (signatures)** dei messaggi inviati via e-mail, risultano molto piu`potenti e affidabili dei sistemi convenzionali. La loro capacita`di realizzare firme elettroniche nasce dal fatto che essi sono implementati con funzioni unidirezionali le cui caratteristiche si prestano, ottimamente, a convertire in forma digitale i requisiti che una firma deve avere :

1) essere riconoscibile come autentica da chiunque;
2) non essere riproducibile da nessun altro, se non dal firmatario.

Sotto il profilo crittografico, tali requisiti si traducono nella necessita` che una **firma elettronica sia legata sia al messaggio (message-dependent) che al firmatario (signer-dependent)**; infatti, il ricevente potrebbe modificare il messaggio prima di mostrare al giudice la coppia messaggio-firma oppure potrebbe attaccare la firma a qualche altro messaggio, dal momento che e`impossibile rilevare un "cutting and pasting"elettronico.

Da un punto di vista matematico, per poter realizzare una firma elettronica, la coppia di trasformazioni E_k e D_k deve soddisfare ad un'ulteriore condizione (che indicheremo con il numero 5) nelle (4.6) per conservare l'ordine di partenza), oltre quelle gia`indicate in precedenza:

5) Per ogni $k \in \{K\}$, E_k e`l'inversa di $D_k$ . Cioe`per ogni k e per ogni M, $E_k D_k (M)=M$.

Essa garantisce che **l'algoritmo di decifratura possa essere applicato a messaggi non cifrati e che un'operazione di decifratura di un messaggio M, seguita da un'operazione di cifratura restituisca come risultato ancora M.**

Una funzione E che soddisfa la proprieta` 5), oltre che la 1)-3) delle (4.6), e`chiamata **permutazione unidirezionale (one-way function)**. Vediamo adesso come puo`un utente mandare un messaggio "firmato" ad un altro utente.

Per illustrare la sequenza delle operazioni facciamo riferimento ancora una volta all'esempio riportato in precedenza e supponiamo che sia l'utente B a voler inviare il messaggio all'utente A; si avra`che:

Utente B

1) Calcola la propria "firma" S dal messaggio M usando la propria procedura di decifratura D_B:
$S = D_B(M)$; (Tale operazione e`consentita in virtu`della proprieta`5) di cui sopra).

2) Cifra il nuovo testo S ottenuto con la procedura di cifratura E_A di A; (questa operazione viene effettuata solo se B vuole mantenere segreto il messaggio inviato ad A, altrimenti puo`essere saltata).
3) Invia il risultato $E_A(S)$ ad A.

Utente A

1) Decifra il testo cifrato $E_A(S)$ inviatogli, utilizzando la procedura di decifratura D_A per ricavare S (tale operazione deve essere omessa se l'utente B non ha eseguito il proprio step 2)); cosi`facendo, e`in grado di conoscere il presunto mittente della firma.
Estrae il messaggio M da S con la procedura di cifratura E_B del mittente, disponibile nella directory pubblica; si ha che:

$$M = E_B (S).$$

L'utente A possiede, ora, una coppia messaggio-firma (M,S) con proprieta`simili a quelle di un documento cartaceo firmato. Infatti B non potra`piu`negare di aver inviato ad A questo messaggio, poiche`nessun altro puo`aver creato $S=D_B(M)$, (essendo $D_B$ noto soltanto a B) e, d'altra parte, A potra`convincere un giudice che $E_B(S) = M$, avendo cosi`una prova che B ha firmato il documento.
Inoltre, l'utente A non puo`modificare il messaggio originale M e crearne uno nuovo M' poiche`corrispondentemente dovrebbe poter creare anche una nuova firma $S'=D_B(M')$, il che non e`possibile.

Concludiamo questa analisi dei crittosistemi a chiavi pubblica impiegati per la realizzazione di firme elettroniche, con alcune considerazioni di carattere piu`pratico, ma ugualmente importanti.
Quando la cifratura e`impiegata per le firme elettroniche, e`importante che il componente che realizza la cifratura (*encryption device*) non sia installato tra il computer (o il terminale) e il canale di comunicazione (come avevamo indicato negli schemi riportati), poiche`un messaggio puo`dover essere cifrato successivamente con chiavi diverse. E`forse piu`naturale vedere *l'encryption device* come una *"hardware subroutine"* che puo`essere eseguita se e`necessario.

Nella trattazione precedente, si e`assunto che ogni utente possa accedere alla directory pubblica con tutta tranquillita`, fiducioso delle informazioni che va a prelevare. In una rete di computer questo puo`essere difficoltoso; infatti, un

"intruso"potrebbe creare dei messaggi falsi e spacciarli come messaggi della directory pubblica. L'utente deve avere la sicurezza di ottenere la procedura di cifratura del proprio corrispondente, e non la procedura di cifratura dell'intruso.

Questo rischio puo`essere cancellato se la directory pubblica "firma", per cosi`dire, a sua volta, ogni messaggio che essa invia all'utente. L'utente puo`verificare la firma con l'algoritmo di cifratura E_{PF} della directory pubblica.

Il problema di consultare E_{PF} stessa nella directory pubblica e`evitato fornendo ad ogni utilizzatore una descrizione della E_{PF} al momento in cui costui, per la prima volta, deposita, di persona, la propria procedura di cifratura pubblica al responsabile della directory pubblica e si collega al crittosistema pubblico. Un'altra soluzione, potrebbe essere, invece, quella di eliminare la directory pubblica e fornire a ciascun utente una sorta di elenco telefonico con tutte le chiavi di cifratura degli utilizzatori del sistema.

4.5 Il crittosistema a chiave pubblica RSA

L'RSA rappresenta il primo esempio di crittosistema a chiave pubblica. Esso fu presentato nel 1978 da alcuni ricercatori del M.I.T., R.L.Rivest, A. Shamir, e Adleman (da cui il nome RSA dell'algoritmo) [ref. 12] e [ref. 1, 17,18] come prima proposta di una possibile funzione unidirezionale. Sebbene, infatti, Diffie e Hellman avessero elaborato il concetto di crittosistema a chiave pubblica e quello di funzione unidirezionale, tuttavia non ne avevano presentato una pratica implementazione.

L'RSA e`molto semplice, ma richiede, per essere compreso a fondo, la conoscenza di alcune idee di base della teoria dei numeri per cui rimandiamo al par. 2.2 e un approfondimento del concetto di funzione unidirezionale di cui ci occuperemo nel prossimo paragrafo, prima di passare ad analizzare l'algoritmo vero e proprio.

4.5.1 Le funzioni unidirezionali e le funzioni unidirezionali

Abbiamo visto nel paragrafo 4.2 che nei sistemi a chiave pubblica gli algoritmi di cifratura e decifratura sono implementati utilizzando una classe di funzioni , dette funzioni unidirezionali, che, per le loro caratteristiche, si prestano ottimamente a soddisfare l'esigenza del sistema crittografico di rendere le due operazioni di cifratura e decifratura indipendenti l'una dall'altra per far si`che la rottura del cifrario sulla base delle semplici informazioni che si possono ricavare dal solo crittogramma sia il piu`difficoltosa possibile (se non impossibile) .

Il concetto di funzione unidirezionale e`posteriore a quello di funzione unidirezionale pura e semplice ed e`da esso mutuato; occorre, dunque, per capire a fondo l'uno fare riferimento prima all'altro [ref. 6]. Una funzione *f* e`una funzione unidirezionale (one-way function) se, per ogni argomento *x* nel dominio di *f* , e`facile calcolare il corrispondente valore *f(x)*, mentre , per quasi tutti gli *y* nel range di *f*, e`computazionalmente impossibile risolvere l'equazione *y=f(x)* per qualunque opportuno argomento *x*.

In questo modo, viene definita una funzione che e`non invertibile da un punto di vista computazionale, ma la cui *non-invertibilita`* e`completamente diversa da quella normalmente incontrata in matematica. Una funzione *f* e`normalmente definita *"non-invertibile"* quando l'inversa di un punto *y* e` *non unica*, cioe`esistono due distinti punti x_1 e x_2 tali che $f(x_1) = y = f(x_2)$.Qui, non e`richiesto questo tipo di difficolta`d'inversione; piuttosto, dato un valore *y* e una funzione nota *f* ,deve essere davvero difficoltoso calcolare un *x* tale che $f(x)=y$.

Quindi, se *f* e`non invertibile nel senso usuale, paradossalmente, l'operazione di trovare una controimmagine *x* diventa addirittura piu`facile; se poi, al limite, $f(x)=y_0$ per tutti gli *x* del dominio, allora il range di *f* e`$\{y_0\}$ ,ed e`possibile ricavare un qualunque *x* come $f^{-1}(y_0)$. I polinomi offrono un esempio elementare di funzione unidirezionale : è più difficile cercare una radice x_0 dell'equazione polinomiale $p(x)=y$ che valutare il polinomio $p(x)$ per $x= x_0$.

Le funzioni unidirezionali trovano facile applicazione in pratica ; il caso più classico è quello del cosidetto problema della "login" nei multiuser computer systems . In breve :ogni utente possiede una password PW che gli consente di accedere al sistema ; la prima volta che egli inserisce tale password, il sistema calcola una funzione f(PW) che viene memorizzata in una apposita directory detta "password

directory" ;alle successive login, il computer calcola f(X), dove X è la corrente password digitata, e la confronta con f(PW). Se il match ha esito positivo, allora l'utente è riconosciuto come autentico. La funzione f deve essere facilmente calcolabile (il suo tempo di calcolo deve essere molto breve), ma assai complicata da invertire in modo tale che chi acceda illegalmente alla password directory non possa ricavare PW da f(PW) ; quindi, deve avere le caratteristiche di una one-way function. Per altri dettagli su questo problema rimandiamo al [ref. 6].

Una funzione unidirezionale (one-way function) conserva la caratteristica essenziale della funzione unidirezionale (cioè la non invertibilità computazionale), ma sotto opportune ipotesi tale condizione può essere rimossa. Si tratta, infatti, di una funzione che non è realmente unidirezionale,in quanto possiede una inversa che può essere facilmente calcolata. Tuttavia, dato un algoritmo per calcolare la funzione diretta, è computazionalmente impossibile cercare l'inversa (seppur facilmente calcolabile) a partire da esso. Soltanto attraverso la conoscenza di certe informazioni-trabocchetto (trap-door information) uno può facilmente calcolare l'inversa.

Una funzione unidirezionale, viene definita, dunque, come una famiglia di funzioni invertibili f_z, indicizzate da un parametro *z*, tali che, dato *z*, è facile cercare degli algoritmi E_z *e* D_z che calcolano facilmente $f_z(x)$ *e* $f_z^{-1}(y)$ per tutti gli *x* e gli *y* nel dominio e nel range, rispettivamente, di f_z ; ma per quasi tutti gli *z* e per quasi tutti gli *y* nel range di f_z, è computazionalmente impossibile calcolare $f_z^{-1}(y)$ anche quando uno conosce E_z.

L' RSA, come vedremo, utilizza una di queste funzioni e più precisamente l'esponenziale discreto :

$$f_z(x)=x^e.$$

4.5.2 L'algoritmo

Per cifrare un messaggio M con il metodo RSA, si utilizza una *chiave pubblica di cifratura (e,n),* e si procede come segue.(Qui *e* e *n* sono una coppia di numeri interi positivi.) Innanzi tutto, si rappresenta il messaggio sotto forma di un intero tra 0 e *n* -1. Se il messaggio e`lungo, lo si rompe in una serie di blocchi e si rappresenta ciascun blocco con un intero. Si puo`utilizzare a tal fine una qualunque rappresentazione standard. Questa operazione non serve per cifrare il messaggio, ma semplicemente per metterlo in una forma numerica necessaria alla cifratura.

Quindi, si cifra il messaggio elevandolo alla *e* -sima potenza modulo *n*; cio`significa che il risultato (ossia il testo cifrato C) e`il resto che si ottiene quando M^e viene diviso per *n*. Per decifrare il crittogramma, lo si eleva ad un'altra potenza *d*, nuovamente modulo *n*. Gli algoritmi di cifratura e decifratura E e D sono dunque:

$$C= E(M)=M^e \pmod n, \text{ per un messaggio M}$$

$$D(C)=C^d \pmod n, \text{ per un crittogramma C.}$$

Nota: Utilizziamo qui, per indicare i processi di cifratura e decifratura, la notazione introdotta per i crittosistemi a chiave pubblica nel cap. 4 al fine di ricondurci piu`facilmente dal caso particolare a quello generale. Per chiarezza, occorre evidenziare, pero`, che si potrebbe fare riferimento anche alla notazione piu`matematica del paragrafo 6.2.1, dove, parlando di funzioni e volendo sottolineare la caratteristica di reciprocita`, si erano indicate la cifratura e la decifratura rispettivamente con $f_Z()$ e $f_Z^{-1}()$. Per essere precisi, *E* e *D* rappresentano gli algoritmi per il calcolo di $f_Z()$ e $f_Z^{-1}()$, ma si tratta solo di una questione formale; il concetto che sta alla base e`identico e risulta quindi equivalente utilizzare una notazione o l'altra.

Si puo`notare che la cifratura non accresce la dimensione di un messaggio; tramite l'aritmetica modulo *n* sia messaggio che testo cifrato sono numeri interi nel range 0:*n*-1.

La *chiave di cifratura* e`rappresentata dalla coppia di interi (*e,n*); analogamente, *la chiave di decifratura* dalla coppia (*d,n*) (Fig.6.4). Ogni utilizzatore rende pubblica la sua chiave di cifratura e tiene segreta la

corrispondente chiave privata di decifratura (le quantita`sopra indicate, in realta`dovrebbero avere tutte un pedice, come ad esempio n_A, e_A, e d_A, poiche` ogni utente ha il proprio set; qui lo omettiamo considerando un set tipico).

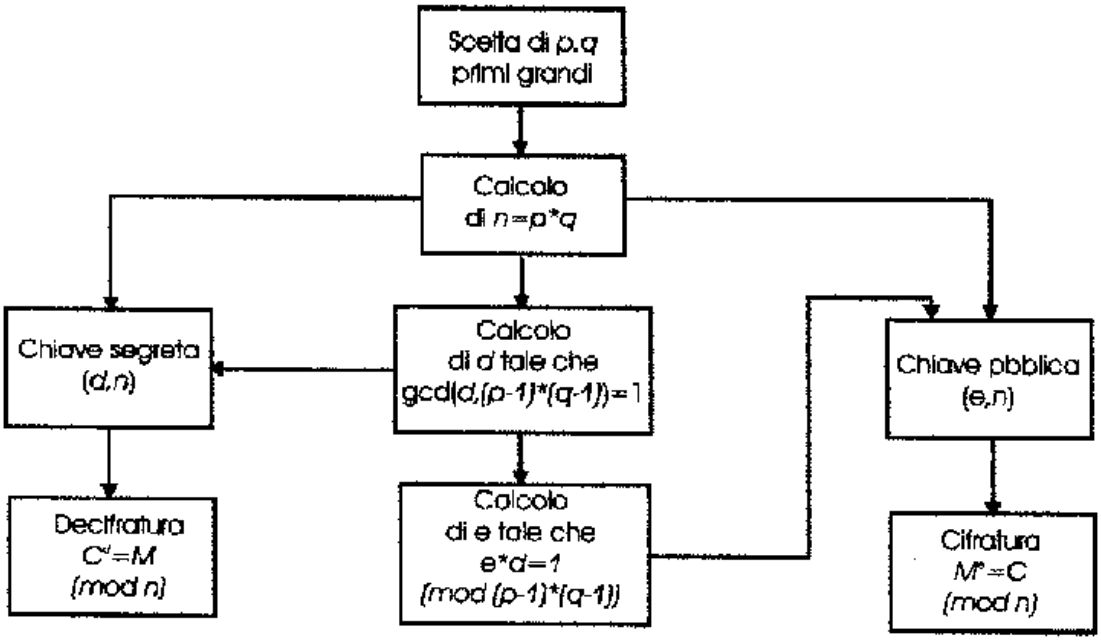


Fig.6.4 :Diagramma a blocchi dell'algoritmo RSA

Ovviamente esiste una relazione tra la chiave di cifratura e quella di decifratura e delle regole per poterle scegliere in maniera opportuna. Il procedimento è illustrato in Fig.6.4.

1) Il primo passo e`calcolare *n* come il prodotto di due numeri primi *p* e *q*:

$$n=p*q$$

Tali numeri primi sono numeri primi "random" molto grandi; nel cap.2 è indicata una procedura per sceglierli in maniera adeguata. Gli autori del RSA raccomandano di utilizzare numeri primi *p* e *q* con 100 cifre (decimali) cosichè *n* abbia 200 cifre. Tale dimensione si ritiene comunque oggi forse già limitativa viste le potenze di calcolo disponibili.
Il fato che questi numeri debbano essere grandi, si spiega tenendo presente che, rispettando questa condizione, risulta computazionalmente irrealizzabile per chiunque fattorizzare $n=p*q$ e, quindi, sebbene il numero *n* debba essere reso di dominio pubblico, tuttavia i fattori *p* e *q* hanno altissima probabilita`di rimanere incogniti a chi non li ha scelti in partenza, a causa dell'enorme difficolta`che s'incontra nel fattorizzare *n*. Questo fa si`che, a sua volta, rimanga incognito anche il modo in cui la grandezza *d* puo`essere ricavata da *e*, offrendo al sistema ottime garanzie di sicurezza (infatti, pur conoscendo la chiave di cifratura (*n*,*e*) e`molto improbabile poter ricavare quella di decifratura (*n*,*d*) senza prima conoscere *p* e *q*; si puo`dunque identificare nell'insieme {*p*,*q*,*e*} il parametro *z* cui si faceva riferimento nel paragrafo 5.1, ossia *z*={*p*,*q*,*e*} rappresenta il cosidetto "trabocchetto" dalla cui conoscenza dipende la possibilita`di trovare facilmente l'algoritmo *D* per calcolare f_Z^{-1}).

2) Si sceglie, quindi, l'intero *d* come un numero intero grande, random e primo rispetto a *(p-1)*(q-1)*, tale cioe`che soddisfi:

$$gcd(d,(p-1)*(q-1))=1$$

3) L'intero e viene calcolato, infine, da p, q, e d come l'inverso di d, modulo (p-1)*(q-1). Si ha, dunque, che :

$$e*d=1 \pmod{(p-1)*(q-1)}. \qquad (6.10)$$

La correttezza di tale operazione e`garantita dalle proprieta`enunciate nel paragrafo precedente. Infatti, scegliere *d* primo rispetto a (*p*-1)*(*q*-1) e`equivalente, sulla base della proprietà (2.2), a dire che *d* e`primo rispetto a φ(*n*); in virtu`delle caratteristiche di φ(*n*) e del teorema (2.4), si puo`concludere che *d* ha un inverso *e* nella cerchia degli interi modulo φ(*n*), cioe`:

$$e*d=1 \pmod{\varphi(n)} \qquad (6.11)$$

Ma la (6.11) e`semplicemente una riscrittura della (6.10). Inoltre, se *e* e *d* vengono scelti come indicato sopra, si puo`provare che *E* e *D* sono permutazioni inverse, come richiesto dalla teoria dei crittosistemi a chiave pubblica. Possiamo scrivere infatti che :

$$D(E(M)) = (E(M))^d = (M^e)^d = M^{ed} \pmod{n} \qquad (6.12)$$

$$E(D(M)) = (D(M))^e = (M^d)^e = M^{ed} \pmod{n}$$

Inoltre, la (6.11) e`equivalente, nell'ordinaria aritmetica dei numeri interi, ad imporre che:

$$e*d = k\,\varphi(n)+1 \quad \text{(per interi } k\text{)}. \qquad (6.13)$$

Dalla (6.12) e (6.13) otteniamo

$$M^{ed} = M^{k\varphi(n)+1} \pmod{n}$$

$$= M\,(M^{\varphi(n)})^k \pmod{n}$$

$$= M \pmod{n}$$

dove all'ultimo passaggio è stato usato il teorema di Eulero (2.3).

Questa uguaglianza mostra che elevare un numero M alla potenza *d* (modulo *n*) e`davvero l'inverso di elevare lo stesso numero M alla potenza *e* (modulo *n*): quindi, *E* e *D* rappresentano due permutazioni inverse,come volevasi dimostrare. Consideriamo adesso piu`da vicino un aspetto importante dell'algoritmo.

4.5.2.1 Un metodo per cifrare e decifrare efficientemente

Il calcolo di $M^e \pmod{n}$ puo`essere eseguito utilizzando la seguente procedura e richiede in tal caso al piu`$2*\log_2(e)$ moltiplicazioni e $2*\log_2(e)$ divisioni (la decifratura puo`essere realizzata in maniera analoga usando *d* al posto di *e*):

Step 1. Si scrive l'esponente *e* in forma binaria $e_k e_{k-1} \ldots e_1 e_0$ dove e_k rappresenta il bit piu`significativo e e_0 quello meno significativo.
Step 2. Si setta la variabile C a 1.
Step 3. Si ripetono gli steps 3a e 3b per $i = k, k-1, \ldots, 0$:
Step 3a. Si setta C al resto di C^2/n.
Step 3b. Se $e_i = 1$, allora si setta C al resto di (C*M)/*n*
Step 4. Fine. C rappresenta la forma cifrata di M.

Questa procedura e`chiamata *"elevamento a potenza attraverso ripetute quadrature e moltiplicazioni"*. Si puo`capire facilmente facendo alcune semplici considerazioni. Un qualunque numero *e* puo`essere pensato come la somma di potenze di 2.

$$\text{Es. } e=17= 16+1=2^4+2^0$$

Tale scomposizione puo`essere ottenuta dalla rappresentazione binaria del numero *e*; infatti, se *e*=17=10001 in binario, per definizione di notazione binaria, si ha che :

$$e = 10001_2 = 1*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 17_{10}$$

Si nota che le potenze di 2 da cui e`composto il nostro numero decimale sono quelle che si hanno in corrispondenza degli 1 nella sua rappresentazione binaria.

In generale,qiundi, data la rappresentazione binaria $e_k e_{k-1} ... e_1 e_0$ del numero *e*, se la generica cifra e_i=1 si avra` un termine 2^i nella scomposizione in potenze di 2 del numero *e*.

A questo punto,volendo elevare un qualunque numero M alla potenza *e* (mod *n),* si potra`utilizzare tale scomposizione e scrivere ad esempio che:

$$\begin{aligned} M^e \pmod n &= M^{17} \pmod n \qquad (\text{se } e=17) \\ &= M^{16+1} \pmod n \\ &= M^{16} \pmod n * M^1 \pmod n \end{aligned} \qquad (6.14)$$

La nostra potenza M^e (mod *n*) e`ricondotta al prodotto di fattori in ciascuno dei quali compare il numero M elevato ad una diversa potenza di 2; tenendo conto di quanto detto sopra, si puo`facilmente osservare che tali fattori saranno in corrispondenza degli 1 della notazione binaria del numero *e* , e in particolare in corrispondenza di e_i=1 si avra` un fattore $M^{2^i}$ . Ognuno di tali fattori puo`essere calcolato semplicemente elevando M al quadrato un numero di volte pari a *i* .

Facendo riferimento a (6.14) si avra`ad esempio:

$$\begin{aligned} M^{16} \pmod n * M^1 \pmod n &= M^{2^4} \cdot M^{2^0} \\ &= (((M^2)^2)^2)^2 * M \pmod n \end{aligned}$$

Queste operazioni sono realizzate dagli step 3a e 3b della procedura indicata;si parte dalla cifra piu`significativa $e_k$ perche`ad essa corrisponde la potenza di 2 piu`grande ed e`quella, dunque, che richiede il numero maggiore di quadrature (pari a *k*); essendo l'operazione di quadratura inserita in un loop, inizializzando il numero di passi del loop a *k* e procedendo a ritroso si e` sicuri di eseguirle tutte.

Questa procedura per il calcolo di potenze si colloca a meta`strada tra quelle considerate buone e le migliori; ne esistono sicuramente di piu`efficienti. Le prestazioni sono comunque di tutto rispetto: un computer ad alta velocita`puo`cifrare un messaggio M a 200 cifre in pochi secondi; inoltre, c'e`il vantaggio che, essendo cifratura e decifratura identiche, si puo`realizzare una semplice implementazione.

4.5.3 RSA: sicurezza e applicazioni

Non esiste nessuna tecnica per provare se uno schema di cifratura e`sicuro; l'unico test utilizzabile e`vedere se qualcuno riesce a trovare un modo per violarlo. Nel caso del DES (Data Encryption Standard) tale modo e`stato certificato, ma si è altresì visto che esso richiede un'enorme quantità di tempo e, questo ha portato a definire il DES sicuro per le normali applicazioni.

Abbiamo gia`accennato al fatto che l' RSA, invece, fonda la sua sicurezza sulla difficolta`a fattorizzare *n*; un qualunque crittanalista nemico che sia in grado di fattorizzare *n* e`automaticamente in grado di violare il sistema di cifratura poiche`la conoscenza dei fattori *p* e *q* di *n* gli consente di calcolare □*n*) e quindi *d* .La fattorizzazione, però, e`ben lontana dall'essere un'operazione di facile realizzazione.

Oggi esistono diversi algoritmi che effettuano la fattorizzazione di grandi numeri, il piu`veloce dei quali e`dovuto a Richard Schroeppel, e il segreto per garantire l'inviolabilità del RSA sta, quindi, nel fatto di rendere tale fattorizzazione un'operazione *computazionalmente irrealizzabile* dal punto di vista del tempo. Questo e`possibile utilizzando un numero *n* di 200 cifre decimali; in tal caso infatti, utilizzando il metodo di

Schroeppel, si è visto che ci vorrebbe circa un miliardo di anni per fattorizzare *n*, a dispetto del caso invece in cui *n* ha soltanto 50 cifre decimali che richiederebbe "soltanto" 104 giorni.

Naturalmente e`possibile utilizzare lunghezze piu`corte o piu`lunghe di *n* a seconda dell'importanza relativa della velocità di cifratura e della sicurezza nell'applicazione del sistema; un *n* a 80 cifre fornisce una moderata sicurezza rispetto alle odierne tecnologie; usando 200 cifre si fornisce un margine di sicurezza anche rispetto a sviluppi futuri.

Sono stati analizzati anche approcci alternativi alla fattorizzazione di *n* per rompere il sistema, (per la cui trattazione completa rimandiamo a [ref. 12]):

- calcolo di $\phi(n)$ senza fattorizzare *n;*
- calcolo di *d* senza fattorizzare *n* o calcolare $\phi(n)$;
- calcolo di *D* in altri modi;

ma tutti hanno mostrato una difficolta`ad essere applicati pari a quella della fattorizzazione.

Per ciò che concerne l'applicazione del RSA possiamo dire che ci sono, attualmente, dei chips VLSI che possono implementare la cifratura e la decifratura RSA ad una "data rate"di pochi Kilobits al secondo; è stato mostrato, altresì, che data-rates significativamente piu`alte non si potranno mai ottenere.

Per alcune applicazioni crittografiche queste data-rates sono troppo basse; in tali casi, il crittosistema a chiave pubblica RSA può essere impiegato per effettuare la distribuzione di chiavi che saranno utilizzate, poi, in cifrari a chiave segreta a piu`alta velocita`, quali il DES o alcuni cifrari a stringa, oppure può essere utilizzato per l'autenticazione nella sua versione "digital signature".

5
I cifrari a stringa

5.1 Struttura e principi generali

I cifrari a stringa o stream ciphers sono sistemi crittografici che operano una trasformazione *tempovariante* sulle singole cifre di testo in chiaro; questo significa che la trasformazione di cifratura cambia da carattere a carattere e ogni singolo carattere di input viene cifrato in un carattere di output mai allo stesso modo, ma in una maniera che dipende dallo stato interno del sistema.

Utilizzando un approccio di teoria dei sistemi e ammettendo che x_i, y_i, e s_i,denotino, rispettivamente, la cifra di testo in chiaro, la cifra di testo cifrato e lo stato interno del sistema all'istante *i*, un generico cifratore a stringa (stream encryptor) è descritto dalle equazioni:

$$s_{i+1} = F(k,s_i,x_i) \qquad (5.1)$$

$$y_i = f(k,s_i,x_i) \qquad (5.2)$$

dove *F* è la funzione dello stato successivo, *f* è la funzione di uscita e *k* la chiave scelta in accordo ad una distribuzione di probabilità P_k.
La (5.1) e la (5.2) ci dicono che la trasformazione crittografica è tempovariante poichè dipende dallo stato del sistema che evolve nel tempo: questo significa, in pratica, che uno stesso carattere può essere cifrato in maniera diversa se si presenta in ingresso al sistema in istanti diversi (vedi anche par. 2.3). I più popolari stream-ciphers in uso oggi sono i cosidetti binary additive stream ciphers per i quali la (5.2) può essere riscritta nella forma

$$y_i = x_i \oplus z_i \qquad i=1, \ldots ,M \qquad (5.3)$$

dove y_i ,x_i , z_i sono tutte cifre binarie e l'insieme $\{ z_i = f(k,s_i) : i \geq 1\}$ rappresenta una sequenza binaria, denominata running key o keystream, di lunghezza *M* molto maggiore della lunghezza *N* della chiave *k*.

In pratica, in questi sistemi gli *N* bit della chiave *k* sono mandati in ingresso ad un algoritmo chiamato generatore pseudorandom di bits (pseudorandom bit generator) per creare una lunga sequenza binaria, denominata, come abbiamo visto, *keystream* di lunghezza *M>>N*.

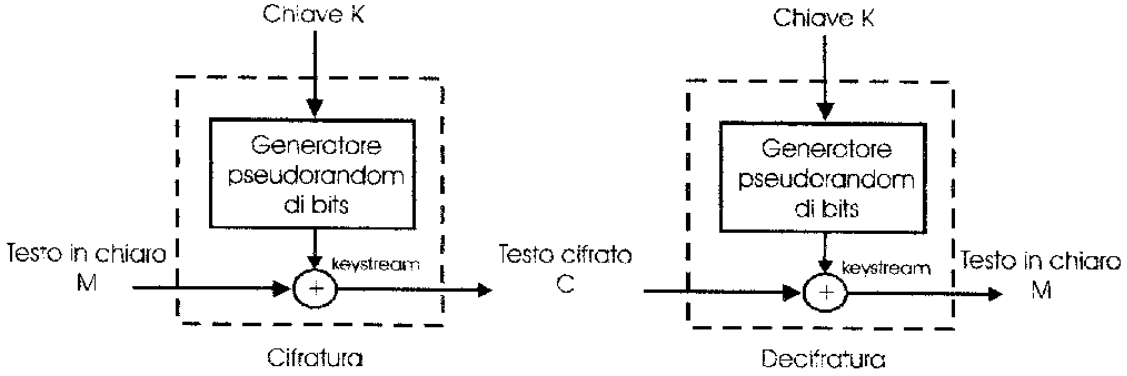


Fig. 5.1 : *Schema di un cifrario a stringa additivo binario (binary additive stream cipher)*

Questa sequenza viene, quindi, sommata modulo 2, tramite un'operazione di XOR, con le cifre del testo in chiaro per produrre il testo cifrato. La fig. 5.1 illustra questo principio. Dal momento che addizione e sottrazione modulo 2 coincidono, si ricava facilmente dalla (5.3) la relazione

$$x_i = y_i \oplus z_i \qquad i=1, \ldots, M \tag{5.4}$$

che mostra (insieme alla (5.3)) che cifratura e decifratura possono essere ottenute da componenti identici.
E' da rilevare, inoltre, che per sistemi di questo tipo un singolo bit di testo in chiaro condiziona un solo bit di testo cifrato (il che rappresenta la peggiore *diffusione di testo in chiaro* che si può desiderare), ma nel contempo, ogni bit della chiave segreta *k* può influenzare diversi bits di testo cifrato, e quindi la *diffusione della chiave* risulta buona.

Il principio che sta alla base di un *binary additive stream cipher* è lo stesso del sistema *binary one time pad* ; infatti, il binary addittive stream cipher è nato come pratica realizzazione (e approssimazione) del binary one time pad, per ovviare al problema della dimensione della chiave.

Nel one-time pad il crittogramma è ottenuto da un'operazione di XOR tra il messaggio originale *M* e la chiave segreta *k* rappresentata da una sequenza random della stessa lunghezza del messaggio.
Se la chiave *k* è prodotta da una sorgente binaria simmetrica tale che la probabilità, per ogni segnale di cifratura, di essere uguale a 1 è pari a 1/2 indipendentemente dagli altri segnali, il one-time pad è perfettamente sicuro rispetto ad un *ciphertext-only attack* nel quale il crittanalista non ha alcuna conoscenza circa il messaggio originale.

L'inconveniente sta, però, nel dover operare con una chiave di dimensioni troppo grandi; in molti casi, infatti, avere una chiave segreta di lunghezza pari a quella del messaggio risulta troppo oneroso.

Si è ripiegato, così, su versioni approssimate del one-time pad, quali appunto i binary stream ciphers, che utilizzano chiave segrete corte per generare lunghe sequenze di bits (keystreams) da sommare al testo in chiaro; sequenze di questo tipo risultano, però, non essere più random, ma pseudorandom, vale a dire che le loro caratteristiche si avvicinano a quelle delle sequenze random senza mai raggiungerle.

La generazione di sequenze pseudorandom di bits il più possibile simili a sequenze random, che costituisce il problema centrale della crittografia, in generale, rappresenta il campo su cui gli stream ciphers concentrano tutti i loro sforzi.

E'difficoltoso, infatti, generare lunghe imprevedibili sequenze di segnali binari da una chiave casuale e corta e, tuttavia, sequenze di questo tipo sono desiderabili in crittografia, poichè, dato un ragionevole segmento dei loro segnali, è impossibile ricavare da esso altre informazioni sulla sequenza stessa.
Dal grado di imprevedibilità della sequenza pseudorandom dipende il grado di sicurezza del sistema: tanto più la sequenza pseudorandom si avvicina ad una sequenza random, tanto più alta è la probabilità del sistema di non essere forzato.

5.2 Sequenze pseudorandom e generatori pseudorandom di bits

L'obiettivo principale nella progettazione di stream cipher è produrre sequenze pseudorandom che siano il più vicino possibile a sequenze random. In generale una sequenza è considerata random se :

- non si riesce a riconoscere in essa alcun *modello (pattern)* rappresentato, per esempio, da gruppi di bits che si ripetono in un dato ordine;
- non si può fare alcuna previsione su di essa;
- non si può dare alcuna descrizione di essa.

In termini pratici ciò significa che il periodo della sequenza deve essere grande e vari patterns di una data lunghezza devono essere uniformemente distribuiti sulla sequenza. Questo non accade, in generale, per una sequenza pseudorandom: infatti, i bits appaiono essere *random in senso locale*, ma sono in qualche modo riproducibili e, quindi, sono soltanto *pseudorandom in senso largo*.

Il fatto stesso che la *keystream* (pseudorandom) possa essere generata efficientemente, dice che esiste certamente una descrizione di essa (cfr. punto 3)) che si può dare. Tuttavia, il generatore può produrre

sequenze cosidette indistinguibili se nessun calcolo fatto sulla keystream può rivelare questa semplice descrizione.

La chiave originale deve essere trasformata in un modo così complicato che risulta computazionalmente irrealizzabile ricavare la chiave. Da un punto di vista del sistema, è possibile formulare tre requisiti per avere dei generatori di keystream (pseudorandom bit generators) crittograficamente sicuri:

(1) Il periodo della keystream deve essere largo abbastanza per assecondare la lunghezza del messaggio trasmesso.

(2) I bits di output devono essere facili da generare.

(3) I bits di output devono essere difficili da predire. Dato il generatore e i primi *n* bits di output, *a(0), a(1), ... , a(n-1),* dovrebbe essere computazionalmente impossibile predire l' *(n+1)-simo* bit *a(n)* nella sequenza con una probabilità superiore al 50%. Questo significa che, data una porzione della sequenza di output, il crittanalista non dovrebbe poter generare altri bits nè in avanti (forward) nè indietro (backward).

Non esistono criteri universalmente applicabili e praticamente testabili per certificare la sicurezza dei generatori di bits.Tuttavia, è sempre valido il principio, del tutto empirico, che un dato metodo è considerato sicuro finchè non viene forzato. Così facendo, diversi sistemi sono stati proposti, e poi violati, nella storia della crittologia.

5.3 Tecniche per la generazione di sequenze pseudorandom

In questo paragrafo descriviamo brevemente alcuni dei metodi più semplici per generare sequenze pseudorandom.

5.3.1 Linear congruence generators (LCGs)

Questo metodo per generare numeri pseudorandom è basato su ricorrenze della forma:

$$X_{i+1}=aX_i+b \mod m.$$

Qui, (*a,b,m*) sono i parametri che descrivono il generatore e possono essere utilizzati come chiavi segrete. Se essi sono scelti con giudizio, i numeri X_i possono non ripetersi finchè tutti gli interi dell'intervallo [0, *m*-1] non sono occorsi. Per esempio, la sequenza generata da $X_i = 5X_{i-1}+3 \mod 16$ con $X_0 = 1$ è

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

Le sequenze generate da LCGs non sono, però, crittograficamente sicure. Data una sufficientemente lunga parte della sequenza, è possibile ricostruire i parametri *m, a, b.*

5.3.2 Feedback shift registers (FSRs)

Sequenze crittografiche "robuste" (cioè difficili da identificare) possono essere progettate sulla base di shift registers anche quando il feedback è lineare.

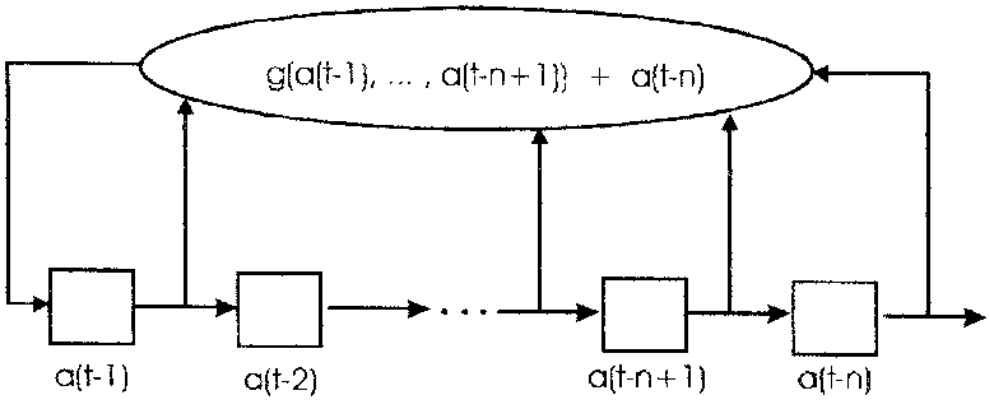


Fig. 5.2 : Shift register a n-stages

Un feedback shift register consiste di *n* flip-flops e una funzione di feedback (*feedback function*) che esprime ogni nuovo elemento *a(t)*, quando $t \geq n$, della sequenza in termini degli elementi *a(t-n), a(t-n+1), ... ,a(t-1),* precedentemente generati, come mostrato in Fig. 5.2. Affinchè il diagramma degli stati dello shift register sia costituito soltanto da cicli senza diramazioni, occorre che la feedback function sia non singolare, cioè sia della forma

$$a(t)=g(a(t-1), a(t-2),...,a(t-n+1)) \oplus a(t-n)$$

dove $\oplus$ denota l'operazione di XOR e l'elemento *a(t-n)* figura esplicitamente sulla destra.
A seconda che la funzione *g* sia lineare (e quindi implementabile con XORs) o no, il generatore è chiamato linear feedback shift register (LFSR) oppure non linear feedback shift register (NLFSR). Ogni elemento del FSR è chiamato stage e i segnali binari *a(0), a(1), a(2), ... ,a(n-1)* sono caricati al loro interno come dati iniziali per generare la sequenza.

Il periodo della sequenza prodotto da un FSR dipende sia dal numero di stages che dalle caratteristiche della connessione di feedback. Chiaramente il massimo periodo di una sequenza che può essere generata da un *n*-stage FSR con feedback non singolare è 2^n, il numero di possibili stati che un shift register a *n*-stages può avere.

5.3.2.1 Non linear feedback shift registers (NLFSRs)

Il diagramma degli stati di un FSR non singolare può avere diversi piccoli cicli, e la sequenza di output diventa insicura quando il generatore finisce in uno di essi. Una contromisura che si può adottare è progettare *n*-stage shift registers che generino sequenze con il periodo più grande possibile 2^n, le cosidette sequenze di De Bruijn di grado n. Un esempio di sequenza di De Bruijn di periodo 2^4 generata da un 4-stage NLFSR è la seguente:

1 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 ...

Il numero di possibili sequenze di De Bruijn a *n-stages* è pari a $2^{2^{n-1}-n}$ e queste sequenze hanno una ideale distribuzione di patterns. Per esempio, ogni pattern di lunghezza 4 è prodotto esattamente una sola volta dal generatore in un periodo di lavoro. Le sequenze di De Bruijn realizzabili da algoritmi conosciuti sono, però, tecnicamente difficoltose da implementare per la generazione veloce (*fast generation*) oppure soffrono di una grave debolezza legata alle loro caratteristiche di autocorrelazione.

Per esempio, per un'estesa classe di funzioni trattabili con la *fast generation*, la probabilità di coincidenza tra *a(t)* e *a(t-n)* è molto superiore ad 1/2. Il crittanalista nemico può sfruttare questa caratteristica.

5.3.2.2 Linear feedback shift registers (LFSRs)

I circuiti LFSR rappresentano i più importanti componenti nella costruzione di generatori pseudorandom di bit. Essi hanno una funzione di feedback della forma

$$a(t)=c_1a(t-1) \oplus c_2a(t-2) \oplus ... \oplus c_{n-1}a(t-n+1) \oplus a(t-n)$$

con $c_i \in \{0, 1\}$. La connessione di feedback di un LFSR può essere rappresentata formalmente dal cosidetto polinomiale di feedback (feedback polinomial)

$$f(x)=1+c_1x+c_2x^{n-2} + ... +c_{n-1}x^{n-1}+x^n$$

nell'incognita *x*, che non ha nessun altro significato che quello di simbolo matematico.

Il *feedback polinomial* decide il periodo e il comportamento statistico della sequenza di output. Per evitare output non significativi, lo stato zero dovrebbe essere escluso dal settaggio iniziale. Per esempio, se un LFSR a 4-stages ha un polinomiale di feedback

$$f(x)=1+x+x^2 +x^3+x^4,$$

allora, a seconda dei dati iniziali caricati dentro ai suoi stages, esso genera una delle tre sequenze significative di periodo 5 qui di seguito riportate:
a) 1 1 1 1 0 1 1 1 1 0 ...
b) 1 0 0 0 1 1 0 0 0 1 ...
c) 0 1 0 0 1 0 1 0 0 1 ...
Al contrario, se il LFSR ha il polinomiale di feedback

$$f(x)=1+x+x^4,$$

esso genera una singola sequenza significativa di periodo 15 con la migliore statistica che una sequenza di questa lunghezza può possedere:

1 0 1 1 0 0 1 0 0 0 1 1 1 1 0...

In generale, per garantire il periodo più largo possibile 2^n-1 per la sequenza di output, il polinomiale di feedback *f(x)* del LFSR deve essere *primitivo.* Questo significa che *f(x)* deve essere scelto in modo tale che il minimo intero positivo *T* che rende X^T-1 divisibile per *f(x)* possa essere $T=2^n$-1.

Esistono algoritmi pronti per verificare se un polinomiale è primitivo o no; in generale, si può comunque dire che il numero di polinomiali primitivi di grado *n* è

$$\square(2^n-1)/n$$

dove □(*x*), nota come funzione di Eulero (par. 2.2), indica il numero di interi positivi minori dell'intero *x* e primi rispetto ad esso.

Una sequenza generata da un LFSR con un polinomiale primitivo, è chiamata una sequenza LFSR a lunghezza massimale (maximal-length LFSR sequence) o semplicemente una *m*-sequenza. La massimalità del periodo garantisce nello stesso tempo buone statistiche. E' facile mostrare che per un qualunque intero $0 \leq k \leq n$ e per un qualunque *k*-pattern a, la probabilità di un segmento x di lunghezza *k*, scelto a caso dalla *m*-sequenza, di essere uguale al pattern a è

$$prob(\mathbf{x}=\mathbf{a}) = \begin{cases} 1/2^k + 1/2^{n+k}, & if \quad \mathbf{a} \neq 0 \\ 1/2^k - 1/2^n, & if \quad \mathbf{a} = 0 \end{cases}$$

Si può anche dedurre che per ogni intero $0 < q < 2^n$-1, la probabilità di coincidenza tra il segnale *a(t)*, *a(t+q)* è

$$prob(a(t)=a(t+q))=1/2-1/2^n,$$

che significa che la *m*-sequenza ha anche caratteristiche ideali di autocorrelazione.

A dispetto della larga scelta di polinomiali di feedback primitivi, in aggiunta ai periodi grandi e alle statistiche ideali, le *m*-sequenze non possono essere usate come *keystreams* senza sottostare ad un ulteriore

trasformazione crittografica. Infatti, la chiave della segretezza (stato iniziale e connessione di feedback) di un *n*-stage LFSR può essere determinata da appena *2n* bits successivi della sequenza di output (per la dimostrazione di questo si veda [ref.15]).

In generale, ogni sequenza periodica può essere prodotta da un LFSR non singolare. Esiste un'efficiente procedura di sintesi per trovare il polinomiale di feedback del più corto LFSR che può generare la sequenza; la lunghezza di un tale LFSR è chiamata la complessità lineare della sequenza a, ed è denotata con *LC(a)*. Di conseguenza, per progettare un generatore adatto all'uso crittografico, si deve garantire un abbastanza grande limite inferiore, indipendente dalla chiave, per la complessità lineare delle sequenze che esso genera.

BIBLIOGRAFIA

[1] W. Diffie and M.E. Hellman, "Privacy and authentication: An introduction to cryptography ," Proc.IEEE, vol. 67, pp.397-427, Mar.1979.

[2] G.J.Simmons, "Authentication theory/coding theory,"in *Advances in CriptologyProceedings of CRYPTO 84,* G.R. Blakley and D. Chaum, Eds.Lecture Notes in Computer Science, No. 196. New York, NY: Springer, 1985, pp.411-431.

[3] A. Shamir. "A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem,"*IEEE Trans. Informat.Theory*, vol.IT-30, pp.699-704,Sept.1984.

[4] D.B.Newman,Jr., and R. L. Pickholtz,"Cryptography in the private sector,"*IEEE Commun.Mag.*,vol.24, pp.7-10,Aug.1986.

[5] C.E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech.J*vol.28, pp.656-715, Oct.1949.

[6] W. Diffie and M.E. Hellman, "New directions in cryptography,"*IEEE Trans.Informat. Theory,* vol IT-22, pp.644-654, Nov.1976.

[7] R.C.Merkle, "Secure communication over insecure channels,"*Comm. ACM,* pp.294-299, Apr. 1978.

[8] W. Diffie and M.E. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard,"*Computer*, vol. 10, pp.74-84, June 1977.

[9] R.C. Merkle and M.E. Hellman, "On the security of multiple encryption,"*Comm. ACM,* vol.24, pp. 465-467, July 1981.

[10] J. L. Massey,"Shift-register synthesis and BCH decoding,"*IEEE Trans. INformat. Theory*, vol. IT-15, pp.122-127, Jan.1969.

[11] S. C. Pohlig and M. E. Hellman,"An Improved algorithm for computing logaritms in GF(p) and its cryptographic significance,"*IEEE Trans. Informat Theory*, vol. IT-24, pp. 106-110, Jan.1978.

[12] R. L. Rivest, A. Shamir, and L. Adleman, Ä method for obtaining digital signatures and public-key cryptosystems," *Comm. ACM*, vol.21, pp.120-126, Feb.1978.

[13] A. M. Odlyzko, "on the complexity of computing discrete logarithms and factoring integers,"to appear in *Fundamental Problems in Communication and Computation*, B. Gopinath and T. Loven, Eds. New York, NY: Springer.

[14] C. Pomerance, "Analysis and comparison of some integer factoring algorithms,"in *Computational Number Theory*,H. W. Lenstra, Jr., and R. Tijdeman, Eds. Amsterdam, The Netherlands: Math, Centre Tract, 1982.

[15] K. Zeng, C.H. Yang, D.Y. Wei and T.R.N. Rao, "Pseudorandom Bit Generators in Stream-cipher Cryptograph", *Computer* pp. 8-16, Febr. 1991.

[16] C. H. Bennett, G. Brassard and A.K. Ekert, "Crittografia quantistica", *Le Scienze*, numero 292, pp. 84-93, Dic.1992.

[17] "Contemporary Cryptology : The Science of Information Integrity", edited by G.J. Simmons, *IEEE Press* 1991.

[18] J. L. Massey,"An Introduction to contemporary Cryptology", *Proceeding of the IEEE,* vol.76, NO. 5, May 1988.

[19] M.O. Rabin,"Probabilistic algorithm for primality testing", *J. Number Theory,* vol. 12, pp.128-138, 1980.