Cos'è la Crittografia?

Paolo Caressa

Primavera 2005

Capitolo 1

Sistemi crittografici

Col termine *crittografia* o *crittologia* si intende una collezione di metodi, tecniche e algoritmi che consentono di trasformare stringhe di informazioni in altre stringhe di informazioni, al fine di renderle inintellegibili ad estranei ma decifrabili da soggetti che condividano delle informazioni aggiuntive sul modo col quale le stringhe sono state codificate.

La crittografia ha una storia ben più antica di quella della *computer science*[1], ma il carattere deterministico degli algoritmi usati li rendono adatti alle implementazioni informatiche: inoltre, lo sviluppo dell'informatica ha contribuito notevolmente ad incrementare gli studi in materia. Poiché la ricerca ed il conseguente comparire di tecnologie innovative in questo campo procedono a ritmi vertiginosi, l'aggiornamento, sia delle conoscenze teoriche di base che degli strumenti applicativi utilizzati, è fondamentale per contrastare i rapidi tempi di obsolescenza di queste tecnologie.

1.1 Breve descrizione di un sistema crittografico

Esistono varie tipologie di sistemi crittografici, che rientrano nel seguente schema: un *sistema crittografico* è determinato dai seguenti dati:

(1) Il *testo in chiaro*, cioè le informazioni da criptare (testuali, numeriche, etc.);

(2) Il *testo cifrato*, cioè le informazioni criptate da trasmettere (in genere numeriche);

[1] Per una discussione accessibile della storia della crittografia fino ai giorni nostri si può consultare [5].

A	P
B	O
C	I
D	S
E	L
F	T
G	H

H	Y
I	W
J	B
K	U
L	X
M	E
N	V

O	G
P	Z
Q	H
R	K
S	C
T	M
U	N

V	A
W	D
X	Q
Y	F
Z	J

Figura 1.1: Esempio di cifratura basata sulle permutazioni delle lettere

(3) Le *chiavi di codifica* (in genere dati numerici);

(4) Gli *algoritmi di cifratura*, cioè delle procedure che trasformino, utilizzando le chiavi, il testo in chiaro in un testo cifrato;

(5) Gli *algoritmi di decrittazione*, cioè delle procedure che trasformino, utilizzando le chiavi, il testo cifrato in un testo in chiaro;

con la condizione che per ogni algoritmo di cifratura esista un algoritmo di decrittazione che renda possibile in modo completo ricostruire il testo in chiaro a partire dal testo cifrato.

Un esempio molto semplice (ed antico) è la sostituzione di lettere: in questo caso il testo in chiaro è il testo da trasmettere, le chiavi sono date da una permutazione delle lettere dell'alfabeto, l'algoritmo di cifratura consiste nel sostituire ad una lettera la sua corrispondente permutazione, e l'algoritmo di decrittazione consiste nell'effettuare la sostituzione inversa. Ad esempio, usando la permutazione fornita dalla seguente tabella (le cui colonne sono le chiavi nel nostro caso)
possiamo codificare il messaggio `CRITTOGRAFIA` come `IKWMMGRKPTWP`. Questo algoritmo è così semplice che è facile, specie avvalendosi della velocità dei moderni calcolatori, decrittarlo senza conoscerne le chiavi[2].

In effetti il motivo principale per cui si applicano le tecniche crittografiche è che si desidera trasmettere informazioni riservate che sono soggette a "spionaggio": questo vuol dire che ogni sistema crittologico va progettato tenendo in mente che c'è qualcuno desideroso di "spezzarlo", cioè di decrittare le informazioni riservate ad insaputa di chi le trasmette e di chi le riceve.

[2]Un esempio è dato nel racconto *Lo scarabeo d'oro*, (1843) di Edgar Allan Poe, che descrive una completa crittoanalisi basata sull'analisi delle frequenze di un testo cifrato usando un algoritmo di permutazioni di lettere.

1.2 Rischi di un sistema crittografico

Col termine *crittoanalisi* si denota in genere il tentativo di decrittare un messaggio cifrato senza conoscerne né le chiavi né gli algoritmi di cifratura e decrittazione. Apparentemente sembra difficile decrittare un messaggio senza conoscere il metodo utilizzato per cifrarlo, ma basterà pensare che l'informazione del messaggio cifrato, per quanto nascosta, non si è persa, per capire che tecniche statistiche applicate al messaggio cifrato possono fornire molte informazioni sulla sua struttura ed anche sul suo contenuto. In particolare chi trasmette i messaggi deve presumere che si possa riconoscere il sistema crittografico in uso.

Distinguiamo i seguenti tipi di attacco crittoanalitico:

- *Ciphertext-only attack*: in questo caso si conosce solo il testo cifrato del messaggio che si vuole decrittare.
- *Known-plaintext attack*: in questo caso si conoscono esempi di testi cifrati e dei corrispondenti testi in chiaro.
- *Chosen-plaintext attack*: in questo caso si può scegliere fra vari testi in chiaro ed ottenerne i corrispondenti testi cifrati.
- *Chosen-ciphertext attack*: in questo caso si può scegliere fra testi cifrati ed ottenerne i corrispondenti testi in chiaro.

Le strategie per porre in essere questi attacchi sono molteplici: la più ovvia, ad esempio per un *ciphertext-only attack*, è provare con tutte le chiavi possibili (operazione che una rete di calcolatori attuali può tentare di compiere in tempi ragionevoli) e che consente di decrittare ad esempio i messaggi che usano il sistema DES (*Data Encryption Standard*), fino a qualche anno fa lo standard, ora considerato insicuro (sebbene, con alcune varianti, garantisca ancora sicurezza). Altrimenti, per un *known-plaintext attack*, si può utilizzare un'analisi delle frequenze dei simboli nei testi in chiaro e cifrati, e così via.

1.3 Classificazione dei sistemi crittografici

I sistemi crittografici si possono grosso modo distinguere in due classi: *sistemi simmetrici* e *sistemi asimmetrici*.

Chiariamo la differenza con un esempio: supponiamo che Alice[3] voglia inviare un messaggio criptato a Bob, e che usi una chiave k per la cifratura, che necessita della chiave h per la decifratura, usata da Bob.

Se nel sistema crittografico la chiave k è identica alla chiave h, o comunque l'una determina in modo ovvio l'altra, il sistema si dice *simmetrico* o *a chiave segreta* (*secret-key*). Infatti, la sicurezza in un sistema simmetrico è garantita, non solo dalla complessità degli algoritmi di cifratura e decrittazione, ma *soprattutto dal fatto che le chiavi siano note soltanto ad Alice e Bob*. Infatti, se il malintenzionato Oscar conoscesse k o h, potrebbe determinare l'altra chiave e quindi decrittare (o anche cifrare) un messaggio fra Alice e Bob. Pertanto la segretezza della chiave è cruciale in questi sistemi: un esempio di sistema simmetrico è il semplice sistema di permutazioni di lettere descritto in precedenza nella tabella 1.1.

In altri termini, un limite notevole dei sistemi simmetrici è che Alice e Bob debbono *condividere un segreto* per potersi scambiare le informazioni: si pensi alla difficoltà di gestire questa situazione ad esempio per una rete di comunicazioni. Se ci sono n utenti in questa rete, e se si suppone che ciascuno di essi possa scambiare messaggi con un qualsiasi altro, ci sono $n(n-1)/2$ chiavi segrete che debbono essere mantenute (una per ciascuna coppia): ad esempio per $n = 100$, servono 4950 chiavi, tutte da mantenere segrete.

Se invece nel sistema crittografico la chiave k e la chiave h sono distinte ed è *praticamente* impossibile ricostruire l'una dall'altra e viceversa, il sistema si dice *asimmetrico* o *a chiave pubblica* (*public-key*). In un sistema asimmetrico, se Alice deve inviare un messaggio a Bob, lo cifra utilizzando una chiave pubblica k che Bob avrà divulgato a chiunque debba comunicare con lui: infatti, in questi sistemi, Bob possiede anche una *chiave privata* h, che gli consente di decrittare il messaggio cifrato con la chiave pubblica.

Dato che da k non si può risalire a h né viceversa, Bob può tranquillamente diffondere la sua chiave pubblica, il che mette in condizione chiunque di cifrare un messaggio rivolto a lui: ma se un tale messaggio viene intercettato, quando è già cifrato, non si è in grado di decrittarlo, perché per questo è necessaria la chiave privata k che Bob custodisce gelosamente e che non ha il bisogno di rivelare a nessuno.

Se, ad esempio n utenti debbono scambiarsi messaggi fra loro, ciascuno custodirà la propria chiave privata, mentre le chiavi pubbliche saranno elencate in una tabella accanto ai nomi dei loro "proprietari". Ovviamente questa tabella dovrà essere *protetta da aggiornamenti indebiti*: se Oscar vuole inter-

[3]Tradizionalmente in questi esempi i due soggetti che tentano di comunicare in modo sicuro sono Alice e Bob, mentre il potenziale intruso nella loro comunicazione criptata è Oscar.

cettare i messaggi di Bob, proverà a sostituire nella tabella la chiave pubblica di Bob con la propria. Discuteremo questo problema nella sezione 1.5.4.

A prima vista, i sistemi asimmetrici presentano dei notevoli vantaggi su quelli simmetrici, ma, ovviamente, ci sono dei *contro* a questi *pro*: in particolare, mentre gli algoritmi di cifratura e decrittazione per i sistemi simmetrici sono estremamente efficienti, lo stesso non si può dire per quelli che implementano sistemi asimmetrici.

Pertanto, al fine di temperare sicurezza ed efficienza, spesso si ricorre a soluzioni "ibride", che cioè utilizzano la maggior sicurezza dei sistemi asimmetrici quando si tratta di codificare informazioni essenziali e la maggior efficienza dei sistemi simmetrici quando si tratta di codificare informazioni ingenti.

Facciamo un esempio: Alice, al solito, vuole mandare un messaggio a Bob; per farlo utilizza un sistema simmetrico con chiave k per codificare il suo messaggio, e produce il testo cifrato, in modo efficiente (dato che il sistema è simmetrico). Poi trasmette il testo cifrato col sistema simmetrico a Bob, e gli trasmette anche la chiave k usata per la cifratura, solo che non la trasmette in chiaro, ma dopo averla cifrata con un sistema asimmetrico, la cui chiave pubblica è, diciamo, k' (questa cifratura è efficiente anche se usa un sistema asimmetrico, perché la chiave è comunque piccola). Quindi: Bob riceve un messaggio cifrato lungo ed un messaggio cifrato corto: il messaggio corto viene decrittato usando la sua chiave privata, ed il risultato è una chiave da utilizzare per decrittare il messaggio lungo. In questo modo un testo è trasmesso assieme alla sua chiave usando gli standard di sicurezza di un sistema asimmetrico e quelli di efficienza di un sistema simmetrico.

1.4 Sistemi simmetrici

Il semplice esempio di sistema simmetrico che abbiamo illustrato in precedenza è ovviamente del tutto insicuro, perché conserva le informazioni sulla struttura del testo, in particolare la frequenza delle lettere: poiché è facile stabilire quali sono le lettere più frequenti in una lingua, sarà facile farle corrispondere ai simboli più frequenti del testo cifrato e quindi ricostruirne via via le chiavi[4].

Questo problema affligge tutti i sistemi crittografici in cui le chiavi siano funzioni lineari dei dati che vogliono mettere in corrispondenza (ad esempio

[4]Metodo scoperto dagli scienziati islamici medievali: ad esempio era già noto ad Abu Yusuf ibn Ishaq al-Kindi che scrisse un trattato *Sulla decifrazione dei messaggi cifrati*, nel IX secolo.

i classici sistemi ECB, CBC, CFB, OFB). Un sistema simmetrico più sicuro, perché non lineare, è il *Data Encryption Standard* (DES), che tuttavia nel 2000 si è dimostrato non più tanto sicuro[5] ed è stato sostituito con l'*Advanced Encryption Standard* (AES) che utilizza l'algoritmo Rijndael, risultato vincente dopo una competizione durata tre anni. Poiché quest'ultimo utilizza le stesse tecniche del DES, e dato che del DES stesso esistono varianti sicure, ci sembra opportuno descrivere quest'ultimo come modello standard di sistema crittografico simmetrico.

1.4.1 Il sistema crittografico DES

Nel sistema DES sia il testo in chiaro che il testo cifrato che le chiavi sono stringhe di 64 bit[6]: le chiavi hanno la particolarità che, se spezzate in otto byte, la somma dei bit di ciascun byte è un numero dispari[7]: in questo modo il numero delle chiavi è 2^{56}, (circa settandadue milioni di miliardi).

L'algoritmo di cifratura del DES consta di tre passi:

(1) Permutazione iniziale.

(2) Generazione delle *round keys*.

(3) Cifratura in blocchi.

Nel passo (1) viene semplicemente applicata una permutazione ai gruppi di 64 bit del testo in chiaro.

Nel passo (2) si producono, a partire da una singola chiave k, 16 chiavi k_1, ..., k_16, le *round keys*, che sono desunte dalla chiave k usando due algoritmi di manipolazione di bit, il PC1 e il PC2, nel cui dettaglio tecnico non vogliamo entrare[8].

Nel passo (3), il testo viene sottoposto alla *cifratura in blocchi di Feistel*; per farlo vengono adoperate le *round keys* in questo modo: ciascun blocco di 64 bit è per prima cosa spezzato in due blocchi sinistro e destro da 32 bit $(S_0|D_0)$, dopo di che si costruiscono ricorsivamente le sequenze

$$(S_i|D_i) = (D_i - 1|S_i - 1\hat{} \ C_{k_i}(D_i - 1))i = 1, 2, ..., k$$

[5]Per informazioni storiche si vedano gli archivi del *National Institute of Standards and Technology*, in particolare la pagina http://csrc.nist.gov/CryptoToolkit/aes/

[6]Cioè il testo verrà codificato in binario, ad esempio usando i codici ASCII, e spezzato in gruppi di 64 bit, eventualmente aggiungendo dei bit finali se il numero di bit totali non è divisibile per 64.

[7]Pertanto, sette bit di ciascun byte determinano il bit rimanente e, tanto per dirne una, errori di trasmissione di un bit possono essere corretti.

[8]Il lettore interessato può consultare utilmente il testo di Bellare e Goldwasser [1], come pure quello di Buchmann [2]

dove $C_{k_i}(D_i - 1)$ è il testo cifrato prodotto con la chiave k_i a partire dal testo in chiaro $D_i - 1$, e dove con ^ denotiamo l'operazione xor[9] fra i bit del blocco di testo in questione.

Inoltre, e questa è l'operazione che implementa la sicurezza dell'algoritmo, ad ogni applicazione di una chiave, il blocco D_i viene cifrato applicandovi l'operazione xor applicata alla chiave (dopo aver completato il blocco) ed applicando a ciascuno dei sottoblocchi di 6 bit del blocco così ottenuto otto funzioni particolari $S_1, ..., S_8$, altamente non lineari, il cui risultato è il blocco cifrato.

La sequenza finale $(S_{16}|D_{16})$ è il blocco cifrato.

La sicurezza di questo algoritmo risiede essenzialmente nella sequenza intricata di suddivisioni ed operazioni di xor fra i bit dei blocchi coinvolti e nella non linearità delle funzioni applicate: ciascuno dei passi è tuttavia perfettamente invertibile, per questo il sistema DES è simmetrico.

Il suo pregio è l'efficienza col quale può essere implementato, il suo difetto è il fatto che una ricerca esaustiva nello spazio delle chiavi può, con le odierne reti di calcolatori, decrittarne i messaggi (in meno di una giornata[10]), almeno nella forma in cui lo abbiamo presentato: esistono varianti ancora sicure ma che perdono in efficienza.

1.5 Sistemi asimmetrici

Il sistema asimmetrico più vecchio e meglio conosciuto è l'RSA, che prende il nome dai suoi inventori: Ron Rivest, Adi Shamir e Len Adleman[11]. Un altro sistema che ha una sicurezza equivalente (in qualche senso non minore, quindi potenzialmente maggiore) dell'RSA è basato sull'*algoritmo di Rabin*[12]. Infine una terza classe di sistemi a chiave pubblica che offrono una elevata sicurezza si basa sull'algoritmo *ElGamal*.

Curiosamente, il motivo della maggior sicurezza di questi sistemi crittografici sta nel fatto che i loro ideatori hanno fatto tesoro non tanto delle conoscenze matematiche fin qui accumulate nella teoria dei numeri (che costituisce il fondamento teorico della crittografia) ma piuttosto di *quelle ancora da scoprire*.

[9] Si tratta dell'operazione di "or esclusivo" fra bit, definita come 0^0 = 1^1 = 0 e 0^1 = 1^0 = 1.

[10] cfr. e.g. http://www.eff.org/descracker.html

[11] R. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, 21(2), 120-126, 1978.

[12] M.O. Rabin, *Digital Signature and Public Key Functions as Intractable as Factoring*, Technical Memo TM-212, MIT, 1979.

In effetti esistono, in matematica come in altre discipline, i cosiddetti *problemi aperti*, cioè dei problemi universalmente noti e sui quali i più valenti scienziati hanno profuso il loro impegno ma che, malgrado tutto, resistono ancora a tentativi di soluzione. In teoria dei numeri due tali problemi, fra gli altri, sono quelli della *fattorizzazione efficiente dei numeri interi* ed il problema del *calcolo del logaritmo discreto*. Entrambi sono problemi che concernono la determinazione di algoritmi efficienti per svolgere un certo calcolo, ed entrambi, a tutt'oggi, sono problemi insoluti.

Ora, gli algoritmi crittografici asimmetrici sono concepiti in modo tale che la loro decifrazione implicherebbe la soluzione di uno di questi problemi. Per essere precisi, si può dimostrare che *decrittare l'RSA implica il saper fattorizzare un numero intero molto grande*: quindi se esiste un modo, per farlo il problema della fattorizzazione sarà risolto. L'algoritmo di Rabin è ancor meglio legato al problema della fattorizzazione, dato che si può dimostrare che gli è equivalente. Infine, per decrittare un messaggio cifrato con l'algoritmo ElGamal bisogna saper calcolare il logaritmo discreto, il che implica la soluzione di un altro problema aperto.

Dunque la sicurezza di questi algoritmi è garantita dal fatto che nessuno, per quel che se ne sa, ha ancora risolto dei difficili problemi matematici: questo vuole anche dire che, se qualcuno dovesse risolverli, questi algoritmi diverrebbero improvvisamente insicuri. Poiché la comunità scientifica diffonde e fa circolare le informazioni relative alle proprie scoperte, questa è vista come una garanzia che se un giorno questi problemi saranno risolti il fatto sarà subito di pubblico dominio[13].

Un'ultima precisazione: quando asseriamo che non esiste un algoritmo per fattorizzare un numero intero o per calcolare il logaritmo discreto intendiamo che non ne esiste uno il cui tempo di esecuzione sia polinomiale: infatti *teoricamente* si tratta di problemi molto semplici, è la loro *implementazione pratica* che non si può ottenere in modo efficiente.

1.5.1 Il sistema crittografico RSA

Il sistema crittografico RSA si basa, come tutti i sistemi asimmetrici, su tecniche di teoria dei numeri: abbiamo specificato come la sicurezza dell'RSA risieda nel fatto che a quanto pare nessuno è in grado di fattorizzare numeri interi

[13]Naturalmente questa non è una garanzia in assoluto: enti di ricerca segreti, come quelli militari ad esempio, potrebbero custodire scoperte scientifiche senza diffonderle, ed utilizzarle ad insaputa di tutti: ad esempio solo nel 1997 il governo britannico ha rivelato di possedere fin dal 1969 un sistema a chiave pubblica e già nel 1975 una funzione di cifratura sicura, del tipo di quella utilizzata nel metodo di Diffie-Hellman (cfr.2.5.4).

molto grandi (ovviamente esistono numerosi algoritmi di fattorizzazione[14] che tuttavia sono troppo lenti per fattorizzare numeri molto grandi).

Il funzionamento dell'RSA è il seguente[15]: Bob, che vuole ricevere dei messaggi cifrati da Alice, sceglie due numeri primi[16] molto grandi p e q, più o meno dello stesso ordine di grandezza, e ne fa il prodotto $n = pq$. Inoltre sceglie un intero e tale che[17]

$$(*) \qquad \begin{array}{ll} 1 < e < (p-1)(q-1) & \text{in modo che} \\ \mathrm{MCD}(e, (p-1)(q-1)) = 1 & \end{array}$$

La chiave pubblica di Bob è data dalla coppia (n, e); la chiave privata è invece un numero h maggiore di 1 e minore di $(p-1)(q-1)$ e tale che $eh - 1$ sia divisibile per $(p-1)(q-1)$. La condizione $(*)$ garantisce che un tale numero esiste sempre, e (una semplice estensione de) l'algoritmo della divisione di Euclide consente di mostrare che è unico e di calcolarlo effettivamente.

Facciamo l'esempio più semplice possibile: supponiamo che $p = 3$ e $q = 5$; allora $n = 3 \times 5 = 15$, $(p-1)(q-1) = 2 \times 4 = 8$ ed una scelta per e è 3 (infatti nessun numero, eccettuato 1, divide sia 3 che 8). Con queste scelte la chiave privata h è in questo caso 3 (infatti $3 \times 3 = 9$ che diviso per 8 fa 1 col resto di 1, cioè eh diviso $(p-1)(q-1)$ ha resto di 1).

Vediamo ora qual'è l'algoritmo di cifratura: il testo che vogliamo cifrare è un numero t compreso fra 0 e $n-1$ (estremi inclusi). Il testo cifrato è semplicemente il numero che si ottiene come resto della divisione di t^e per n (esiste un algoritmo veloce per effettuare questo calcolo).

Per decifrare un messaggio di Alice (o di chiunque altro abbia usato la sua chiave pubblica $k = (n, e)$), Bob non deve fare altro che prendere il numero c che contiene il testo cifrato e calcolare il resto della divisione di c^h per n: si può infatti dimostrare che questo resto è esattamente il testo in chiaro

[14] Fin dall'antichità: ad esempio il *crivello di Eratostene*, III secolo a.C.; un metodo moderno molto efficiente è il *crivello quadratico*.

[15] Nel descrivere gli algoritmi *non dimostreremo perché funzionano* ma *illustreremo solo i calcoli* che vanno eseguiti per implementarli: il perché questi calcoli effettivamente consentano di cifrare e decrittare messaggi richiede argomentazioni e ragionamenti di teoria dei numeri, per i quali si rimanda ai testi citati in bibliografia.

[16] Rammentiamo che un numero intero (positivo) è primo se non è divisibile per nessun altro intero (a parte se stesso e 1): ad esempio 11 è primo perché una qualsiasi fattorizzazione $11 = n \times m$ implica necessariamente che $n = 1$ e $m = 11$ o viceversa; invece 12 non è primo, dato che ad esempio $12 = 3 \times 4$; pertanto un numero primo non possiede fattori che non siano se stesso e 1. Se un numero non è primo, fattorizzarlo vuol dire proprio esprimerlo come prodotto di due numeri che non siano né 1 né il numero stesso.

[17] Con la notazione $\mathrm{MCD}(a, b)$ denotiamo il massimo numero intero che sia un divisore sia di a che di b.

che era stato codificato (di nuovo si usa l'algoritmo veloce per calcolare un esponenziale).

Ad esempio codifichiamo il numero $t = 12$ nel nostro *toy example*: usando la chiave pubblica $(15, 3)$ dobbiamo calcolare $12^3 = 1728$, e prendere il resto della divisione di quest'ultimo per 15, che fa $c = 3$ ($1728 = 115 \times 15 + 3$). Per ritrovare il testo in chiaro dal testo criptato 3 calcoliamo il resto della divisione di $3^3 = 27$ per 15, ottenendo di nuovo 12, dato che $27 = 15 \times 1 + 12$.

In un esempio realistico si userebbero numeri primi nell'ordine dei 512 bit ciascuno. Esistono inoltre altre considerazioni, sia di efficienza che di sicurezza (spesso contrastanti fra loro): ad esempio la scelta del numero e, che è parte della chiave pubblica: per motivi di efficienza andrebbe scelto il più piccolo possibile, anche se questa scelta può esporre ai cosiddetti *low-exponent attacks*[18].

Dal punto di vista dell'efficienza, l'RSA può essere molto più lento del DES: ad esempio se i primi scelti sono sui 512 bit, l'operazione di decrittazione può richiedere qualcosa come il calcolo di 1024 quadrati e 512 moltiplicazioni (modulo n cioè prendendo i resti delle divisioni).

1.5.2 Il sistema crittografico basato sull'algoritmo di Rabin

Si è spiegato più sopra che decrittare (senza conoscere la chiave privata) l'RSA implica poter fattorizzare un numero molto alto: in realtà non è chiara se questa sia una equivalenza[19]. Un altro sistema crittografico asimmetrico, la cui sicurezza è stato invece dimostrato essere equivalente alla incapacità di fattorizzare gli interi in modo efficiente, è basato sull'*algoritmo di Rabin*, che oltre a questo è notevolmente più veloce dell'RSA: diamone una breve descrizione.

Alice insiste nel voler mandare messaggi sicuri a Bob, ma stavolta la chiave pubblica di Bob è un singolo numero k calcolato come segue: Bob sceglie due numeri primi molto grandi p e q tali che divisi per 4 diano come resto 3; la sua chiave pubblica sarà allora $k = p \times q$ e la sua chiave privata sarà la coppia (p, q).

[18] Dato che è il più importante sistema asimmetrico, l'RSA è stato sottoposto a notevoli studi per quel che riguarda le sue eventuali falle: cfr. e.g. D. Boneh, *Twenty years of attacks on the RSA cryptosystem*, Notices of the American Mathematical Society (AMS), Vol. 46, No. 2, pp. 203-213, 1999.

[19] Alcune evidenze suggeriscono il contrario: cfr. D. Boneh, R. Venkatesan, *Breaking RSA may not be equivalent to factoring*, Pro. Eurocrypt 1998, Lecture Notes in Computer Science, 1233, pp. 59-71, Springer, 1998.

Il messaggio che Alice può inviare è un numero t fra 0 e $n-1$ (estremi inclusi) che viene cifrato calcolando il resto c della divisione di m^2 per n. Per decifrare questo messaggio, Bob deve svolgere un po' di calcoli (apparentemente più complicati che nel caso dell'RSA ma più efficienti da svolgere con un calcolatore). Per prima cosa determina due numeri interi (anche negativi) a e b tali che

$$ap + bq = 1$$

(questo è sempre possibile). Poi calcola i due numeri

$$\begin{array}{rlll} x = & \text{resto della divisione di} & c^{(p+1)/4} & \text{per } p, \\ y = & \text{resto della divisione di} & c^{(q+1)/4} & \text{per } q, \end{array}$$

ed i seguenti coefficienti[20]

$$\begin{array}{rlll} r = & \text{resto della divisione di} & (apy + bqx) & \text{per } n, \\ s = & \text{resto della divisione di} & (apy - bqx) & \text{per } n, \end{array}$$

A questo punto il messaggio in chiaro t è uno fra i seguenti: r, $-r$, s, $-s$. Per capire quale di questi messaggi è quello originale Bob si può affidare alla verosimiglianza oppure può aver convenuto con Alice che un pezzo del messaggio contenga una sequenza identificativa (ad esempio certi bit sono sempre uno, o simili).

Come esempio riprendiamo quello che abbiamo mostrato nel caso dell'RSA: dunque $p = 3$, $q = 7$ (si noti che 5 non va bene perché 5 diviso 4 ha 1 come resto e non 3), quindi $n = 21$ è la chiave pubblica. Alice vuole cifrare il messaggio $t = 12$, e per farlo prende il resto della divisione del suo quadrato per 21:

$$c = \text{resto della divisione di } 144 \text{ per } 21 = 18$$

Vediamo come Bob riesce a decifrare questo messaggio: per prima cosa deve determinare a e b in modo che $3a + 7b = 1$, ad esempio $a = -2$ e $b = 1$. Poi calcola

$$\begin{array}{rlll} x = & \text{resto della divisione di} & 18^{(3+1)/4} & \text{per } 3 = 0, \\ y = & \text{resto della divisione di} & 18^{(7+1)/4} & \text{per } 7 = 2, \end{array}$$

ed infine determina

$$\begin{array}{rlll} r = & \text{resto della divisione di} & (-2 \times 3 \times 2 + 1 \times 7 \times 0) & \text{per } 21 = -12, \\ s = & \text{resto della divisione di} & (-2 \times 3 \times 2 - 1 \times 7 \times 0) & \text{per } 21 = -12, \end{array}$$

pertanto -12 oppure 12 è il messaggio originale.

[20] Questi conti non sono arbitrari: stiamo calcolando, senza saperlo, delle radici quadrate di numeri interi: affinché questo sia vero è necessaria la condizione che sia p che q divisi per 4 abbiano come resto 3.

1.5.3 Il sistema crittografico El Gamal

Il sistema crittografico ideato da Taher El Gamal[21] confida nella incapacità di calcolare efficientemente un certo numero intero, chiamato *logaritmo discreto*: senza entrare nei dettagli di cosa ciò significhi, mostriamo in pratica il funzionamento di questo importante sistema crittografico a chiave pubblica.

L'idea si basa sul seguente fatto: dato un qualsiasi numero primo p esiste sempre un numero intero g tale che le potenze $g^1, g^2, ..., g^{p-2}$, se divise per p, non diano come resto 1, mentre g^{p-1} lo dà[22]. Bob sceglie un numero primo p, un numero g con questa proprietà[23] ed un numero h compreso fra 0 e $p-2$ (inclusi), e calcola il numero B come resto della divisione di g^h per p. La sua chiave pubblica sarà allora la terna (p, g, B), mentre la sua chiave privata sarà h.

Alice può inviare a Bob un messaggio dato da un numero t compreso fra 0 e $p-1$, e per cifrarlo sceglie pure lei un esponente e compreso fra 0 e $p-2$ (inclusi), e calcola il numero A come resto della divisione di g^e per p. A questo punto il testo cifrato sarà dato dal numero c ottenuto come resto della divisione di $B^e \times t$ diviso p.

Per decrittare il messaggio, Bob deve non solo averne il testo cifrato c ma anche la "sotto-chiave" A, che Alice dovrà comunicargli (in chiaro: non importa se altri la conoscono). A partire da A e c, usando la sua chiave privata h, Bob ricostruisce il testo in chiaro t col calcolo seguente:

$$t = \text{ resto della divisione di } A^{p-1-h} \times c \text{ per } p$$

Ad esempio, prendiamo $p = 13$: una scelta per g è allora 2 (vanno bene anche 6, 7 e 11); scegliamo poi un esponente $h = 5$, ad esempio, e calcoliamo $B =$ resto della divisione di $25 = 32$ per 13, ottenendo $B = 6$. Allora la chiave pubblica è $(13, 2, 6)$, mentre la chiave privata è 5.

Ora proviamo a cifrare il messaggio $t = 12$; per farlo dobbiamo scegliere un altro esponente e, prendiamo ad esempio $e = 8$, e calcoliamo A come il resto della divisione di $2^8 = 256$ per 13, cioè $A = 9$. Per cifrare il testo $t = 12$ calcoliamo invece il resto c della divisione di $6^8 \times 12 = 20155392$ per 13, cioè $c = 10$.

Per decrittare questo messaggio $c = 10$ conoscendo la "sotto-chiave" $A = 9$, dovremo calcolare t come il resto della divisione di $9^{13-1-5} \times 10 = 9^7 \times 10 = 47829690$ per 13, ottenendo di nuovo $t = 12$.

[21] T. El Gamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, 31(4), 469- 472, 1985.

[22] Questi numeri sono le *radici primitive dell'unità*, introdotte da C.F. Gauss nel 1801.

[23] Esistono modi efficienti per calcolare g a partire da p: evidentemente g dipende da p, quindi se si cambia p è necessario calcolare un nuovo g.

Questo algoritmo è più efficiente dell'RSA perché i calcoli esponenziali che coinvolge possono essere svolti indipendentemente dal testo da cifrare, cioè sono in qualche senso un calcolo preliminare. La sua sicurezza sta nella difficoltà di calcolare il minimo numero intero x tale che $a = bx$, dati gli interi a e b, quello che si dice il *logaritmo discreto* di a in base b.

Si noti che le "sotto-chiavi" A e B possono essere scambiare anche pubblicamente senza invalidare la sicurezza del sistema crittografico: questa considerazione permette di modificare l'algoritmo per ottenere una comunicazione sicura attraverso un canale insicuro, ad esempio per lo scambio di chiavi.

1.5.4 Metodo Diffie-Hellman per lo scambio di chiavi

Spieghiamo come una tecnica simile a quella dell'algoritmo di El Gamal consenta uno scambio di chiavi su un canale insicuro: questa tecnica si dice *protocollo di Diffie-Hellman* dal nome dei suoi inventori, Withfield Diffie e Martin Hellman[24].

Supponiamo che Alice e Bob vogliano condividere una chiave segreta, ma che possano comunicare solo attraverso un canale di comunicazione insicuro. Per farlo avranno convenuto di utilizzare un numero primo p molto grande ed un numero g con le caratteristiche indicate nella sezione precedente: questi numeri possono essere di dominio pubblico.

Poi sia Alice che Bob scelgono ciascuno un numero a caso fra 0 e $p-2$, diciamo a e b. Alice calcola A come il resto della divisione di g^a per p e Bob calcola B come il resto della divisione di g^b per p: poi si scambiano questi numeri A e B, ma *entrambi tengono segreti i propri esponenti a e b.*

Ora ciascuno di essi può calcolare la chiave segreta: Alice calcola k come il resto della divisione di B^a per p, e Bob calcola k come il resto della divisione di A^b per p (il risultato è infatti lo stesso!) In questo modo Alice e Bob possono stabilire una chiave comune basandosi su una scelta privata di due interi a caso e su uno scambio pubblico dei dati p, g, A e B, conoscendo i quali non si riesce a risalire a k, a meno di non saper calcolare un logaritmo discreto: infatti l'unico modo che Oscar avrebbe per trovare k a partire da questi dati è di calcolare il logaritmo discreto di B in base g per ottenere b e quindi procedere al calcolo di k usando la stessa formula di Bob.

L'unico modo che Oscar può avere di forzare questo sistema è di spacciarsi per Bob con Alice e per Alice con Bob al momento dello scambio dei dati pubblici: questa possibilità può essere vanificata usando la tecnologia delle firme elettroniche, spiegata nel prossimo capitolo.

[24] W.Diffie, M.E.Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, 22(6), pp.644-654, 1976.

1.5.5 Cenni sui sistemi crittografici basati sulle curve ellittiche

La rassegna precedente non esaurisce i sistemi a chiave pubblica che possono essere utilizzati: in particolare un metodo alternativo all'RSA, forse il principale, si basa sulla teoria matematica delle *curve ellittiche*, che risale alla metà del XIX secolo. I vantaggi di questi sistemi sono non soltanto di basarsi su altri parametri per garantire la sicurezza, ma soprattutto nell'essere più compatti: dove all'RSA occorrono 1024 bit, ad un sistema a curve ellittiche, per ottenere gli stessi standard di affidabilità, ne bastano 163. Anche le chiavi sono notevolmente più semplici.

Per una discussione si rimanda ai testi in letteratura.

Capitolo 2

Firme digitali

Come il nome suggerisce, la firma digitale è un meccanismo che si può utilizzare per autenticare un documento elettronico (o comunque un flusso elettronico di dati) in modo analogo alla funzione svolta dalla firma "a mano" che si appone sui documenti cartacei: si tratta quindi di una "impronta personale" che un certo soggetto può accludere ad un documento elettronico e che si deve poter riconoscere come non contraffatta da terzi.

In questo capitolo discutiamo gli scenari tecnologici attualmente disponibili per realizzare le firme digitali ed accenniamo a qualche problematica ad essi relativa: una discussione degli aspetti legati alla effettiva realizzazione delle firme digitali sarà data nel prossimo capitolo.

2.1 Cos'è la firma digitale?

La possibilità di realizzare un sistema di firma digitale è legata all'esistenza di sistemi crittografici asimmetrici che siano sicuri.

Fissiamo le idee con un esempio: Alice vuole firmare un documento digitale d in modo che tutti possano riconoscere che è stato da lei "sottoscritto": per farlo utilizza una chiave privata h e la utilizza per produrre una firma digitale $f_{h,d}$. Se Bob vuole verificare che un documento sia stato effettivamente firmato da Alice, utilizza la chiave pubblica k di Alice per verificare che effettivamente che $f_{h,d}$ è la firma del documento d. In tutto questo un ruolo chiave (!) è giocato dal fatto che *solo Alice a Bob sanno che il documento firmato è proprio* d.

In altri termini, la firma digitale $f_{h,d}$ è sicura se Oscar non è in grado di produrla a meno che non conosca il testo in chiaro del documento d. Si noti che la firma e il documento sono entità distinte, tuttavia ai fini dell'autenticazione della firma è fondamentale la presenza del documento: in particolare non è possibile firmare un documento vuoto.

Nelle seguenti sezioni si analizzano alcuni algoritmi che realizzano il meccanismo appena spiegato, se ne discutono i limiti e le soluzioni per renderli sicuri.

2.2 Firme digitali RSA

Il sistema crittografico RSA può essere utilizzato anche per produrre e validare firme digitali: spieghiamo brevemente come.

Alice vuole firmare un documento d e per farlo deve munirsi di una chiave segreta h che è un numero (da usare come esponente) prodotto nel modo seguente: si scelgono due numeri primi p e q grandi e dello stesso ordine di grandezza ed un esponente e tale che

$$1 < e < (p-1)(q-1), \qquad \text{MCD}(e,(p-1)(q-1)) = 1$$

Dopo aver definito $n = pq$, la chiave pubblica di Alice è (n, e) e la chiave privata è un numero h da lei scelto in modo che

$$1 < h < (p-1)(q-1), \qquad eh - 1 \text{ è divisibile per } (p-1)(q-1)$$

Supponiamo ora che Alice voglia firmare un documento rappresentato da un numero d compreso fra 0 e $n-1$: per farlo definisce la sua firma $f_{h,d}$ come il resto della divisione di d^h per n.

Ora supponiamo che Bob, che conosce sia la chiave pubblica (n, e) di Alice che il documento d che è stato firmato, voglia verificare che $f_{h,d}$ è proprio la firma di Alice sul documento d: per farlo Bob calcola $(f_{h,d})^e = d^{he}$, e ne prende il resto della divisione per n, che fa esattamente d; in questo modo, conoscendo d, può verificare che la firma corrisponde a quella di Alice (se il risultato del suo calcolo non fa d vuol dire che la firma è contraffatta).

Consideriamo il solito esempio: Alice sceglie $p = 3$, $q = 5$, così che $n = 3 \times 5 = 15$, ed $e = 3$; la chiave privata h è in questo caso pure uguale a 3. Se Alice vuole comprare da Bob un *notebook* che costa 1300€, può inviargli il messaggio 13 e firmarlo nel modo seguente: $d^h = 13^3 = 2197$ diviso per 15 ha resto 7. Dunque $f_{h,d} = 7$.

Bob riceve l'ordine di acquisto da parte di Alice: sa che l'importo è di 1300€, e vuole verificare che il messaggio sia effettivamente firmato da Alice: allora calcola il resto della divisione di $(f_{h,d})^e = 7 \times 7 \times 7 = 343$ per 15, che fa appunto 13.

2.3 Sicurezza delle firme digitali RSA

Il meccanismo appena descritto per la firma digitale con l'algoritmo RSA è soggetto a diversi tipi di attacco: per prima cosa l'intruso Oscar potrebbe spacciare la propria chiave pubblica per quella di Alice, e quindi convincere Bob che il documento da lui firmato è in realtà firmato da Alice. Per prevenire questa eventualità i sistemi di forma elettronica si affidano ad una *certification authoriry*, cioè un organo (anche istituzionale) di garanzia.

Oltre a ciò, Oscar potrebbe scegliere un numero x fra 0 e $n-1$, usarlo per firmare un documento e convincere Bob che il documento è firmato da Alice: in effetti se Bob non dispone del documento firmato da Alice e se il risultato della decrittazione di Oscar corrisponde ad un documento verosimile, Bob potrebbe credere che la firma è infatti quella di Alice: questo attacco si dice *existential forgery*.

Un ulteriore tipo di attacco sfrutta una proprietà teorica delle firme digitali RSA, la *proprietà moltiplicativa*: questo vuol dire che se $f_{h,d}$ e $f_{h,d'}$ sono firme digitali valide di Alice, allora anche il prodotto $f_{h,d}f_{h,d'}$ lo è (o meglio il suo resto per la divisione per n) e precisamente è la firma del documento dd'.

Questi attacchi possono essere scongiurati usando una semplice tecnica di codifica dei documenti, che usa la cosiddetta *redundancy function*; ad esempio, un messaggio d prima di venire firmato è convertito in un numero binario b e questo numero è "raddoppiato", cioè la stringa delle sue cifre viene scritta due volte, a dare luogo ad un numero c col doppio delle cifre binarie, che viene così codificato. In questo caso non è noto applicare la *existential forgery*, ed è del tutto improbabile riuscire in un attacco che usi la proprietà moltiplicativa.

Infine va osservato che le stesse, eventuali, debolezze di un sistema RSA si possono manifestare se questo viene utilizzato per la firma digitale: ad esempio n va scelto in modo che non sia facilmente fattorizzabile, etc.

2.4 Firma digitale e codifica *hash*

Il documento che è possibile firmare con il sistema RSA richiede che il numero d che codifica il documento da firmare sia minore di n, il che, sebbene n possa essere un numero molto grande (ad esempio di 1024 bit), è una limitazione.

Per aggirare questa limitazione esiste un metodo che consente anche di difendersi dagli attacchi citati nella sezione precedente, la codifica *hash*[1]. Ci soffermeremo nella seconda parte di questo capitolo sui dettagli tecnici di

[1] La teoria delle funzioni *hash* è un argomento classico della scienza dei calcolatori: queste funzioni sono utilizzate ad esempio per gestire tabelle molto grandi di dati il cui accesso debba essere veloce.

questa tecnica di codifica, per il momento basterà dire cosa è una funzione *hash*: si tratta di un algoritmo che consente di trasformare una sequenza arbitrariamente lunga di stringhe in una stringa di lunghezza fissa. Ad esempio con una codifica *hash* si può trasformare un testo qualsiasi in un testo con, diciamo, 1024 caratteri.

In altri termini, una funzione *hash* assegna a ciascuna possibile stringa una corrispondente stringa di lunghezza fissa: è chiaro che in questo modo moltissime stringhe di lunghezza arbitraria (virtualmente infinite!) corrispondono ad una medesima stringa di lunghezza fissa, quindi le funzioni *hash* sono studiate in modo da minimizzare la frequenza con la quale una stessa stringa di lunghezza fissa corrisponde a diverse stringhe di lunghezza variabile.

Infine la caratteristica che rende le funzioni *hash* sicure da utilizzare nei sistemi di firma digitale è il fatto che *non sono invertibili*, o almeno non lo sono facilmente: questo vuol dire che da un codice *hash* $H(x)$ non è possibile ricostruire x stesso, cioè la stringa originaria di lunghezza variabile.

Torniamo alle firme digitali: Alice riesce a firmare un documento rappresentato da un numero minore di n. Se ha necessità di firmare un documento più lungo, può usare una funzione *hash* H che trasformi il suo documento in un numero minore di n che a quel punto è possibile firmare. Ovviamente, Bob deve disporre di un algoritmo di *decodifica hash*, che gli consenta di ricostruire il documento originale a partire dalla sua *codifica hash*.

Si noti che in questo modo, Oscar, anche se riesce a sostituirsi ad Alice, non riuscirà a portare a termine un attacco per *existential forgery* perché non è in grado di produrre documenti sensati se non conosce la decodifica *hash* usata da Bob, e non può conoscerla anche se conosce la codifica *hash* H perché quest'ultima, come si è detto, non è invertibile.

Anche un attacco che sfrutti la proprietà moltiplicativa è impossibile: quello che potrebbe restare ad Oscar è di utilizzare una *collisione* della funzione *hash*. In effetti, come abbiamo detto, più stringhe originali posso dare luogo alla stessa codifica *hash*, e quindi, Oscar potrebbe, anche se non conosce come decodificare un numero *hash*, trovare un documento che ha quella stessa codifica, non importa se non è quello originario di Alice, e quindi poter realizzare un *existential forgery attack*. Ma questa possibilità è esclusa se la codifica *hash* utilizzata è, come si dice, *collision resistant*, ed esistono metodi per ottenere codifiche *hash* di questo tipo, come gli standard PKCS.

2.5 Firme digitali basate su altri sistemi asimmetrici

Esiste un criterio semplice per trasformare un sistema crittografico asimmetrico in un sistema di firma digitale: i dettagli sono specificati nello standard ISO IEC/9796, del 1991: ad esempio sia RSA che l'algoritmo di Rabin che il sistema ElGamal possono dar luogo a sistemi di firma digitale. Non insisteremo su questo punto, piuttosto descriveremo un ulteriore sistema di forma digitale, il DSA, introdotto dal *National Institute of Standards and Technology* (NIST) statunitense.

2.6 Firme digitali DSA

Il *Digital Signature Algorithm* (DSA) è una variante più efficiente della versione dello schema di ElGamal per la firma digitale.

Per firmare un documento col sistema DSA Alice deve per prima cosa scegliere un numero primo q che, in binario, abbia *esattamente* 160 bit, un numero primo p che abbia almeno 512 ma non più di 1024 bit e che sia divisibile sia per 64 che per $p-1$ (questo può effettivamente essere fatto in modo efficiente).

Inoltre Alice deve scegliere, a partire da p, un numero g con le stesse proprietà specificate nella sezione (una radice primitiva dell'unità) e quindi calcola il numero

$$x = \text{ resto della divisione di } g^{(p-1)/q} \text{ per } p$$

Oltre a ciò, Alice deve scegliere un numero a caso a compreso fra 1 e $q-1$, e calcolare il numero

$$A = \text{ resto della divisione di } g^a \text{ per } p$$

A questo punto la sua chiave pubblica è (p, q, x, A), mentre la sua chiave privata è a che per essere calcolata richiede un calcolo del logaritmo discreto di numeri con 160 bit.

Ora Alice può utilizzare questi dati per firmare un documento d: a questo scopo utilizza una funzione di codifica *hash* H che trasformi d in un numero $H(d)$ compreso fra 1 e $q-1$ (questa funzione è *pubblica*), e la usa nel seguente modo: sceglie un numero a caso u compreso fra 1 e $q-1$ e calcola i numeri

$$\begin{array}{lllll} r = & \text{resto della divisione di} & g^k & \text{per } p, \\ s = & \text{resto della divisione di} & r & \text{per q ,} \\ t = & \text{resto della divisione di} & ((H(d)+as)/u) & \text{per } q \end{array}$$

Infine la sua firma digitale è la coppia di numeri (s, t).

Il procedimento di verifica della firma di Alice comporta calcoli ancor più complicati ma sorprendentemente efficienti da realizzare: in particolare, se Bob deve verificare che il documento d è stato firmato da Alice a partire dalla firma (s, t), utilizzando la chiave pubblica (p, q, g, A) e la codifica *hash* pubblica H di Alice, per prima cosa deve verificare che

$$1 \leq s \leq q - 1, \qquad 1 \leq t \leq p - 1$$

Se queste condizioni sono violate Bob è sicuro che la firma digitale è contraffatta (questa verifica permette in particolare di determinare se si è sotto un attacco con la tecnica di *existential forgery*). Altrimenti, Bob continua e calcola i seguenti valori:

$$\begin{array}{lll} R = & \text{resto della divisione di} & H(d)/t \quad \text{per } q, \\ S = & \text{resto della divisione di} & s/t \quad \text{per } q, \\ T = & \text{resto della divisione di} & g^R \times A^S \quad \text{per } p, \end{array}$$

A questo punto se il resto della divisione di T per q è diverso da r allora la firma è contraffatta, altrimenti è autentica.

La validità di questi calcoli segue da semplici considerazioni di teoria dei numeri; la loro efficienza è notevole, a dispetto di quanto sembra, ed è possibile dimostrare che l'unico attacco che realisticamente Oscar potrebbe portare a questo sistema di firma digitale può essere condotto solo sapendo calcolare il logaritmo discreto di numeri con 160 bit, cosa che attualmente nessuno sa fare.

2.7 Codifiche *hash*

Si è già specificato che una funzione *hash* H da utilizzarsi in un sistema di firma digitale deve soddisfare ai requisiti seguenti:

- H deve non essere invertibile, cioè deve essere impossibile risalire dalla stringa codificata $H(x)$ alla stringa di partenza x;
- H deve essere *collision resistant*, cioè data una stringa x deve essere impossibile trovare una stringa y cha abbia la stessa codifica *hash* di x, cioè tale che $H(x) = H(y)$;
- H deve distribuire in modo uniforme i codici che produce, il che vuol dire che deve, in media, rendere equiprobabile che due stringhe x e y diano luogo alla stessa codifica *hash* $H(x) = H(y)$. In altri termini, due stringhe a caso dovranno avere una alta probabilità di non possedere la stessa codifica *hash*.

In realtà non è stato nemmeno dimostrato che una funzione *hash* che sia *collision resistant* esista effettivamente. Tuttavia, sebbene non supportati da nessun risultato teorico consistente, esistono alcuni metodi che *sembrano* essere sicuri in questo senso. In particolare, a partire da un algoritmo sicuro di cifratura è possibile dedurre una funzione *hash* che sembra essere *collision resistant*: questo ha consentito di determinare alcune funzioni *hash* utilizzate comunemente e che si ritengono sicure, ad esempio MD5, RIPEMD-128, SHA-1, RIPEMD160 (le ultime due utilizzano numeri con al più 160 bit, a differenza delle altre che utilizzano numeri con 128 bit). Va osservato che anche MD5 non sembra più essere tanto sicura (il suo predecessore MD4 è stato dimostrato non essere più sicuro).

Infine osserviamo che esiste un algoritmo per produrre una funzione di codifica *hash* che si può dimostrare essere sicura *finché non si è in grado di calcolare efficacemente il logaritmo discreto*: questo metodo, ideato da David Chaum, Eugene van Heijst e Birgit Pfitzmann[2] non è tuttavia efficiente e quindi non consente l'implementazione effettiva di una funzione di codifica *hash* da utilizzare in applicazioni reali, ma riveste più che altro un interesse teorico.

[2]David Chaum, Eugene van Heijst, Birgit Pfitzmann, *Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer*, Lecture Notes in Computer Science 576, Springer, 1991.

Bibliografia

[1] M. Bellare, S. Goldwasser, *Lecture Notes on Cryptography*, MIT, 2001.

[2] J.A. Buchmann, *Introduction to Cryptography*, Springer, 2002.

[3] A.J. Menezes, P.C. Van Oorshot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[4] RSA Labs, Frequently Asked Questions About Today's Cryptography, RSA Sec. Inc., 2000.

[5] S. Singh, *Codici e segreti*, Rizzoli, Milano, 1999.