Andrea Pasquinucci

Aspetti di Crittografia Moderna

Da DES alla Crittografia Quantistica

001

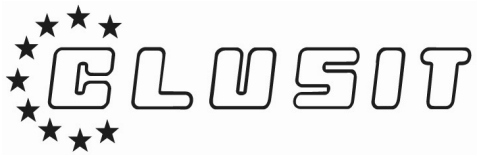
QUADERNI CLUSIT

Aspetti di Crittografia Moderna

Da DES alla Crittografia Quantistica

Andrea Pasquinucci

Comitato Tecnico Scientifico



Associazione Italiana per la Sicurezza Informatica

Quaderni CLUSIT - Ottobre 2004

CLUSIT

Il CLUSIT - Associazione Italiana per la Sicurezza Informatica, è una associazione "no profit" con sede presso l'Università degli studi di Milano, Dipartimento di Informatica e Comunicazione, fondata nel luglio 2000.

Le principali attività del CLUSIT sono:

- la diffusione di una cultura della sicurezza informatica rivolta alle Aziende, alla Pubblica Amministrazione ed ai cittadini;
- l'elaborazione sia a livello comunitario che italiano di leggi, norme e regolamenti che coinvolgono la sicurezza informatica;
- la definizione di percorsi di formazione per la preparazione e la certificazione delle diverse figure professionali operanti nel settore della sicurezza ICT;
- la promozione dell'uso di metodologie e tecnologie che consentano di migliorare il livello di sicurezza delle varie realtà.

Nell'ambito della Sicurezza Informatica, i soci del CLUSIT sono rappresentativi dell'intero "sistema Paese", in particolare della ricerca, dell'industria, del commercio, del settore bancario ed assicurativo, della Pubblica Amministrazione, della Sanità, dei servizi, delle telecomunicazioni e di Internet.

Copyright e Disclaimer

Presentazione del Presidente del CLUSIT

La crittografia è, in questo momento, lo strumento più formale e, in relazione a tutta una serie di attacchi informatici, il più efficace di cui dispone la nostra comunità. Una conoscenza anche non approfondita delle leggi che regolano questa disciplina e dei suoi principali risultati, dovrebbe oggi essere parte del bagaglio culturale di ogni professionista della sicurezza informatica. Per il suo aspetto particolarmente rigoroso e formale però la crittografia non è facilmente accessibile a molti, che memori delle fatiche spese, a suo tempo, sui libri di matematica e algebra si mantengono ad una opportuna distanza da questa disciplina, nutrendo al tempo stesso una notevole ammirazione per i risultati che periodicamente i crittografi ci propinano.

Il presente contributo nasce con l'obiettivo principale di consentire proprio a queste persone di avvicinarsi al modo della crittografia e carpirne i suoi segreti. Per svolgere questa missione il CLUSIT ha deciso di chiedere ad Andrea Pasquinucci (socio CLUSIT) e persona con un notevole bagaglio tecnico-scientifico (in settori però del tutto estranei alla crittografia, o meglio che per ora sembrano esserlo), di raccontarci la crittografia a modo suo.

Ne è uscito questo contributo, originale nell'impostazione che sicuramente lo differenzia da tutti i testi di crittografia. In questo testo Andrea non si dilunga in noiose spiegazioni di dettagli implementativi di algoritmi e protocolli, ovviamente fondamentali per chi è interessato alla loro implementazione ma decisamente superflui per chi non ha questo interesse, e riesce a mantenere l'esposizione degli argomenti trattati ad un ottimo livello di astrazione, consentendo al lettore di cogliere gli aspetti e i principi di riferimento, e quindi cogliere il perché di certe scelte.

Sfruttando poi la formazione di Fisico di Andrea, abbiamo osato quello che sinora, credo, nessuno in Italia ha anche solo tentato di fare. Un testo divulgativo sulle nuove frontiere della crittografia, che vedono una convergenza e fusione tra i principi della crittografia classica e la meccanica quantistica, per dare origine ad una disciplina fortemente innovativa quale la crittografia quantistica.

Anche in questo caso il testo è particolarmente introduttivo e riesce a far cogliere gli aspetti e i principi fondamentali senza cadere in divagazioni o descrizione di dettagli superflui.

Ovviamente non poteva mancare anche un accenno ai calcolatori quantistici, che in questo momento sono la principale spina nel fianco della crittografia asimmetrica, ed il cui avvento avrebbe un effetto davvero rivoluzionario sull'intero settore ICT.

In sostanza un testo interessante, dove l'aspetto divulgativo è particolarmente curato senza scadere nell'approssimazione o pressappochismo. Un viaggio nella crittografia a partire dal V secolo a.c. per arrivare ad un futuro che non sappiamo ancora se ci sarà.

Non ci resta che consigliarne la lettura a tutti i nostri soci, che sicuramente sapranno apprezzarlo, sia che conoscano o non conoscano già la disciplina trattata.

Buona Lettura

Prof. Danilo Bruschi

Abstract

In questo documento viene fatta una breve rassegna su vari aspetti della Crittografia moderna. Partendo dai principali elementi teorici della crittografia del XX secolo, si passa a DES, AES, RSA per poi dare uno sguardo a cosa potrebbe succedere nel prossimo futuro e finire considerando alcuni tra gli aspetti più innovativi provenienti dalla ricerca ma che ormai si affacciano all'implementazione commerciale, quale la Crittografia Quantistica.

L'Autore

Andrea Pasquinucci

PhD in Fisica, membro del Comitato Tecnico Scientifico CLUSIT. E' un esperto in Sicurezza Informatica e si occupa prevalentemente di crittografia, di sicurezza delle reti e dei sistemi operativi. Con un esteso background di ricerca universitaria all'estero ed in Italia, partecipa a progetti di ricerca finanziati dall'Unione Europea ed insegna in corsi universitari e di specializzazione. Svolge attività professionale e di consulenza sia presso aziende che per fornitori di servizi di sicurezza informatica e telecomunicazioni.

Indice

0.1 Introduzione

Queste note sono state pensate come una rassegna tra il tecnico e l'illustrativo di alcuni aspetti della crittografia moderna. Non vi è nessuna pretesa di completezza né dal punto di vista storico né da quello scientifico. Si è cercato però di seguire un filo sia logico che storico nella esposizione di alcuni concetti fondamentali in modo che questi possano costituire la base per un futuro individuale approfondimento dell'argomento.

Queste note si rivolgono a persone con un minimo di conoscenze nell'ambito dell'informatica e di dimestichezza con i concetti base sia di matematica che di logica.

A parte due capitoli introduttivi, abbiamo diviso questo documento in tre parti secondo il seguente schema.

Nella prima parte, partendo da alcune considerazioni storiche, vengono descritti i principali algoritmi crittografici attualmente in uso. La scelta degli algoritmi di cui parlare è soggettiva e limitata dallo spazio, pensiamo però che possa essere indicativa dello stato dell'arte attuale.

Nella seconda parte vengono illustrate alcune applicazioni degli algoritmi crittografici presentati ed alcuni protocolli a loro associati. Questo porterà a considerare alcuni dei principali problemi aperti connessi alla crittografia.

Infine nella terza parte viene fatto un salto ancora più avanti considerando aspetti molto più vicini alla ricerca scientifica di punta descrivendo brevemente la Crittografia Quantistica. Questa è una proposta molto nuova nel campo della crittografia, si basa sull'utilizzo della fisica delle particelle elementari e potrebbe avere degli interessanti sviluppi nei prossimi anni.[1]

1 Va comunque notato che i primi prototipi commerciali della Crittografia Quantistica sono apparsi sul mercato alla fine del 2003.

0.2 Crittografia non sempre vuol dire Sicurezza

Uno dei principali problemi della Crittografia è la comune percezione che con la sua adozione qualunque problema di sicurezza sia risolto. Purtroppo vale quasi il contrario, l'adozione di tecniche e protocolli crittografici senza le adeguate conoscenze, al di fuori di un preciso ambito di applicazione od in maniera sommaria, porta ad un falso senso di sicurezza e quindi al risultato opposto: sentirsi sicuri quando non lo si è.

Azzardando un paragone, sarebbe come dire che l'invenzione della ruota ha risolto per sempre tutti i problemi di trasporto dell'uomo. Molti direbbero che basta guardare al traffico nelle nostre città per dubitare di questa affermazione. Ovviamente, l'invenzione della ruota è di fondamentale utilità per l'uomo e per le sue attività, ma non ha risolto una volta per tutte il problema. A ben vedere, la ruota è solamente uno strumento che permette all'uomo di fare cose che altrimenti gli sarebbero impossibili o molto più difficili a realizzare. In pratica il problema astratto degli spostamenti di persone e cose è rimasto tale, sono solo cambiati gli strumenti tecnici a disposizione per poterlo affrontare.

Proseguendo nel nostro paragone, la crittografia è solamente uno strumento tecnico che ci permette di affrontare alcune problematiche di sicurezza in modo efficace e, quando possibile, matematicamente provato. Dipende poi dall'implementazione se la crittografia fornisce o meno un qualche livello di sicurezza.

La realtà inoltre ci insegna che molto spesso anche quando la crittografia è utilizzata a dovere, non sempre aiuta a risolvere i problemi di sicurezza dato che questi si trovano altrove. Bisogna sempre ricordarsi del principio fondamentale valido in qualunque problematica di sicurezza: *un sistema è sicuro quanto il suo punto più debole*. E bisogna ricordarsi di applicare questo principio al sistema visto in modo globale, e non solamente ad una sua piccola parte.

0.3 La Crittografia è difficile

Un altro aspetto che va tenuto in considerazione è che la crittografia è difficile, sotto molti punti di vista.

Da un punto di vista **teorico**, le tecniche crittografiche sono, potremmo dire da sempre, basate sulla matematica più avanzata. Questo non per un capriccio o per cercare di realizzare una forma di Security-by-Obscurity, ma molto semplicemente perché le tecniche matematiche più avanzate offrono delle possibilità altrimenti non esistenti. Non è facile quindi discutere dei dettagli di crittografia senza addentrarsi nella matematica più avanzata. In queste note cercheremo comunque di illustrare al lettore i principali risultati e tecniche matematiche adottate dalla crittografia, senza però entrare nei dettagli matematici dei vari algoritmi.

Vogliamo sottolineare che non è facile anche per gli addetti ai lavori, matematici di professione, creare dei nuovi algoritmi crittografici. Spesso all'idea ed alla formulazione di un nuovo algoritmo segue un lungo lavoro di messa a punto, correzione ed eliminazione di debolezze e possibili punti di attacco, ed ancora più spesso abbandono del tentativo e sviluppo di un nuovo algoritmo sulla base delle esperienze ed errori precedenti. Infatti per poter avere un algoritmo *sicuro,* bisogna essere in grado di dimostrare matematicamente quale è l'attacco più efficiente contro l'algoritmo e che questo attacco è impraticabile. Questo lavoro oggi viene fatto in pubblico poiché, con la sola possibile eccezione di alcune agenzie di servizi segreti, solo la collaborazione a livello mondiale dei massimi esperti in crittografia permette di raggiungere questi risultati. Purtroppo spesso vengono annunciati e venduti algoritmi creati con il fai-da-te, rigorosamente “segreti” e “sicurissimi” che promettono cose mirabolanti. Bruce Schneier, uno dei massimi esperti di sicurezza informatica al mondo, nella sua newsletter mensile[2] dedica una sezione, chiamata *Doghouse*, a chi propone nuovi, incredibili e spesso segretissimi algoritmi di crittografia che ovviamente hanno ben poche probabilità di poter resistere ad un attacco di un crittoanalista professionista.

Da un punto di vista **implementativo**, si pongono due tipi di problemi. Il primo è di trovare il modo di implementare in maniera corretta gli algoritmi teorici in Hardware o Software, con le usuali richieste di ottime prestazioni e minime risorse utilizzate, già queste spesso difficili da soddisfare. Realizzare ciò non è facile se quello che si vuole implementare è un algoritmo basato su nozioni avanzate di matematica. Inoltre al momento dell'implementazione compaiono molti fattori non presenti nella formulazione teorica dell'algoritmo. Possiamo citare come esempi la sicurezza stessa del codice quando è in esecuzione, la possibilità che qualcuno legga i dati nella memoria di massa, la sicurezza dei file di configurazione e di tutte quelle attività normali per un programma informatico ma che non compaiono per nulla nell'algoritmo crittografico. E' ovvio che una sola debolezza della implementazione anche in un fattore prettamente tecnico quale il modo di accedere alle risorse hardware dello elaboratore può ridurre a zero il livello di sicurezza offerto.

Da un punto di vista degli **utilizzatori** vi è poi il problema di comprendere come uti-

2 http://www.schneier.com/crypto-gram.html

lizzare il programma o l'hardware che implementa un algoritmo crittografico. Se non si è a conoscenza dell'algoritmo o del protocollo crittografico che si vuole usare, come è possibile scegliere ed utilizzare in maniera corretta per i propri scopi i parametri, le modalità ecc. che sono possibili? Quali sono i rischi che si corrono se si *clicca* sul bottone sbagliato?

PARTE 1 – ALGORITMI DI CRITTOGRAFIA

1.1 Una breve regressione storica

Non è questa la sede per una introduzione storica alla crittografia, proponiamo solo un breve riassunto (del tutto soggettivo) nella Tabella 1. Vi sono molti testi ben fatti tra cui ricordiamo [Singh] e [Kahn]. E' necessario però dare alcune indicazioni su come si sia giunti al punto in cui siamo oggi anche per capire meglio cosa potrebbe succedere domani.

Il problema della protezione delle comunicazioni è sempre stato sentito dall'uomo, in particolare nell'ambito di attività belliche e commerciali. Sin dall'antichità sono stati ideati metodi per poter inviare informazioni a distanza in maniera *sicura*. Il problema è quello che ora chiamiamo **confidenzialità**, ovvero trasferire un messaggio in un formato tale da renderlo inintelligibile a chiunque lo intercetti, e nel caso anche a chi lo trasporti. Nell'antichità non si faceva una distinzione netta tra quelle che oggi chiamiamo **crittografia** e **steganografia**, ovvero rispettivamente l'arte del cifrare e del nascondere o confondere. La differenza principale tra i due metodi può essere riassunta nel seguente modo: la crittografia modifica il messaggio, ad esempio ri-scrivendolo in un alfabeto ignoto, in modo che sia impossibile capirne il contenuto anche se è ovvio a chiunque che si è in presenza di un messaggio. Al contrario la steganografia non cifra il messaggio ma lo nasconde in modo tale che in pratica solo chi è a conoscenza del nascondiglio lo possa trovare. Gli esempi classici sono il cifrario di Cesare che discuteremo in dettaglio più avanti, e per la steganografia inchiostri invisibili che riappaiono in determinate condizioni (i.e. temperatura), puntini quasi invisibili sopra certi caratteri di un testo normale, variazioni nelle *gambe* delle lettere scritte a mano in corsivo ecc. ecc. La versione moderna più conosciuta della steganografia è il nascondere messaggi all'interno di immagini elettroniche giocando sulla codifica numerica dei colori di ogni punto. In queste note non ci occuperemo di steganografia ma solo di crittografia.

Bisogna poi anche distinguere tra **codici** e **cifrari**. In linea generale un codice è un modo di assegnare convenzionalmente a parole, gruppi di parole o frasi, un numero, una parola od una breve frase. Un codice rappresenta un concetto, una informazione con un simbolo scelto a rappresentarlo. Pertanto un codice è strettamente legato al contesto in cui è utilizzato e di solito può rappresentare un limitato numero di concetti ed informazioni. Un codice potrebbe essere assegnare alla parola ROSSO il significato '*devi andare a Roma'*, ed alla parola VERDE il significato '*devi andare a Milano'*. Un cifrario invece è completamente indipendente dal contesto, dalle informazioni che si vogliono comunicare, ed agisce direttamente ad esempio sulle lettere dell'alfabeto. Un qualunque simbolo dell'alfabeto cifrato non ha nessun significato di per se, è solamente un simbolo in un altro alfabeto in cui è stata rappresentata l'informazione. In un cifrario ogni simbolo del testo da cifrare viene trasformato in un altro simbolo nello stesso od in un altro alfabeto, utilizzando una procedura matematica detta algoritmo crittografico. In queste note ci occuperemo di cifrari, anche se spesso in pratica i codici sono utilizzati insieme ai cifrari per rendere ancora più confuso il testo cifrato.

Ovviamente le tecniche di protezione delle informazioni hanno seguito passo passo lo sviluppo dei canali di comunicazione. Finché le comunicazioni erano rare ed eccezionali, pochi erano in grado di scrivere e leggere e le tecniche di comunicazione erano totalmente manuali, anche le tecniche di protezione non hanno necessitato di grandi sviluppi.

Però in particolare a partire dalla fine del secolo XIX, i canali, i metodi e le tecniche di comunicazione, la comunicazione scritta manuale, automatica ed elettronica, hanno rivoluzionato completamente il modo di inviare, distribuire e ricevere informazioni. Questo ha richiesto uno sviluppo adeguato di tecniche per proteggere le informazioni visto che è diventato più facile inviare informazioni, ma al contempo è diventato più facile anche intercettarle e decifrarle. Si è quindi reso necessario lo sviluppo della teoria della crittografia per cercare di soddisfare le richieste di cifrari più robusti ed al contempo di tecniche per decifrare le comunicazioni del nemico.

Infatti i maggiori clienti della crittografia sono sempre stati i militari. Battaglie e guerre sono spesso state vinte o perse anche grazie alla possibilità di comunicare velocemente con le proprie truppe in modo sicuro ed al contempo di intercettare e decifrare le comunicazioni del nemico. E' nata una caratterizzazione più precisa del termine crittografia. Ad essere formali, si dovrebbe indicare

- con **Crittologia** la scienza dei cifrari in generale
- con **Crittografia** l'arte del cifrare
- con **Crittoanalisi** l'arte del decifrare.

Spesso però il termine Crittografia viene utilizzato anche al posto di Crittologia.

Se da una parte le richieste dei militari hanno portato ad un grande sviluppo della crittografia e della crittoanalisi, basti pensare alla seconda guerra mondiale ed al ruolo avuto nella stessa dalla decifrazione dei messaggi Enigma[3] da parte degli Inglesi, dall'altra questo ha voluto dire che sino a pochi anni fa i principali risultati teorici in crittografia erano segreti, ignoti ai più anche negli ambienti universitari. Non è quindi un caso che l'algoritmo noto oggi con il nome di RSA[4] del 1977 fosse stato scoperto già alla fine degli anni '60 da scienziati del servizio segreto inglese GCHQ (Government Communications HeadQuaters). Come vedremo nel capitolo 1.3.2, la mancanza di informazioni ha prodotto una lunga polemica a proposito di DES,[5] dal 1975 al 1992, che si è risolta solo con la ri-scoperta pubblica delle tecniche di crittoanalisi differenziale che erano già note venti anni prima alla NSA americana.[6]

Ma la seconda guerra mondiale è anche stata in qualche modo il momento cruciale per la nascita della crittografia moderna. Già dalla fine dell'ottocento era iniziato lo sviluppo di tecniche per produrre macchine per cifratura automatiche che è culminato nelle macchine adottate nella seconda guerra mondiale, tra cui la appena citata Enigma. La guerra ha ovviamente accelerato lo sviluppo di queste macchine ed al contempo delle tecniche per decifrare i messaggi cifrati con esse. In particolare, per decifrare i messaggi tedeschi prodotti dalle macchine Enigma, gli Inglesi raccolsero

3 Enigma era una macchina a rotori utilizzata dai tedeschi nella seconda guerra mondiale per cifrare le comunicazioni.

4 Dai nomi dei suoi inventori: R. Rivest, A. Shamir e L. Adleman; questo algoritmo verrà descritto nel capitolo 1.3.5.

5 Data Encryption Standard, forse il più importante algoritmo crittografico della seconda metà del XX secolo.

6 National Security Agency, agenzia dello stato Americano che si occupa dei principali problemi di sicurezza e di crittografia.

un gruppo di scienziati divenuti poi molto famosi, tra cui Alan Turing, considerato il padre dell'informatica. E' inevitabile che gli sforzi, le idee e le collaborazioni di quegli anni si siano poi riversati, al loro ritorno negli ambienti universitari, nella ricerca universitaria che lentamente, anche perché molti dei migliori hanno spesso deciso di lavorare per i militari, ha portato allo sviluppo della attuale crittografia.

Ci si può porre la domanda di quale sia la situazione oggi, ovvero la crittografia pubblica di fonte universitaria è al passo con quanto conosciuto negli ambienti militari più avanzati? Non conosciamo la risposta e vi sono molti pareri discordanti, certo i militari non ci forniscono alcuna indicazione. Però, da vari segnali quali il riconoscimento e l'adozione di AES[7] anche da parte dell'esercito e della amministrazione americana nel 2003, possiamo immaginare che ormai il divario fra i due mondi si sia chiuso. Oppure, come i più pessimisti sostengono, è talmente ampio che non siamo neanche in grado di rendercene conto ?

Un altro aspetto da sottolineare è che lo sviluppo dei mezzi e della comunicazione informatica ha reso la crittografia uno strumento praticamente di tutti, e non solo per militari, governi o grandi aziende. E' inevitabile che questo interesse generale abbia portato ad un rapidissimo sviluppo, per quanto possibile in questo campo, anche grazie all'interesse e coinvolgimento di molti.

Tornando alla teoria della crittografia, un passo fondamentale fu nel 1883 l'enunciazione da parte di Kerckhoffs del principio che

> *la sicurezza della crittografia si basa sulla segretezza della sola Chiave, mentre l'algoritmo prima o poi diviene pubblico.*

Anche se oggi possiamo considerare questo punto ovvio, la distinzione tra algoritmo e chiave, ed il ruolo giocato da entrambi sono di fondamentale importanza.

Inoltre la crittografia non è solo cifratura per garantire la confidenzialità, ma si è capito che la crittografia ed i suoi algoritmi e protocolli sono utili per affrontare i seguenti problemi:

- Integrità
- Autenticità
- Confidenzialità

a cui possiamo aggiungere anche il

- Non Ripudio.

Si è quindi passati dalla seguente definizione intuitiva di crittografia

> *l'arte di rendere un messaggio inintelligibile a qualunque persona non autorizzata a leggerlo*

alla seguente e più appropriata definizione tecnica

> *lo studio di tecniche matematiche connesse ad aspetti della sicurezza delle informazioni quali integrità, autenticità e confidenzialità.*

Da questa definizione segue che la Crittografia interagisce con due campi più generali:

- La Teoria delle Informazioni

7 Advanced Encryption Standard, recente algoritmo crittografico che dovrebbe sostituire DES; verrà descritto nel capitolo 1.3.3.

- La Sicurezza (Informatica).

Con il primo nel senso che le informazioni sono l'oggetto su cui agisce la crittografia, ed il secondo nel senso che la sicurezza è il fine che la crittografia vuol raggiungere.

Data	***Evento***
V A.C.	(Grecia) Steganografia (steganos=coperto e graphein=scritto) e Crittografia (crypto=segreto e graphein=scritto); lo Scytale è un cifrario a trasposizione a mo' di cintura
IV A.C.	(India) nel Kama-Sutra è insegnata l'arte dello scrivere in segreto (cifrario a sostituzione)
I sec.	Cifrario di Cesare (monoalfabetico)
VII – XIII sec.	(Arabia) sviluppo della crittoanalisi e rottura dei cifrari monoalfabetici
IX sec.	(Arabia) crittoanalisi tramite lo studio delle frequenze di apparizione delle lettere
XIV sec.	Rinascita della crittografia in Europa (Inghilterra, Francia, Italia)
1586	Vigenère pubblica il *Traicté de Chiffres*, con un cifrario polialfabetico che resiste sino al XIX secolo
XVII sec.	(Francia) cifrari homofonici (Il *Grande Cifrario di Luigi XIV*, di padre e figlio Rossignol, sostituzione di sillabe con numeri, “rotto” nel 1893 da Bazeries)
XVIII sec.	Il Cilindro di Jefferson, prima macchina a rotori, 36 cilindri con 26 lettere ciascuno
1854	Babbage “rompe” il cifrario di Vigenère (non pubblicato)
1863	Kasiski “rompe” il cifrario di Vigenère
1883	Kerckhoffs pubblica *La Cryptographie Militaire* ove enuncia il principio di separazione tra chiave ed algoritmo
1918-1919	Prime macchine a rotori di Koch, Hebern, Damm, Hagelin, Scherbius & Ritter
1926	Vernam introduce il One-Time-Pad (OTP)
1927	Entrano in funzione i primi modelli di Enigma (Scherbius & Ritter)
1940	Turing e collaboratori “rompono” Enigma a Bletchley Park
1942-1950	L'uso ripetuto della stessa chiave per OTP da parte dei Russi permette agli americani di decifrare alcuni messaggi e scoprire una rete di spie (progetto Venona http://www.odci.gov/csi/books/venona/venona.htm)
1948	Shannon introduce la moderna Teoria della Infomazione
1949	Shannon dimostra matematicamente la sicurezza dell'OTP se usato correttamente

Data	***Evento***
1975	Pubblicazione di DES
1976	Diffie e Hellmann (ed indipendentemente Merkle) enunciano la crittografia a Chiave Pubblica
1977	Pubblicazione di RSA
1991	Philip Zimmermann pubblica *Pretty Good Privacy* (PGP)
2001	Pubblicazione di AES

Tabella 1 Alcuni eventi (soggettivamente) importanti nella storia della crittografia

1.2 Elementi di base di Crittografia

Si può riassumere Il procedimento generale della crittografia come l'uso di un algoritmo matematico per operare una trasformazione su di una sequenza di caratteri. Questa trasformazione può essere reversibile o meno, nel primo caso è possibile riottenere il testo originario applicando la trasformazione inversa, nel secondo caso questo non è possibile. Anche se potrebbe sembrare strano utilizzare delle trasformazioni non invertibili, vedremo che in realtà esse giocano un ruolo molto importante nei protocolli crittografici.

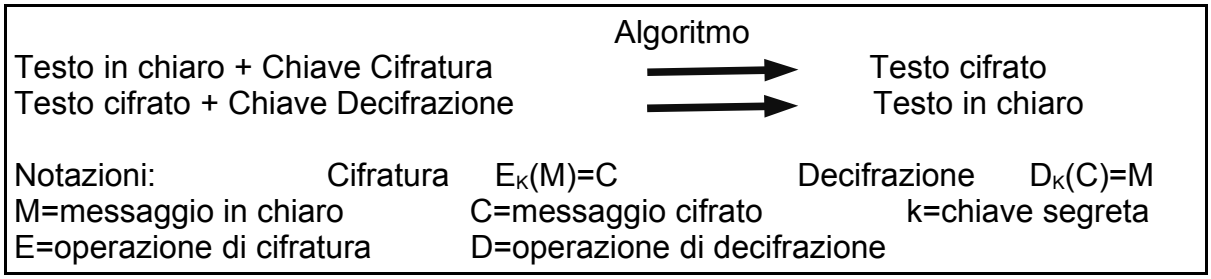
La trasformazione del testo dipende dal valore di una chiave segreta, e la segretezza di questa chiave è il punto cruciale della sicurezza offerta dalla crittografia. Per cifrare un testo si usa

- Un **algoritmo crittografico** pubblico
- Una **chiave segreta**

La sicurezza risiede nella

- bontà dell'algoritmo (con verifica pubblica o da sorgente fidata) e
- nella segretezza della chiave.

Pertanto un algoritmo crittografico può essere descritto come una scatola, di cui studieremo il contenuto, che ha in ingresso da un lato un testo in chiaro ed una chiave segreta, ed in uscita dal lato opposto un testo cifrato. L'operazione inversa è data da un algoritmo che ha in ingresso da un lato un testo cifrato ed una chiave segreta, ed in uscita dal lato opposto un testo in chiaro.



Come vedremo nel proseguio, elementi fondamentali sono:

- Per generare le chiavi segrete si usano spesso **numeri casuali** (bisogna notare che sugli elaboratori è impossibile generare in software dei numeri casuali secondo la definizione matematica del termine, questo semplicemente perché gli elaboratori sono dei sistemi finiti e deterministici, possono rappresentare solo un numero finito di numeri ed effettuare calcoli determinati, e quindi non solo una sequenza di numeri è univocamente determinata dall'algoritmo che la produce, ma prima o poi qualunque sequenza di numeri si ripeterà; sugli elaboratori è possibile, anche se non è facile, generare sequenze di numeri **pseudo-casuali** le cui caratteristiche di impredicibilità si avvicinano molto a quelle di una sequenza di numeri veramente casuali; bisogna sottolineare che la generazione in software di numeri pseudo-casuali su elaboratori è un processo molto delicato ed è un fattore impor-

tantissimo nella sicurezza della implementazione degli algoritmi crittografici; d'altra parte esistono generatori hardware di numeri casuali con buone caratteristiche di impredicibilità.)

- Alcuni algoritmi Asimmetrici utilizzano anche **numeri primi**.[8]

Per essere concreti, e non perderci in discussioni astratte, introdurremo le proprietà fondamentali degli algoritmi crittografici studiandone i più semplici esempi.

1.2.1 Il Cifrario di Cesare

Consideriamo per primo un cifrario semplicissimo, ma del tutto insicuro.[9] Il cifrario di Cesare, adottato proprio da Giulio Cesare in alcune delle sue campagne militari, si applica ad esempio all'alfabeto italiano che prendiamo per questo esempio di 21 lettere.

Il cifrario di Cesare è definito da

- *Algoritmo*: sostituisci ogni lettera con la N-esima lettera successiva dell'alfabeto, ricominciando da capo quando si arriva in fondo (ovvero ricominciando dalla A quando si arriva alla Z); in modo più formale, assegnamo un numero da 0 a 20 a tutte le lettere dell'alfabeto (A=0 ... Z=20), dato il carattere X, il corrispondente carattere cifrato Y è dato da **Y=X+N mod(21)** ove l'operazione mod(21) può essere definita come *sottrarre/sommare 21 alla somma X+N sino a che il risultato è compreso tra 0 e 20 (inclusi)*.
- *Chiavi*: chiave di cifratura **N**, chiave di de-cifratura **-N** (l'algoritmo di de-cifratura è lo stesso che per la cifratura).

Esempio: N=3 A => D, B => E, ..., Z => C

E' facilissimo rompere il cifrario di Cesare e qui indichiamo alcuni modi.

Notiamo per prima cosa che il cifrario di Cesare cambia una lettera sempre nella stessa nuova lettera, ad esempio nel caso appena citato di N=3 la lettera E viene sempre sostituita con la lettera H. Pertanto nel testo cifrato la frequenza della lettera H è la stessa della frequenza della lettera E nel testo in chiaro. Il crittoanalista può quindi calcolare la frequenza di ogni lettera nel testo cifrato, paragonarla con la frequenza nota nella lingua originale, e così ottenere la tabella di conversione. Per un crittoanalista di medie capacità è facile fare molto di più. Supponiamo che invece dell'algoritmo di Cesare si abbia una tabella di trasformazione casuale del tipo A=>F, B=>Z, ..., Z=>H in cui ogni lettera viene sostituita con una altra a caso, ma sempre con una sostituzione univoca, ovvero non ci sono due lettere che vengono sostituite dalla stessa. Supponiamo inoltre che il crittoanalista non sappia in quale lingua è stato scritto il messaggio originale. Anche in questo caso la decifrazione è facile. Basta calcolare la frequenza di tutte le lettere cifrate, e confrontando queste e

8 I numeri primi sono divisibili, senza resto, solo per 1 e per se stessi.

9 Ai giorni nostri una qualunque persona con qualche esperienza in parole crociate è in grado di rompere il cifrario di Cesare.

la frequenza media di ogni lettera cifrata con i valori noti per le varie lingue, scoprire sia la lingua in cui è scritto il messaggio originale che il messaggio stesso.

Il problema fondamentale del cifrario di Cesare è che il testo cifrato non è per nulla una sequenza quasi casuale di lettere.

Ma il cifrario di Cesare ha anche altri problemi. Anche se il testo cifrato fosse una sequenza quasi casuale di lettere, lo spazio delle chiavi è troppo piccolo. Infatti i possibili valori della chiave N sono, nell'esempio fatto, solo 20 (da 1 a 20). E' facile perciò provare tutte le 20 chiavi, produrre 20 testi e scegliere tra questi quello giusto.

Infine, la chiave N viene riutilizzata per tutte le lettere del testo da cifrare, il che ovviamente è una concausa dei problemi precedenti. Per dare una indicazione di quanto importante sia questo punto basta considerare il cifrario di Vigenère. Una delle caratteristiche fondamentali di questo cifrario detto polialfabetico, è di avere t chiavi N1 ... Nt e di sostituire il primo carattere utilizzando la chiave N1, il secondo utilizzando la chiave N2 e così via sino al t-esimo carattere sostituito utilizzando la chiave Nt. Il carattere (t+1)-esimo viene sostituito di nuovo con la chiave N1 e così via. Sia ha quindi un *periodo* t nel ciclo delle sostituzioni. La chiave segreta è formata dal periodo t e dalle chiavi N1 ... Nt. E' chiaro che calcolarsi la frequenza con cui compaiono i caratteri nel testo cifrato non aiuta più di tanto alla decifrazione del messaggio. Infatti il cifrario di Vigenère è rimasto valido per circa 3 secoli, sino a quando Babbage e Kasiski nella seconda metà del XIX secolo hanno travato la maniera di calcolare, a partire dal testo cifrato, il periodo t adottato, e una volta trovato questo è facile ottenere, come se fossero t cifrari di Cesare, le chiavi N1 ... Nt.

1.2.2 One-Time-Pad

Questo algoritmo, il linea di principio semplicissimo, è stato studiato inizialmente da Vernam nel 1926. Vernam considerò l'applicazione dell'algoritmo al caso di alfabeti del linguaggio naturale, quale Italiano o Inglese, mentre noi per semplicità ed anche per interesse informatico, lo consideriamo in forma binaria. Le considerazioni sono comunque valide anche per un alfabeto di 21 o 26 caratteri, sostituendo ad esempio l'operazione XOR che vedremo tra un attimo con la somma modulo 21 o 26, ove ad ogni carattere si assegna un numero corrispondente alla propria posizione nell'alfabeto (partendo da 0).

Il One-Time-Pad (OTP) è definito da

- *Algoritmo*: XOR di ogni bit del messaggio con il corrispondente bit della chiave
- *Chiave*: numero casuale lungo quanto il messaggio.

Prima di considerare le proprietà dell'OTP, vediamo un semplice esempio. L'operazione di XOR è definita da:

0 XOR 0 = 0 0 XOR 1 = 1 1 XOR 0 = 1 1 XOR 1 = 0 che implica (a XOR a) = 0

E l'algoritmo ci dice che per cifrare e decifrare dobbiamo

Messaggio-in-chiaro XOR Chiave ==> Messaggio-Cifrato
Messaggio-Cifrato XOR Chiave ==> Messaggio-in-chiaro

Quindi abbiamo come esempio banale

Messaggio in chiaro:	`1011010001`
Chiave:	`0110100011`
XOR	━━━━━━━━━━
Messaggio Cifrato:	`1101110010`

La verifica che l'operazione XOR serve sia a cifrare che a decifrare è immediata. Si noti anche che prendendo l'XOR del messaggio cifrato con il messaggio in chiaro si ottiene la chiave segreta.

Shannon, l'ideatore della teoria dell'Informazione, nel 1949 ha mostrato che con l'algoritmo OTP se la chiave segreta è una stringa veramente casuale di bit, allora anche il messaggio cifrato è una stringa casuale di bit. Chi attenta a decifrare senza avere la chiave un messaggio cifrato con OTP, non ha altra possibilità che provare tutte le chiavi ed ottenere tutti i possibili messaggi di quella lunghezza. Si noti che il numero delle chiavi è dato da 2^N ove N è il numero di bit del messaggio. Per un messaggio di una pagina di un qualunque testo, N è un numero ragionevolmente grande e 2^N è un numero enorme. Quindi, anche se fosse possibile generare in un tempo (e spazio) ragionevole tutti i messaggi, sarebbe impossibile selezionare quello giusto poiché comparirebbero tutte le possibili variazioni di ogni messaggio con tutti i suoi possibili significati.[10]

Questo è però vero *solo se* la chiave segreta è usata una sola volta. Infatti se si riutilizzasse la stessa chiave, allora incomincerebbero ad apparire ripetizioni e somiglianze tra diversi messaggi cifrati, che sarebbero ovviamente da imputare alla stessa chiave. Da qui, con un sufficiente numero di messaggi cifrati con la stessa chiave, risulta teoricamente possibile risalire alla chiave segreta e quindi rompere completamente la sicurezza dell'OTP.

L'OTP è l'unico algoritmo per cui sia stato possibile dimostrare matematicamente la sicurezza assoluta ma *solo se* la chiave è veramente casuale ed usata una volta sola.

L'utilizzo dell'OTP con chiavi pseudo-casuali è possibile in pratica, anche se in questo caso non vale più la dimostrazione matematica, ma bisogna stare molto attenti alle proprietà dei numeri pseudo-casuali adoperati. Ad esempio, il periodo del generatore pseudo-casuale deve essere superiore alla lunghezza della chiave richiesta, non bisogna riutilizzare lo stesso generatore nella stessa configurazione per creare chiavi diverse, e così via. Il problema è che se non utilizzato correttamente un generatore di numeri pseudo-casuali può generare (sequenze) di numeri che sono correlate. Visto che la sicurezza dell'OTP si basa sulla pura casualità della chiave, qualunque correlazione fra chiavi diverse può favorire la rottura del cifrario. Non solo, spesso i generatori pseudo-casuali hanno bisogno di un numero iniziale, detto *seed*,

10 Si confronti questo caso a quello del Cifrario di Cesare ove è facile trovare il messaggio originale tra i 20 possibili.

ed una volta specificato il seed la sequenza di numeri prodotta dal generatore è fissata. In questo caso la sicurezza dell'OTP si riduce alla sicurezza del seed, infatti se un attaccante è a conoscenza del generatore utilizzato può o indovinare il seed utilizzato o provare tutti i possibili seed e ottenere la chiave segreta.

Un altro problema dell'OTP è che se l'attaccante è a conoscenza di parte del testo in chiaro (come ad esempio headers, caratteri di formattazione ecc.), facendo l'XOR della parte di testo in chiaro a lui nota con la corrispondente parte del testo cifrato, egli può facilmente ottenere i bit della chiave segreta che corrispondono alla parte del testo in chiaro che conosce. Se la chiave segreta è stata generata da un generatore pseudo-casuale questa informazione può essere molto utile all'attaccante per ricostruire tutta la chiave segreta. E' pertanto ovvio come sia molto difficile, e quindi pericoloso per la sicurezza, utilizzare l'OTP con chiavi segrete che non siano veramente casuali.

L'utilizzo pratico dell'OTP è ovviamente non semplice poiché bisogna avere a disposizione chiavi veramente casuali e sufficientemente lunghe. In pratica quello che si può fare è scambiarsi in anticipo delle chiavi molto lunghe e casuali, e poi utilizzarle a seconda della necessità per cifrare ed inviare al proprio corrispondente i messaggi. Questa tecnica è stata ed è tuttora adottata dai servizi segreti per comunicazioni di altissima importanza, e solo per queste vista la difficoltà pratica di creare e scambiarsi le chiavi, e si dice che la famosa linea rossa tra Washington e Mosca fosse (sia ?) cifrata con OTP.

Il consiglio pratico che possiamo dare oggi è di utilizzare l'OTP se si è ragionevolmente sicuri della casualità e sicurezza della chiave, altrimenti conviene utilizzare degli algoritmi di tipo Stream che, come vedremo tra poco, sono molto simili all'OTP ma semplificano il problema della generazione delle chiavi segrete.

L'OTP ci mostra alcuni aspetti fondamentali della crittografia in un algoritmo semplicissimo ma al contempo sicurissimo se usato correttamente. In particolare la casualità della chiave ed il suo utilizzo limitato sono requisiti generali di quasi tutti gli algoritmi crittografici.

1.2.3 Algoritmi Moderni

I due esempi discussi nelle precedenti sezioni ci indicano quali sono le caratteristiche di un moderno algoritmo crittografico. Partiamo dal considerare le proprietà che dovrebbe possedere una chiave segreta per poter essere realisticamente utilizzata:

- La chiave deve essere il più breve possibile
- La chiave deve poter essere riutilizzata più volte
- La chiave non può essere perfettamente casuale (la richiesta della massima casualità possibile della chiave rimane in ogni caso anche solo per evitare che un attaccante possa indovinarla senza grandi difficoltà).

E' chiaro che un algoritmo quale l'OTP che affida in pratica tutta la sua sicurezza alle proprietà della chiave, ma che in se stesso è molto semplice, non potrà mai essere sicuro con chiavi di questo tipo. Al contrario, l'onere del garantire la sicurezza passa all'algoritmo che quindi deve essere complicato per il fatto che deve cercare di gene-

rare testi casuali utilizzando un algoritmo deterministico partendo dal testo in chiaro e da una chiave segreta. Claude Shannon, alla fine degli anni '40, studiò il problema da un punto di vista della Teoria della Informazione e mostrò che vi sono solo due tecniche generali per poter ottenere il risultato sperato:

- Confusion ovvero rendere confusa la relazione tra il testo in chiaro e quello cifrato, tipicamente tramite la sostituzione di un carattere con un altro secondo una tabella
- Diffusion ovvero distribuire l'informazione su tutto il testo cifrato, ad esempio permutando (trasponendo) i caratteri.

Vedremo come molti degli algoritmi crittografici usati oggi siano costruiti proprio combinando in maniera opportuna queste due tecniche.

Qualche anno dopo i risultati di Shannon, tra la fine degli anni '60 e la metà degli anni '70, i matematici capirono che vi è anche una alternativa matematica a questo approccio per cifrare dei dati. L'alternativa è basata sull'esistenza di una

- One-way-trapdoor function

Una one-way-trapdoor function **f** è una funzione matematica con le seguenti 3 proprietà

1. **f** è facile da calcolare
2. **f** è *praticamente impossibile* [11] da invertire, ovvero calcolarla nella direzione inversa
3. **f** è facile da invertire se si conosce una informazione ulteriore, una chiave segreta (trapdoor).

A prima vista sembra che funzioni matematiche che godono di tali proprietà debbano essere estremamente complicate e che questo approccio sia pertanto non implementabile. Vedremo invece che esistono funzioni molto semplici, basate però su problemi matematici molto difficili, che *dovrebbero* godere di queste proprietà. Dal punto di vista matematico il condizionale è qui d'obbligo in quanto nessuno è stato in grado al giorno d'oggi di *dimostrare* che queste funzioni sono in grado di soddisfare il punto 2. Ovviamente se si dimostrasse che il punto 2 non è valido, la sicurezza dell'utilizzo crittografico di queste funzioni verrebbe a mancare immediatamente.

Il lettore attento potrebbe rimproverarci una contraddizione, abbiamo scritto che Shannon dimostrò l'esistenza di solo due metodi per realizzare algoritmi crittografici e subito dopo abbiamo introdotto le one-way-trapdoor function. In realtà da un punto di vista matematico non vi è contraddizione. Il punto fondamentale è che ogni dimostrazione matematica si basa comunque su delle ipotesi. Per quanto generali fossero le ipotesi fatte da Shannon, queste non includevano il possibile utilizzo di one-way-trapdoor function. Torneremo nella terza parte di queste note, a considerare sviluppi ancora più avanzati che a prima vista potrebbero ancora sembrare in contraddizione con i risultati di Shannon ma che ovviamente non lo sono.

11 Con questo termine si intende qui e nel proseguio che anche nel caso esista un algoritmo matematico per risolvere il problema, la difficoltà computazionale è tale che anche con gli elaboratori più potenti che esistono non si è in grado di trovare la soluzione.

Se vi sono fondamentalmente due gruppi di tecniche per creare algoritmi crittografici, gli algoritmi stessi sono classificati in tre grandi famiglie. Come per le varie classificazioni che introdurremo successivamente, dobbiamo sottolineare che alle volte i confini fra le varie famiglie non sono ben definiti. In particolare, è spesso possibile utilizzare un algoritmo di una famiglia per generare un algoritmo di un altra famiglia. Ad esempio, è facile creare un algoritmo di Hash utilizzando un algoritmo simmetrico. Le tre grandi famiglie di algoritmi crittografici usati per garantire la sicurezza delle informazioni in formato digitale, sono:

- Algoritmi Simmetrici: le chiavi di cifratura e decifratura sono uguali o semplicemente deducibili una dall'altra
- Algoritmi Asimmetrici o a Chiave Pubblica: hanno due chiavi, una Privata ed una Pubblica, ed è *in pratica impossibile* ottenere la chiave Privata dalla chiave Pubblica in un tempo ragionevole
- Algoritmi di Hash o Digest: sono funzioni one-way (ovvero praticamente impossibili da invertire) tali che data una stringa di lunghezza arbitraria generano una stringa di lunghezza fissa con particolari proprietà crittografiche che descriveremo nella sezione 1.3.6.

Le principali differenze fra le tre famiglie sono:

- gli algoritmi Simmetrici e di Hash utilizzano le tecniche di Shannon, mentre gli algoritmi Asimmetrici utilizzano le one-way-trapdoor function
- gli algoritmi Simmetrici e Asimmetrici sono invertibili (usando la chiave segreta) mentre quelli di Hash non sono invertibili e normalmente non richiedono l'utilizzo di una chiave segreta
- gli algoritmi Simmetrici ed Asimmetrici sono usati principalmente per garantire la Confidenzialità, mentre gli algoritmi di Hash sono usati principalmente per garantire l'Integrità; l'Autenticità si può ottenere combinando l'utilizzo di chiavi segrete o Pubbliche/Private ed i relativi algoritmi con l'uso di Hash.

Nelle prossime sezioni illustreremo in un maggiore dettaglio i principali algoritmi e le loro caratteristiche.

1.2.4 Tipi di attacchi

Anche se in queste nostre brevi note non avremo l'opportunità di discuterne in dettaglio, è doveroso illustrare brevemente le tecniche di attacco agli algoritmi. Bisogna premettere che le tecniche più avanzate di attacco agli algoritmi sono spesso costruite ad hoc per il particolare algoritmo e che le tecniche matematiche adottate in questi casi sono alle volte ancora più sofisticate di quelle utilizzate per creare gli algoritmi.

Possiamo inizialmente classificare i tipi di attacco in Attivi o Passivi a seconda del fatto che l'attaccante sia in grado o meno di inviare alla cifratura testi in chiaro di propria scelta. Il più semplice tipo di attacco è quello comunemente detto di Forza

Bruta, ovvero provare tutte le chiavi sino a trovare quella che decifra correttamente il messaggio. Una ulteriore classificazione è la seguente:

- Ciphertext only: ottenere la chiave od il testo in chiaro dal solo testo cifrato (passivo)
- Known plaintext: ottenere la chiave noti alcuni testi in chiaro ed i corrispondenti testi cifrati (questo caso è molto frequente in pratica, basta considerare che in molte comunicazioni informatiche sono presenti headers dei messaggi ecc. che l'attaccante può facilmente ricostruire - passivo)
- Chosen plaintext: l'attaccante sceglie i testi in chiaro che fa cifrare e da questi risale alla chiave (attivo)
- Adaptive chosen plaintext: come il precedente ma la scelta del testo in chiaro dipende dai testi cifrati ottenuti precedentemente (attivo).

Un buon algoritmo crittografico deve far fallire tutti questi tipi di attacchi.

Oltre a queste classi storiche di attacchi, vi sono delle classi più recenti che possono anche essere viste come sotto-classi speciali dei casi precedenti. Ad esempio vi sono:

- Differential Cryptoanalysis: sottoclasse di attacchi chosen-plaintext, in questo caso si preparano dei testi non cifrati con delle particolari differenze che dipendono dall'algoritmo dato, e si analizza la differenza tra testi cifrati così ottenuti[12]
- Related-Key Cryptoanalysis: l'attaccante è a conoscenza della differenza fra alcune chiavi usate, ma non le chiavi stesse e studiando le differenze fra i testi cifrati con le diverse chiavi è in grado di decifrare i testi stessi
- Linear Cryptoanalysis: l'idea nasce dal One-Time-Pad ove se si fa l'XOR del testo in chiaro con il testo cifrato si ottiene la chiave, in questo caso si cerca una approssimazione matematica *lineare* all'algoritmo in modo di poter fare qualche cosa di simile, giungendo a formulare il problema come un (complicato) sistema di equazioni lineari o comunque di ordine basso.

In ogni caso possiamo enunciare come segue la richiesta di sicurezza (cioè non rottura) per un algoritmo:

> *un algoritmo è sicuro, o non-rotto, se l'attacco più efficace contro di esso è equivalente ad un attacco di forza bruta, ovvero a provare tutte le possibili chiavi.*

Ovviamente per poter dare un qualche livello di sicurezza, l'algoritmo deve avere un numero di chiavi tali che sia praticamente impossibile per un attaccante provarle tutte in un tempo ragionevole, ed al contempo chi utilizza l'algoritmo deve scegliere una chiave che l'attaccante non possa indovinare facilmente, per cui la cosa migliore è che la chiave sia puramente casuale.

Per quanto riguarda gli Hash, vedremo nella sezione a loro riservata i particolari tipi di attacchi che sono specifici per questa famiglia di algoritmi.

12 Come vedremo questo tipo di attacco ha svolto un ruolo significativo nella storia di DES.

1.3 I Principali Algoritmi

In questa sezione considereremo molto brevemente i principali algoritmi crittografici. Questa rassegna sarà molto rapida e ci limiteremo solo ai principali, a nostro parere, algoritmi attuali. Sappia il lettore che ci saranno delle notevoli omissioni, a cui cercheremo di rimediare nella seconda parte di queste note.

1.3.1 Algoritmi Simmetrici

Come abbiamo già indicato, gli algoritmi simmetrici hanno una storia secolare e quindi ci dilungheremo un poco di più sulle loro proprietà generali. Per prima cosa, a partire dal risultato di Shannon citato precedentemente, possiamo classificare i metodi adottati per creare gli algoritmi in base alla loro struttura. Pertanto alcune delle classi di metodi storici sono:

Trasposizione/Permutazione:
- a periodo fisso
- a periodo variabile a seconda della chiave

Sostituzione:
- monoalfabetica semplice come ad esempio il Cifrario di Cesare (c=t+b modN), o affine (c=at+b modN)
- poligrammatica 2 o più caratteri sono sostituti in gruppi corrispondenti (cifrario di Playfair)
- omofonica 1 carattere viene sostituito con 1 tra k caratteri scelto a caso, k dipende dalla frequenza del carattere in modo che la frequenza di ogni carattere nel testo cifrato sia la stessa (allargamento dell'alfabeto)
- polialfabetica ad esempio monoalfabetica ma con k alfabeti che si ripetono con periodo t (cifrario di Vigenère, macchine a rotori, Enigma)

Come vedremo, negli algoritmi moderni sia le permutazioni che le sostituzioni hanno delle caratteristiche ancora più generali.

Inoltre gli algoritmi simmetrici si possono dividere in due grandi famiglie, quella degli algoritmi a Blocchi e degli algoritmi Stream o a Caratteri.

Gli Algoritmi a Blocchi suddividono il testo da cifrare in blocchi di lunghezza fissa (spesso pari alla lunghezza della chiave) e producono blocchi cifrati di norma della stessa lunghezza. La cifratura di ogni blocco può essere fatta in due modalità principali:

- ECB (Electronic Code Book): ogni blocco è cifrato in modo indipendente
- CBC (Cipher Block Chaining): ogni blocco ha un riporto dal blocco precedente, il primo blocco ha un IV (Initial Value) noto (self-synchronizing).

Si noti che ECB è resistente agli errori poiché se si perde o modifica un blocco nella trasmissione i blocchi seguenti non ne risentono, ma se si ripete un blocco nel testo in chiaro questo verrà cifrato nello stesso modo. Al contrario CBC cifra diversamente uno stesso blocco che si ripete, ma è suscettibile agli errori di trasmissione nel qual caso più di un blocco verrà perso, sino a quando l'algoritmo non si risincronizza.

Gli algoritmi di tipo Stream cifrano un bit (o carattere) alla volta usando una trasformazione che cambia per ogni bit (o carattere). Tipicamente l'operazione è del tipo

bit-cifrato = bit-in-chiaro XOR bit-chiave

ove la chiave (anche essa 1 bit od 1 carattere) è generata dinamicamente dall'algoritmo, con una formula analoga se l'algoritmo agisce su di un carattere. In questo caso l'algoritmo è detto generare una *key-stream*.

Vediamo quindi come gli algoritmi di tipo Stream sono simili all'OTP. Possiamo infatti pensare agli algoritmi di tipo Stream come a degli OTP con chiavi non esattamente casuali ove l'algoritmo specifica come la chiave dell'OTP viene generata a partire dalla (breve) chiave segreta condivisa dai corrispondenti. Ovviamente la dimostrazione di assoluta sicurezza dell'OTP non è più valida, ma gli algoritmi di tipo Stream riducono al minimo possibile i rischi connessi all'uso di una chiave non esattamente casuale con l'OTP.

In questo caso il bit della chiave può essere generato in due modalità principali

- Sincrono: il bit della chiave è generato indipendentemente dal testo cifrato e dal testo in chiaro (l'OTP può essere visto come il più semplice esempio ove l'algoritmo è un generatore puro di bit casuali)
- Asincrono: il bit della chiave dipende da un numero fissato di bit cifrati precedenti (self-synchronizing).

E' chiaro che un algoritmo di tipo stream pone tutta la propria sicurezza nella generazione della key-stream. Più la key-stream che viene generata è simile ad una stringa puramente casuale di bit come nell'OTP, più l'algoritmo è sicuro. Ovviamente la generazione della key-stream non è (e non può essere) casuale ma al contrario deve essere unicamente determinata a partire dalla chiave segreta a lei associata, altrimenti non sarebbe possibile decifrare il testo cifrato.

Non è sempre ovvia quale sia la differenza fra gli algoritmi a blocchi e quelli di tipo stream. Ad esempio un carattere di un algoritmo Stream può essere interpretato come un blocco (piccolo) di un algoritmo a Blocchi. In pratica si denotano:

- come Stream algoritmi veloci e che agiscono su di un singolo bit (od un carattere) di testo, adatti quindi a cifrare comunicazioni in tempo reale facendo un XOR del bit con la stream-key
- come Blocchi gli algoritmi ove vengono fatte delle operazioni contemporaneamente su tutti gli elementi di un blocco di n (ad esempio n=64) bit, come ad esempio delle permutazioni.

Alcuni tra gli algoritmi simmetrici più noti sono:

- DES (Data Encryption System): blocco di 64bit, chiave di 56bit (non sicuro) pensato per essere particolarmente efficiente in hardware
- 3DES: DES $(DES)^{-1}$DES – blocco 64bit, chiave 168 o 112 bit – sicuro (non tutte le chiavi sono buone)
- AES (Advanced Encryption Standard, originariamente chiamato Rijndael): blocco 128 bit, chiave di 128, 192 o 256 bit
- IDEA (International Data Encryption Algorithm): blocco 64bit, chiave 128bit – molto sicuro (ma Proprietario)
- RC4: Stream, 10 volte più veloce di DES, chiave a partire da 40bit ma si suggerisce di utilizzare una chiave molto più lunga (anch'esso Proprietario)

Oltre a questi possiamo citare anche: GOST, Blowfish, RC5, Twofish, SERPENT, RC6, MARS, DEAL, SAFER+, FROG, LOKI-97, CAST-256, Magenta, DFC, CRYPTON, E2, HPC. Nella sezione successiva illustreremo alcune delle caratteristiche comuni di questi algoritmi.

1.3.2 DES

La storia di DES ha un interesse in se, la riportiamo succintamente nei seguenti punti:

- nel 1973 l'NBS (National Bureau of Standards) americana (ora NIST, National Institute of Standards and Technology) bandisce un concorso per un algoritmo crittografico per scopi commerciali; vi era l'esigenza di garantire la sicurezza delle comunicazioni per le grandi imprese americane tra di loro e nei loro rapporti, ad esempio come Contractor, con lo stato e le varie agenzie statali americane
- nel 1974 IBM fornisce un algoritmo basato su Lucifer, un algoritmo con blocco e chiave di 128 bit sviluppato da un gruppo di ricercatori dell'IBM tra cui vi era Don Coppersmith
- L'NBS sottopone per una valutazione l'algoritmo alla NSA (National Security Agency) la quale modifica l'algoritmo, tra le altre cose limitando la lunghezza della chiave a solo 56bit
- L'intervento della NSA provocò polemiche per molti anni: l'NSA aveva inserito delle backdoor o modificato l'algoritmo in modo da poter rompere, con i propri mezzi e conoscenze avanzate, i messaggi cifrati con DES? Nel 1990 si capì che questo non era vero, l'NSA aveva modificato l'algoritmo per proteggerlo contro la Differential Cryptoanalysis
- nel 1975 l'NBS pubblica l'algoritmo in modo che sia possibile per tutti implementarlo in software o hardware, sembra però che questo sia stato un malinteso con l'NSA che pensava che l'algoritmo dovesse rimanere segreto e fosse implementabile solo in hardware; chiarito l'equivoco anche se troppo tardi, NBS ed IBM decisero di non rendere noti i principi di progettazione e le caratteristiche teoriche dell'algoritmo, che furono mantenuti segreti sino al 1992.

Possiamo considerare DES come il padre di tutti gli algoritmi simmetrici moderni e di parte di quelli di Hash. La sua struttura nasce dall'analisi di Shannon e dalle esperienze dei cifrari storici evoluti sino alle macchine a rotori. La sua principale proprietà è di essere un algoritmo *semplice* sia dal punto di vista teorico che implementativo. La semplicità è una proprietà molto importante perché consente di provare matematicamente la sicurezza dell'algoritmo, o meglio poter dimostrare che tutti gli attacchi noti sono almeno veloci quanto l'attacco di forza bruta. La semplicità implementativa ovviamente rende più facile evitare errori nella implementazione in software e hardware e potenzialmente favorisce anche la velocità di esecuzione.

Come abbiamo detto, le operazioni possibili secondo Shannon per cifrare un testo sono due: Sostituzioni e Permutazioni che implementano rispettivamente la Confusione e la Diffusione dell'informazione. Da sola ognuna di queste tecniche non è sufficiente, ed anche se usate congiuntamente una volta sola il risultato è di troppa poca confusione e diffusione. Pertanto l'idea di base è quella di ripetere molte volte una serie di operazioni di confusione e diffusione (sostituzioni e permutazioni). Pertanto abbiamo che la struttura di un algoritmo moderno è in generale quella di

- un cifrario prodotto o rete di Sostituzioni-Permutazioni (SP-network)
- queste operazioni sono ripetute in N round dando luogo ad un *Iterated Block Cipher*

Alle sostituzioni e permutazioni dobbiamo ovviamente aggiungere il ruolo della chiave segreta. Invece che modificare le sostituzioni o le permutazioni a seconda della chiave segreta, il procedimento adottato da molti algoritmi è

- dalla chiave segreta originale generare, con un appropriato algoritmo, tante sotto-chiavi quanti sono i round dell'algoritmo
- all'interno di un round fare l'XOR del blocco (o di una sua parte) con la sotto-chiave relativa a quel round.

Quindi un algoritmo simmetrico a blocchi di norma è composto da due sotto-algoritmi

1. il sotto-algoritmo che specifica gli N round di permutazioni, sostituzioni e XOR con la sotto-chiave a cui viene sottoposto ogni blocco
2. il sotto-algoritmo che data la chiave segreta genera per ogni round la relativa sotto-chiave.

Senza entrare troppo nei dettagli anche perché DES è sì semplice ma fino ad un certo punto, vediamo come questi concetti sono applicati a DES.

DES ha 16 round ed una chiave segreta iniziale di 56 bit (64 bit di cui 8 sono di parità). Dalla chiave segreta un algoritmo formato da permutazioni e sostituzioni, genera 16 sotto-chiavi di 48 bit ciascuna.

L'algoritmo principale è quello che definisce i round. Prima di tutto, il testo da cifrare è diviso in blocchi di 64 bit ciascuno, ad ogni blocco è poi applicato l'algoritmo in modalità ECB o CBC. Le operazioni effettuate all'interno di un round su ogni blocco sono:

1. un blocco di 64 bit è diviso in due sotto-blocchi di 32 bit, ogni round agisce solo su di un sotto-blocco a 32 bit

2. con dei raddoppi e permutazioni di bit, il sotto-blocco a 32 bit viene esteso ad un sotto-blocco a 48 bit
3. viene fatto l'XOR del sotto-blocco a 48 bit con la sotto-chiave a 48 bit relativa al round in esecuzione
4. il sotto-blocco a 48 bit viene diviso in 8 sotto-sotto-blocchi di 6 bit; i 6 bit di ogni sotto-sotto-blocco vengono utilizzati come indirizzi che specificano un elemento di una tra 8 tabelle, dette **S-Box**; ogni elemento delle tabelle è costituito da 4 bit che forniscono l'output della trasformazione; questa è l'operazione di sostituzione, vengono sostituiti i 6 bit in ingresso con i 4 bit in uscita; alla fine si riforma un sotto-blocco di 32 (8x4) bit
5. viene fatta una permutazione sul sotto-blocco a 32 bit
6. si scambiano il sotto-blocco a 32 bit su cui si è appena agito con quello non modificato.

La divisione del blocco di 64 bit in due sotto-blocchi da 32 è una caratteristica particolare di DES e dei cifrari detti di *Feistel* e non è una proprietà generale. Quello che è comune a quasi tutti gli algoritmi è l'idea (non l'ordine) delle operazioni 3, 4 e 5, ovvero l'XOR con la sotto-chiave, la sostituzione tramite S-Box, e la permutazione.

Il punto cruciale per la sicurezza dell'algoritmo, e di tutti gli algoritmi con questa struttura, è la sostituzione tramite le S-box. Infatti questo è l'unico elemento non lineare dell'algoritmo, e quindi l'unico elemento che introduce delle proprietà simili alla casualità. Le S-box sono fisse, in pratica sono niente altro che delle matrici di numeri. I bit in input fungono da indirizzo dell'elemento della matrice (o tabella) ed il contenuto dell'elemento è l'output: molto semplice! Nel caso di DES ci sono 8 S-Box ognuna con 64 (2^6) numeri di 4 bit. La difficoltà sta nella scelta dei numeri contenuti nelle S-Box, e questi per DES sono stati forniti dalla NSA che li ha ottenuti richiedendo che DES fosse resistente contro gli attacchi di Differential Cryptoanalysis.[13]

Lo studio della NSA sulla resistenza di DES alla Differential Cryptoanalysis, oltre ai numeri nelle S-Box, ha anche indicato che il numero minimo di round per ottenere una sufficiente diffusione della confusione è 16.

L'inverso di DES (per decifrare) è molto semplice, basta utilizzare le 16 sotto-chiavi in ordine inverso. Questa proprietà di DES è veramente notevole.

DES però ha 64 chiavi deboli da non usare, ad esempio quelle in cui si ripetono blocchi di quattro 0 e quattro 1 come 00001111....

In generale DES è comunque un ottimo algoritmo, oggi non è più sicuro solo perché è possibile fare degli attacchi di forza bruta a qualunque algoritmo con chiavi di 56-bit, e possiamo dire che chiavi di lunghezza inferiore ai 100bit sono tutte in pericolo. Purtroppo DES non può utilizzare chiavi più lunghe direttamente, quello che è possibile fare è applicare DES più volte con chiavi diverse. Il modo più comune di utilizzare più volte DES consecutivamente è detto **3-DES** ed è definito dall'applicazione di **DES (DES)$^{-1}$DES**. E' possibile usare 2 o 3 chiavi a 56 bit diverse, nel primo caso la prima e l'ultima applicazione di DES utilizzano la stessa chiave. Il caso comune è quello di usare 3 chiavi a 56 bit diverse, ovvero una chiave a 168 bit. Esiste però un attacco contro tutti gli algoritmi così composti che è meglio della forza bruta su tutte

13 Il procedimento dettagliato utilizzato dalla NSA per ottenere i numeri contenuti nelle S-Box di DES è ancora tenuto segreto.

le chiavi, e permette di ridurre i tentativi a 2/3, quindi effettivamente per quanto riguarda la resistenza agli attacchi è come se 3-DES avesse una chiave a 112 bit.

1.3.3 AES

Come per DES, riportiamo succintamente la storia molto più recente di AES.

- Dal 1987 NIST ha cercato un sostituto a DES preoccupata dalla brevità della chiave
- Nel 1997 NIST annunciò una competizione pubblica internazionale per un algoritmo simmetrico a blocchi di 128 bit e chiavi di 128, 192, e 256 bit; doveva essere implementabile sia in HW che SW ed anche in sistemi embedded (veloce, e con poche richieste di RAM e ROM); l'algoritmo vincitore sarebbe divenuto di pubblico dominio
- ci furono 15 concorrenti principalmente da università ed enti di ricerca, 5 finalisti, e nell'ottobre 2000 un vincitore: **Rijndael** di J.Daemen e V. Rijmen. Gli altri concorrenti furono Twofish, SERPENT, RC6, MARS, DEAL, SAFER+, FROG, LOKI-97, CAST-256, Magenta, DFC, CRYPTON, E2, HPC.
- AES è diventato uno Standard Pubblico il 26/11/2001: FIPS-PUB-197
- nel giugno 2003 la NSA ha approvato AES a 128 bit per i documenti classificati dalle amministrazioni USA come SECRET, e AES a 192 o 256 bit per i documenti classificati TOP-SECRET.

Questo ultimo punto porta molti a concludere che la NSA considera AES un buon algoritmo anche se ci sono sempre gli scettici che ritengono che la NSA sia già in grado di romperlo. Ovviamente i servizi segreti americani non faranno commenti su questo punto a breve.

AES è un algoritmo molto più efficiente di DES, questo per disegno e volontà dei suoi creatori. E' anche molto più *semplice* nella struttura e matematicamente più facile da analizzare. La struttura generale è la stessa di quella di DES, anche se i blocchi sono di 128 bit e sono divisi in 4x4 sotto-blocchi di 8 bit. Ogni round agisce su tutti i sotto-blocchi, e non solo su metà come nel caso di DES. AES ha 10 round quando usa una chiave a 128 bit, 12 round con chiave a 192 bit e 14 round con chiave a 256 bit. Di nuovo la sua sicurezza si basa principalmente sulle 16 S-Box che questa volta hanno 256 elementi ognuno di 8 bit (8 bit di indirizzo/input e 8 bit di elemento/output). A differenza di DES, l'algoritmo per la creazione delle sotto-chiavi è ragionevolmente complicato, e l'inverso di AES è praticamente un altro algoritmo.

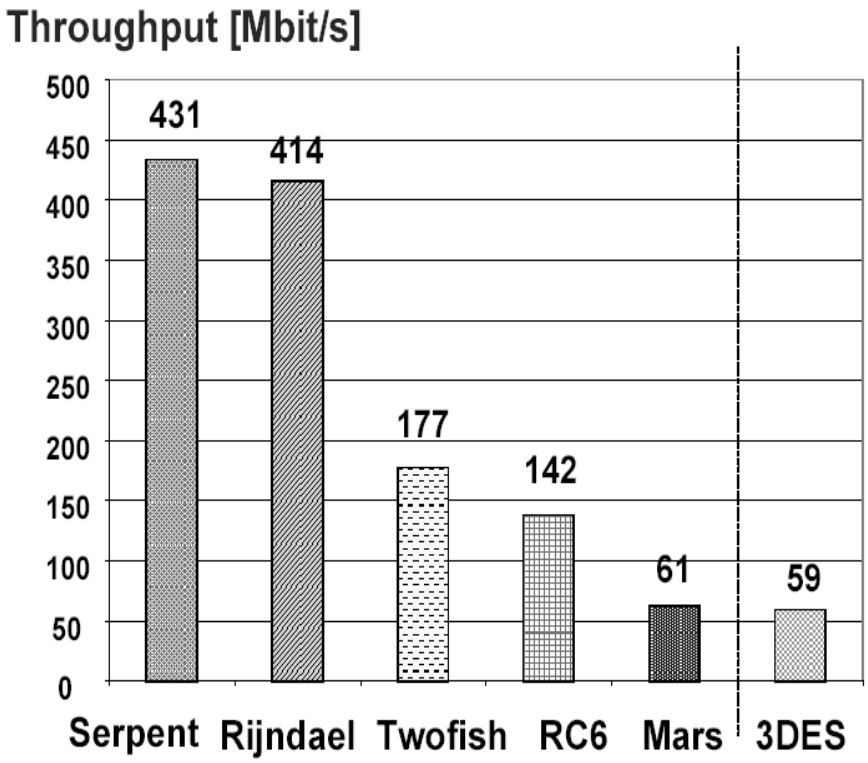
Ma la differenza principale da DES e AES dal punto di vista dell'algoritmo, è che mentre DES si basa su semplicissime operazioni matematiche sui bit (spostamenti, XOR ecc.), AES è basato su matematica molto avanzata: tutte le operazioni sono compiute all'interno del campo finito di Galois GF(2^8). AES basa la propria sicurezza su concetti avanzati di matematica ma al contempo è molto semplice da implementare. Infatti, implementare AES in hardware e software è di solito più facile ed efficiente di implementare DES, richiede meno risorse ed è più performante. Ma capire

perché AES è sicuro e funziona, richiede una buona dose di matematica.

In pratica AES è dalle 2 alle 10 volte più veloce di 3-DES. E' difficile paragonare le prestazioni degli algoritmi perché ad esempio nelle implementazioni software, queste dipendono anche dal linguaggio usato, dai compilatori, dai sistemi operativi ecc. Indicativamente per le implementazioni software nei test sono stati ottenuti risultati di questo tipo:

Algoritmo	Chiave	test1	test2 (in MB/s)
AES	128	27.6	62.046
AES	192	24.4	56.067
AES	256	22.0	49.155
3DES	168(112)	10.4	18.790

ove la differenza tra test1 e test2 è data dall'utilizzo di hardware e sistemi operativi diversi. Per le implementazione in hardware, in questo caso FPGA, si sono ottenute tavole di paragone di questo tipo (da Pawel Chodiviec):



1.3.4 Algoritmi Asimmetrici

Come abbiamo già indicato, la seconda e più recente famiglia di algoritmi si basa sull'esistenza di problemi matematici di difficile soluzione. L'argomento non è per

nulla semplice e cercheremo di dare delle indicazioni per cercare di comprendere la validità ed i problemi associati a questo approccio.

Abbiamo già dato la definizione di *One-Way-Trapdoor function*, ed abbiamo visto che per costruirla abbiamo bisogno di un *problema matematico difficile*.

Gli informatici considerano *difficili,* tra altri, i problemi per i quali al crescere della lunghezza dei numeri in input, il numero di operazioni elementari richieste da un algoritmo per risolverli cresce velocissimamente.[14] Esistono problemi, di cui vedremo esplicitamente un semplice esempio, per i quali già con numeri in input relativamente brevi, di qualche centinaio di cifre decimali, il numero di operazioni elementari necessarie per risolverlo è superiore alla potenza di calcolo disponibile oggi e nel prossimo futuro. E' chiaro quindi che da un punto di vista pratico questi problemi oggi sono insolubili quando i numeri in input sono sufficientemente lunghi.

E' importante però sottolineare che per molti di questi problemi ritenuti *difficili* nessuno è stato sinora in grado di *dimostrare* la loro difficoltà reale. In altre parole, nessuno sa se esistono algoritmi in grado di risolvere questi problemi molto più velocemente. E' chiaro che se si scoprissero algoritmi *veloci* per risolvere questi problemi, essi non sarebbero più *difficili*. Molti matematici sono fiduciosi che questi algoritmi veloci, almeno per gli elaboratori classici, non esistono, ma la garanzia assoluta non esiste. Vedremo poi nella terza parte di queste note, come gli elaboratori quantistici potrebbero eseguire alcuni di questi algoritmi velocemente e risolvere quindi alcuni di questi problemi.

Il più semplice problema di questo tipo è quello della **Fattorizzazione in numeri primi.** Possiamo formulare il problema come segue:

1. *Definizione*: Un numero primo è un numero positivo divisibile solo per 1 e per se stesso (senza resto)
2. siano dati due numeri primi grandi p e q (ad esempio con più di 100 cifre decimali)
3. si prenda il prodotto n = p q
4. noto solamente n trovare p e q è un problema *difficile*; se ad esempio n ha 300 o più cifre decimali, nessun elaboratore o rete di elaboratori esistente utilizzando il più veloce algoritmo noto, è in grado di trovarne i fattori primi in un breve tempo, ma impiegherebbe centinaia o migliaia di anni
5. noti n e p, trovare q è banale, p è una *trapdoor* o chiave segreta.

Quindi il prodotto di due numeri primi relativamente grandi è in pratica una One-Way-Trapdoor function. La difficoltà del problema sta nel fatto che se è noto solo n (ed il fatto che n è il prodotto di due numeri primi) trovare i due numeri primi con il più veloce algoritmo noto richiede tantissimo tempo (tantissime operazioni elementari da fare) anche utilizzando gli elaboratori più veloci esistenti.

14 In questa sede non possiamo introdurre la teoria matematica che studia queste problematiche. Per il lettore più interessato possiamo indicare che i matematici hanno classificato i problemi matematici in varie classi a seconda della loro *difficoltà;* le classi più note sono le classi **P** e **NP**.

Per dare un'idea della dimensione del problema da risolvere per fattorizzare n, supponiamo di adottare la tecnica più semplice ed ovviamente meno efficace, ovvero provare a dividere n per tutti i numeri primi a partire da 3: se n ha circa 300 cifre decimali in media dovremo provare 10^{147} numeri! Non solo, come prima cosa dobbiamo trovare i numeri primi, od almeno mostrare che ogni numero che proviamo è primo, operazione che ovviamente richiede anch'essa tempo.

Un problema *difficile* con una trapdoor si presta immediatamente ad applicazioni crittografiche: infatti chi non conosce la trapdoor (o chiave segreta) non è in grado di risolvere il problema, almeno in pratica.

Un altro problema *difficile* simile a quello della fattorizzazione in numeri primi, su cui si basano molti algoritmi crittografici è quello dei Logaritmi Discreti, che però non descriveremo in questa sede.

Gli algoritmi Asimmetrici hanno la caratteristica fondamentale di avere due diverse chiavi, una *Privata* e l'altra *Pubblica*. Nell'esempio fatto della fattorizzazione in numeri primi, la chiave Privata corrisponderebbe a p e la chiave Pubblica a n. La chiave Privata deve essere mantenuta segreta dal proprietario, e non deve essere condivisa con nessun altro. Non vi è pertanto il problema comune agli algoritmi Simmetrici di distribuire le chiavi segrete ai corrispondenti.

La chiave Pubblica invece, è nota a tutti i propri corrispondenti. La sicurezza dell'algoritmo non richiede alcuna protezione della chiave Pubblica in quanto è praticamente impossibile ricavare la corrispondente chiave Privata quando si è a conoscenza solo della chiave Pubblica.

Alla chiave Privata e Pubblica non è assegnato un ruolo univoco di chiave di cifratura o decifrazione. A seconda di come viene utilizzata, ogni chiave può essere usata per cifrare o per decifrare. In pratica, se un corrispondente cifra utilizzando la chiave Pubblica dell'altro, il ricevente decifra utilizzando la propria chiave Privata. In questo caso è garantita la confidenzialità del messaggio, visto che solo una persona è in possesso della chiave Privata e solo lui può decifrare il messaggio. E' possibile anche il contrario, se prima uno cifra con la chiave Privata, poi qualunque corrispondente può decifrare con quella Pubblica. In questo caso non si garantisce la confidenzialità del messaggio poiché chiunque può decifrarlo, ma si garantisce l'autenticità dello stesso perché solo una persona, il possessore della chiave Privata, può averlo cifrato. Torneremo nella seconda parte di queste note a descrivere più in dettaglio gli usi di questi algoritmi.

1.3.5 RSA

Descriviamo brevemente l'algoritmo di Rivest, Shamir e Adleman (1977):

1. si scelgano due numeri primi molto grandi **p** e **q** (ad esempio tali che il loro prodotto abbia non meno di 310 cifre decimali, ovvero 1024 bit)
2. si scelga un numero (casuale) **e** con le proprietà: $1 < \mathbf{e} < \mathbf{r=(p-1)(q-1)}$ e relativamente primo a **r**, ovvero **e** e **r** hanno solo 1 come divisore comune

3. si calcoli **n =pq**
4. si calcoli **d** come l'unico intero 1 < **d** < **r** tale che **ed≡1 mod(r)** (ovvero esiste un k intero tale che ed=1+kr, l'operazione mod(r) può essere vista anche come il resto della divisione per r, oppure sottraendo/sommando r, ridurre il numero dato all'intervallo [0,r-1])
5. la Chiave Pubblica D è la coppia **(e,n)**
6. la Chiave Privata C è la coppia **(d,n)**
7. data la Chiave Pubblica ed il messaggio **m** (il messaggio viene usualmente diviso in blocchi della stessa lunghezza in bit di **n,** nelle formule seguenti **m** rappresenta quindi un blocco del messaggio) è possibile de/cifrare con la formula

   $$c = m^e \bmod(n)$$

8. data la Chiave Privata ed il messaggio **c** (o un blocco del messaggio di lunghezza in bit uguale alla lunghezza di **n**) è possibile de/cifrare con la formula

   $$m = c^d \bmod(n)$$

Alcune osservazioni sull'algoritmo RSA sono d'obbligo:

- come già indicato, un messaggio *cifrato* con una delle due chiavi viene *decifrato* con l'altra chiave, l'uso delle chiavi è simmetrico in questo senso, ma una chiave è Privata, l'altra è Pubblica
- di solito nelle applicazioni viene scelto **e**=3 oppure **e**=2^{16}+1=65537 (il numero 3, ed in generale **e** piccolo, non dovrebbe essere usato se si cifra lo stesso blocco ripetutamente con la stessa chiave; 65537 contiene solo due 1 nella rappresentazione binaria e rende veloci alcune operazioni)
- si noti che per calcolare **d** è necessario conoscere **e**, **p-1**, **q-1** quindi chi conosce la chiave Pubblica deve, in linea di principio, fattorizzare **n** per calcolare **p**, **q** e con questi calcolare poi **d**; non è però stato dimostrato matematicamente che questo sia l'unico modo di calcolare **d** partendo dalla chiave Pubblica, perciò non si sa se il problema RSA sia veramente equivalente alla fattorizzazione in numeri primi oppure se sia un problema molto più semplice
- in pratica, al giorno d'oggi l'unico modo noto pubblicamente di ottenere la chiave Privata da quella Pubblica è quello di fattorizzare **n**, se questo è un numero di almeno 1024 bit (più di 300 cifre decimali) non è noto pubblicamente alcun elaboratore che sia in grado di farlo in un tempo *ragionevole*
- RSA è circa 1000 volte più lento di DES in hardware e circa 100 volte in software
- chiavi Pubbliche sicure oggi e comunemente utilizzate sono di 1024, 2048, 4096 o più bit!

Gli altri principali algoritmi Asimmetrici sono ElGamal e DSA (solo per le Firme Digitali) i quali usano il problema dei Logaritmi Discreti.

Il principale problema degli algoritmi asimmetrici è la loro lentezza ed il loro impiego di risorse. Per questo essi vengono per lo più utilizzati nella prima parte dei protocolli per le fasi di autenticazione e per lo scambio di chiavi segrete (dette *chiavi di sessione*) da utilizzare poi con gli algoritmi simmetrici.

Vogliamo sottolineare come nei protocolli asimmetrici le chiavi Pubbliche siano appunto a disposizione di chiunque senza che vi siano problemi di sicurezza (nelle condizioni indicate precedentemente). Questo risolve ovviamente uno dei principali problemi degli algoritmi simmetrici, ove tutta la sicurezza dell'algoritmo era nel mantenere segreta la chiave. Ovviamente la chiave Privata deve essere mantenuta segreta ma solo dal suo legittimo proprietario, non vi è mai la necessità che questa venga distribuita, anzi questo è del tutto vietato.

Ricordiamo inoltre come la sicurezza di questi algoritmi crittografici, ovvero il fatto che sia in pratica impossibile ottenere la chiave Privata conoscendo quella Pubblica, è basata su problemi matematici *difficili* per i quali non è stato però sinora dimostrato esattamente che algoritmi veloci di soluzione non esistono. Vi è quindi sempre il rischio, anche se oggi molto limitato, che in un futuro più o meno lontano la sicurezza di questi algoritmi crittografici venga meno.

Come esempio di una applicazione degli algoritmi asimmetrici, consideriamo due corrispondenti i quali devono scambiarsi informazioni. Essi possono utilizzare la seguente procedura. Entrambi adottano un protocollo a chiave Asimmetrica e generano le proprie chiavi Pubbliche e Private. I due corrispondenti si scambiano le chiavi Pubbliche. Poi ogni qual volta uno dei corrispondenti debba inviare all'altro un documento, egli procede come segue:

1. genera una chiave segreta casuale di sessione
2. cifra il documento con un algoritmo simmetrico e la chiave segreta casuale di sessione appena generata
3. cifra la chiave segreta casuale di sessione con l'algoritmo Asimmetrico e la chiave Pubblica del corrispondente
4. invia al corrispondente il documento cifrato e la chiave di sessione cifrata
5. il corrispondente decifra con la propria chiave Privata la chiave segreta di sessione e con quest'ultima decifra il documento.

Utilizzando gli algoritmi Asimmetrici i due corrispondenti si possono quindi scambiare in tutta sicurezza le chiavi segrete da usare con gli algoritmi simmetrici.[15]

1.3.6 Algoritmi di Hash (Impronte)

Scopo degli algoritmi di Hash (Impronta) è quello di generare una breve stringa, appunto una impronta, che identifichi un documento di lunghezza arbitraria. L'uso principale di un Hash è per verificare che il documento non è stato modificato. In realtà gli Hash non sono solo crittografici, sono comunemente usati ad esempio in molti protocolli di trasmissione dati per verificare se vi sono stati errori casuali di trasmissione. Questi algoritmi non sono resistenti contro attacchi malevoli, ma sono so-

15 Ovviamente tutto il procedimento può essere automatizzato, ad esempio questo è il principio di funzionamento dei protocolli che realizzano tunnel cifrati (comunemente chiamati VPN cifrate).

lamente in grado di verificare errori casuali in condizioni *normali*. Dobbiamo perciò distinguere tra Hash normali e Hash crittografici che invece sono resistenti (ma non del tutto immuni) ad attacchi malevoli.

Dobbiamo prima di tutto sgombrare il campo da una idea erronea, l'algoritmo di Hash perfetto, che cioè associa ad un testo una stringa di lunghezza fissa che lo identifica univocamente, non esiste per definizione! Per convincersi di questo fatto basta considerare che il numero di documenti possibili anche di lunghezza finita, ad esempio 100 pagine, è molto più grande del numero di valori possibili di un Hash ad esempio a 160 bit. Ne consegue che ad ogni valore di un Hash devono corrispondere idealmente più documenti.

Visto che l'algoritmo di Hash perfetto non esiste, dobbiamo limitarci a considerare algoritmi di Hash che producono una stringa *praticamente unica* per un documento di una certa lunghezza. Questo è invece fattibile.

Per capire come funziona un algoritmo di hash, cominciamo a descrivere una semplicissima checksum non crittografica. L'algoritmo è

1. Data la stringa originale lunga b bit, si estenda con un padding ad un multiplo di 128 bit (nel padding si mette di solito una rappresentazione di b)
2. Si divida la stringa in m sotto-blocchi di 128 bit (m x 128 = b + pad)
3. Si faccia lo XOR (bit per bit) di ognuno degli m sotto-blocchi con il successivo, ripetutamente a partire dal primo (l'XOR del primo sotto-blocco con il secondo sotto-blocco da il nuovo secondo sotto-blocco; l'XOR del nuovo secondo sotto-blocco con il terzo sotto-blocco da il nuovo terzo sotto-blocco, e così via) il risultato finale è una stringa di 128 bit

E' chiaro che questo algoritmo non ha alcuna proprietà crittografica, però la struttura degli algoritmi di Hash è questa. La differenza principale è che invece dell'XOR nel terzo punto, un algoritmo crittografico ha una funzione **f** specifica. In un Hash crittografico il terzo punto diventa

> Per ogni sotto-blocco $\mathbf{x_i}$ si calcoli $\mathbf{H_i = f(x_i,H_{i-1})}$ ove $\mathbf{H_i}$ è il valore dello Hash sino al sotto-blocco **i**, ed è di lunghezza **h**

In molto algoritmi **h** è di 128 o 160 bit ed i sotto-blocchi $\mathbf{x_i}$ sono di 512 bit. Ovviamente la funzione **f** è il cuore dell'algoritmo di Hash. Questa funzione deve garantire il più possibile le seguenti proprietà:

- Dato un Hash non è possibile ricostruire la stringa o documento originario (resistenza alle *pre-immagini*)
- E' statisticamente impossibile che due documenti generino lo stesso Hash (resistenza alle *collisioni*)
- Una piccola modifica di un documento genera una grande modifica dell'Hash (resistenza alle *correlazioni*).

E' possibile costruire la funzione **f** utilizzando un algoritmo simmetrico, in pratica basta specificare come la chiave ed il blocco in input all'algoritmo simmetrico dipendono da (x_i,H_{i-1}).

E' però meglio, sia dal punto di vista della sicurezza che delle prestazioni, che la funzione **f** sia progettata specificamente per un algoritmo di Hash. La tecnica di costru-

zione della funzione **f** è simile a quella di un algoritmo simmetrico, ma cerca di garantire le 3 proprietà appena indicate. Gli algoritmi di questo tipo più noti sono:

- MD4 (*rotto*, si noti il 2^{20} nella colonna *Collisioni* nella tavola seguente, questo algoritmo è però la base di tutti i successivi)
- MD5 [16]
- SHA-1
- RIPEMD-160

Una tabella comparativa fra di loro è la seguente:

Nome	*Hash (in bit)*	*blocco (in bit)*	*Pre-immagini*	*Collisioni*	*Round x step*	*Velocità relativa*
MD4	128	512	2^{128}	2^{20}	3x16	1,00
MD5	128	512	2^{128}	2^{64}	4x16	0,68
SHA-1	160	512	2^{160}	2^{80}	4x20	0,28
RIPEMD-160	160	512	2^{160}	2^{80}	5x16	0,24

L'utilizzo degli algoritmi di Hash è principalmente per garantire la integrità dei documenti: inviando un documento ed il suo Hash ad un corrispondente, è possibile verificare se il documento (o l'Hash) sono stati modificati. Per poter prevenire attacchi con ad esempio la sostituzione sia del documento che dell'associato Hash, è necessario utilizzare protocolli che uniscono gli Hash ad algoritmi di cifratura. Discuteremo di alcuni di questi nella seconda parte di queste note.

Un esempio semplice di utilizzo pratico degli Hash per garantire l'integrità ed in un certo senso anche l'identità di un documento è il seguente. Supponiamo di avere un documento con valore contrattuale in forma elettronica. Il documento può essere molto lungo, per questo i due contraenti si accordano di identificare il documento attraverso il suo Hash. Potrebbe quindi esistere un documento di una sola pagina firmato di proprio pugno in cui si dice che le regole dettagliate ecc.ecc. sono indicate nel documento elettronico identificato da un particolare Hash ottenuto con un particolare algoritmo di Hash. Ora se uno dei due contraenti volesse imbrogliare l'altro, dovrebbe riuscire a modificare il documento in modo a lui favorevole senza però che l'Hash del documento cambi. Come abbiamo visto in linea teorica questo è possibile ma *in pratica* è *impossibile* in quanto l'unico modo è l'attacco di forza bruta, ovvero provare a calcolare l'Hash di versioni diverse del documento. In questo caso, il numero di versioni diverse da preparare e di cui calcolare l'Hash è enorme, e richiede risorse computazionali non disponibili né oggi né nel prossimo futuro.

Gli algoritmi di Hash possono anche essere utilizzati per l'autenticazione. Un modo semplice descritto nell'RFC-2104 e denominato H-MAC è basato sulla seguente

16 Alla conferenza Crypto 2004 (Agosto 2004) sono stati annunciati nuovi risultati che mettono in dubbio la sicurezza di MD5 e di altri algoritmi crittografici di Hash. Al momento in cui scriviamo questo documento non sono ancora del tutto chiare le conseguenze di questi nuovi attacchi. E' però consigliato non utilizzare più MD5 e passare a SHA-1 od ancora meglio SHA-256, i quali, al momento, sembrano non avere le stesse debolezze.

idea. Supponiamo di avere i soliti due corrispondenti che vogliono essere sicuri che i messaggi che ricevono sono stati inviati dall'altro. I due corrispondenti possono scambiarsi una chiave segreta, analoga a quelle utilizzate per gli algoritmi simmetrici. Prima di inviare un documento all'altro, ciascuno dei due appende al documento la chiave segreta e calcola l'Hash del documento con la chiave segreta appesa. Invia poi al corrispondente il documento e l'Hash, ma ovviamente non la chiave segreta che hanno già entrambi. Se il ricevente calcolasse l'Hash solo del documento, questo risulterebbe diverso dall'Hash ricevuto. Invece anche il ricevente appende al documento la chiave segreta di cui è in possesso e ne calcola l'Hash che confronta con quello ricevuto. In questo modo si verifica non solo l'integrità del documento ma anche che il mittente è in possesso della chiave segreta, ovvero anche l'autenticità del documento. Se solo un'altra persona è in possesso della chiave segreta, allora l'identità del mittente è stabilita.

Un altro modo per stabilire l'autenticità di un documento è utilizzare le firme digitali generate con algoritmi Asimmetrici, di cui discuteremo nella seconda parte di queste note.

PARTE 2 – ALGORITMI E PROTOCOLLI CRITTOGRAFICI OGGI E DOMANI

Il titolo di questa parte è molto ambizioso, ma non siamo preveggenti ed il passato ci ha sicuramente insegnato che le sorprese sono sempre possibili. In realtà l'intento di questa seconda parte delle nostre note è, a partire dalla situazione ad oggi, di guardare ai problemi ancora insoluti, a quelli che stanno nascendo e cercare di capire che ruolo potrebbero giocare gli algoritmi e protocolli crittografici nel risolverli. Nel far questo discuteremo brevemente come gli algoritmi descritti nella prima parte sono utilizzati in pratica all'interno dei più comuni protocolli.

2.1 La Sicurezza dei Principali Algoritmi

Come abbiamo già ampiamente ripetuto, nessun algoritmo eccetto OTP è matematicamente sicuro. Gli algoritmi simmetrici e gli Hash *praticamente sicuri* sono tutti vulnerabili agli attacchi di forza bruta, per definizione. Quindi la loro sicurezza è legata alla lunghezza delle chiavi utilizzate per gli algoritmi simmetrici, alla lunghezza dell'impronta per gli algoritmi di Hash.

Poiché AES è un algoritmo molto nuovo, vi sono molte discussioni sulla sua reale sicurezza. In particolare alcuni ritengono che la semplicità strutturale di AES, che lo rende facilmente studiabile dal punto di vista matematico, possa divenire il suo punto debole. Infatti è stato possibile rappresentare AES come un (complicato) problema algebrico tramite il quale la chiave potrebbe essere ottenuta dalla soluzione di un set di equazioni algebriche. Questo ha dato origine ad una nuova famiglia di attacchi detti appunto **Algebrici** di cui quello relativo ad AES è chiamato XSL. Anche se XSL esiste, al momento non è implementabile ne si sa se lo sarà mai. Tra l'altro XSL si basa su di un sistema di moltissime equazioni algebriche non-lineari accoppiate, ed è noto che molti di questi sistemi sono praticamente irrisolvibili. La maggior parte degli studiosi è comunque fiduciosa nella sicurezza di AES, al più vi sono proposte di aumentare il numero di round (non meno di 13), la lunghezza del blocco a 256 bit e la lunghezza della chiave a 352 bit. Malgrado tutto ciò, ad oggi AES è uno degli algoritmi più sicuri e che potenzialmente offre di rimanere tale per molti anni.

Passando agli algoritmi Asimmetrici, abbiamo già detto che per molti di loro gli algoritmi di attacco più efficienti diventano praticamente non implementabili con il crescere della lunghezza della chiave. Vogliamo ricordare però che questo non vuol dire che per chiavi corte sia difficile risolvere il problema, anzi di solito vale il contrario. Tutti sappiamo fattorizzare il numero 21 ! Rimanendo all'esempio della fattorizzazione in numeri primi, per chiavi molto corte l'attacco di fattorizzazione è sicuramente molto facile, pertanto è necessario utilizzare chiavi piuttosto lunghe, tali che rendano impossibili gli attacchi più efficienti. Per gli algoritmi basati sulla fattorizza-

zione in numeri primi come RSA, il migliore attacco al giorno d'oggi per chiavi con più di 110 cifre decimali è il *Number Field Sieve* proposto inizialmente da Pollard nel 1988. La complessità tecnica di questo algoritmo non ci permette nemmeno di affrontarne una parziale descrizione, possiamo indicare però che la momento i numeri più lunghi fattorizzati sono di circa 500 bit, ad esempio in Dicembre 2003 è stata annunciata la fattorizzazione del numero noto come RSA-576 di 576 bit (o 174 cifre decimali). A breve ci si aspetta la fattorizzazione di numeri di 600/700 bit.[17] Per questo motivo è oggi consigliato utilizzare chiavi Pubbliche di almeno 1024 bit con algoritmi che basano la loro sicurezza sul problema della fattorizzazione in numeri primi.

La tipica domanda che viene sempre posta è quale è la lunghezza delle chiavi da adottare per i vari algoritmi. Una risposta assoluta non esiste in quanto dipende anche da che livello di sicurezza si vuole ottenere e per quanto tempo nel futuro queste chiavi devono rimanere sicure. Dobbiamo considerare di quali mezzi possono disporre i nostri avversari sia oggi che per tutto il tempo per il quale riteniamo che i dati che vogliamo proteggere siano significativi. Vista la velocità con cui la potenza di calcolo è cresciuta negli ultimi anni, non è facile dare delle indicazioni per lunghi periodi, e quindi in generale ci dobbiamo limitare a considerare un breve arco temporale futuro. Bisogna ricordarsi comunque che, a parte l'OTP, nessun algoritmo garantisce che l'informazione sarà protetta per sempre. Ci vorranno magari molti anni, ma prima o poi sarà possibile decifrare qualunque cosa cifrata oggi.

Detto ciò possiamo dire che chiavi simmetriche sotto i 100 bit sono oggi a rischio, nel senso che oggi non sono ancora possibili attacchi di forza bruta su algoritmi con chiavi di 80 o 100bit, ma continuando la crescita di potenza di calcolo attuale, tra pochi anni questi saranno possibili. Equivalentemente oggi sono a rischio chiavi Asimmetriche sotto i 1024 bit per algoritmi che si basano sulla fattorizzazione in numeri primi.

Possiamo concludere riportando alcune Tabelle proposte in varie occasioni a proposito della relazione tra la lunghezza delle chiavi simmetriche e di quelle Asimmetriche (per algoritmi basati sulla fattorizzazione o sui logaritmi discreti). Come sempre in questi casi è difficile paragonare una cosa all'altra, ancora meno fare delle previsioni sul futuro. Abbiamo volutamente riportato opinioni discordanti, almeno nel dettaglio, per rendere più chiaro che quello che è importante è la scelta personale fatta rispetto alle richieste di sicurezza di cui si ha bisogno. Basandosi sulle informazioni che riportiamo, riteniamo che sia possibile per ognuno farsi almeno una idea di come comportarsi.

Le seguenti due tabelle sono prese da Applied Cryptography di B. Schneier, i dati non sono molto recenti, risalgono al 1996 e sono precedenti alle ultime stime dello stesso Schneier sulla velocità del Number Field Sieve per cui le indicazioni riguardo gli algoritmi Asimmetrici dovrebbero essere considerate come *ottimistiche* almeno.

17 Vogliamo ricordare che con gli attuali algoritmi il numero di operazioni da effettuare aumenta moltissimo con l'allungarsi dei numeri in input, quindi passare da fattorizzare numeri di 600 bit a numeri di 1000 bit non è per nulla facile.

Symmetric and Public-key Key Lenghts with similar resistances to brute force attacks (in bits, * broken) - 1996	
Symmetric	Public Key
56*	384
64*	512
80	768
112	1792
128	2304

Recommended Public-key Key Lengths (in bits) - 1996		
Year	Length Min	Length Max
1995	768	1536
2000	1024	1536
2005	1280	2048
2010	1280	2048
2015	1536	2048

La seguente tabella proviene invece dall'RFC 3766 del 2004:

Symmetric key size in bits	***RSA or DH modulus size in bits***
70	947
80	1228
90	1553
100	1926
150	4575
200	8719
250	14596

Infine riportiamo una tabella del NIST del Dicembre 2002:

	Size in bits					
Sym. Key	56	80	112	128	192	256
Hash	160		256		384	512
MAC	64	160	256		384	512
RSA/DSA	512	1k	2k	3k	7.5k	15k
EC	160		224	256	384	512

Sym. Key: Symmetric key encryption algorithms
MAC: Message Authentication code
Pub. Key: Factoring or discrete log based public key algorithms
EC: Elliptic Curve based public key algorithms
White background: currently approved FIPS
Yellow background: under development
Black background: not secure now

2.2 Il Prossimo Sviluppo degli Algoritmi Crittografici

Abbiamo già parlato di AES e dello stato dello sviluppo degli algoritmi simmetrici. Se non succedono cose del tutto inaspettate, nei prossimi anni continuerà l'adozione ed al contempo lo studio di AES, dei suoi pregi e dei suoi difetti.

La situazione è differente per gli algoritmi Asimmetrici e quelli di Hash. Per gli algoritmi Asimmetrici quali RSA o ElGamal, vi sono vari aspetti che richiedono attenzione e potrebbero portare in breve tempo a qualche novità. In primo luogo un problema pratico di questi algoritmi è la necessità di utilizzare chiavi molto lunghe, come abbiamo visto di non meno di 1024 bit. Anche se i moderni processori hanno delle velocità sorprendenti e i dispositivi di memoria crescono anch'essi in modo impressionante, generare, gestire e mantenere chiavi così lunghe non è sempre facile.

Non solo, nel prossimo futuro ci saranno molti dispositivi piccolissimi che comunicheranno con noi ed i nostri elaboratori, pertanto anche in essi ci dovranno essere processori *embedded* in grado di eseguire operazioni crittografiche. Per questo vi è sicuramente interesse a migliorare le prestazioni degli algoritmi Asimmetrici e trovarne altri che permettano l'uso di chiavi più corte. Una possibilità ormai studiata da alcuni anni anche se per il momento non ha trovato ancora grandi applicazioni pratiche, è quella di utilizzare la matematica delle *curve Ellittiche* (o *Hyper-ellittiche*). Questa matematica è sicuramente più complicata di tutte quelle che abbiamo citato sinora, ma come al solito potrebbe portare dei nuovi risultati. In particolare se si formulano gli algoritmi Asimmetrici utilizzando questa matematica, dovrebbe essere possibile ottenere delle semplificazioni nell'implementazione degli algoritmi, e quindi più velocità e minore risorse utilizzate, e la possibilità anche di usare chiavi poco più lunghe di quelle adottate per gli algoritmi simmetrici.

La situazione per gli algoritmi di Hash è simile a quella per gli algoritmi Asimmetrici. Attualmente MD5 genera un Hash troppo corto per poter essere sicuro per sufficiente tempo nel futuro. SHA-1 e RIPEMD-160 generano un Hash di 160 bit, sono quindi ad oggi più sicuri ma sono molto lenti (si veda la tabella nella sezione 1.3.6). Ma anche questi Hash sono in pericolo visto che la loro sicurezza è più o meno equivalente a quella di un algoritmo simmetrico con chiave di 80 bit, e come indicato anche dalla Tabella del NIST riportata nel precedente capitolo, è necessario passare al più presto ad Hash di almeno 256 bit. Infatti nello standard FIPS-180-2 (http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf del 2002 ma modificato nel 2004) sono proposte versioni di SHA con 224, 256, 384 e 512 bit. A breve quindi MD5 dovrebbe essere abbandonato, e sarebbe meglio passare ad utilizzare SHA-224 o SHA-256 piuttosto che SHA-1. Purtroppo SHA non è particolarmente efficiente ed in molte applicazioni si preferisce ridurre la sicurezza piuttosto che la velocità. Sarebbe pertanto utile se fossero sviluppati altri algoritmi di Hash altrettanto o più sicuri ma al contempo più veloci.

La necessità di sviluppare nuovi algoritmi di Hash è stata resa più drammatica dai risultati annunciati alla conferenza Crypto 2004 (Agosto 2004) e già citati nella sezio-

ne 1.3.6. Questi risultati mettono in dubbio la sicurezza di MD5 [18] e di altri algoritmi crittografici di Hash, identificando delle debolezze di progettazione comuni a quasi tutti gli algoritmi attuali. Al momento sembra che la famiglia degli algoritmi SHA, a partire da SHA-1 (SHA-0 è invece rotto), sia immune da questi problemi, anche se gli esperti si aspettano di scoprire debolezze simili anche per questi algoritmi. In pratica al momento è consigliato non utilizzare più MD5 e passare a SHA-1 od ancora meglio SHA-256, mentre per il futuro bisogna attendere che la comunità scientifica crei dei nuovi algoritmi di Hash crittografico.

Dobbiamo infine accennare ad altri algoritmi specializzati per risolvere particolari problemi. Il principale e più famoso di questi algoritmi è quello di Diffie-Hellman. L'algoritmo di Diffie-Hellman (1976) fu il primo algoritmo Asimmetrico a chiave Pubblica inventato ed esso risolve il problema della creazione e scambio di chiavi segrete. Questo algoritmo permette di creare e scambiarsi una chiave segreta di sessione senza però nessun metodo di autenticazione tra le parti. In pratica i due corrispondenti si scambiano le proprie “chiavi Pubbliche” ed ognuno effettuando un calcolo con la chiave Pubblica dell'altro e la propria chiave Privata ottiene la stessa chiave segreta di sessione. Chiunque può esaminare i dati scambiati fra i due, infatti chi è in possesso solo delle chiavi Pubbliche non è in praticamente in grado, come nel caso di RSA, di ottenere una delle chiavi Private, e con le sole chiavi Pubbliche non è possibile calcolarsi la chiave segreta di sessione. L'algoritmo di Diffie-Hellman si basa sul problema dei logaritmi discreti ed è una alternativa al procedimento di scambio di chiavi via un algoritmo quale RSA che abbiamo descritto precedentemente. Si noti inoltre che con Diffie-Hellman nessuno dei due corrispondenti può scegliere la chiave segreta di sessione che dipende dalle chiavi Private di entrambi. Invece se si usa RSA, uno dei due sceglie una chiave segreta di sessione e l'invia all'altro cifrandola con la chiave Pubblica del corrispondente.

Algoritmi quali quello di Diffie-Hellman ci portano a considerare una nuova e grande classe di algoritmi, quelli specializzati a risolvere un problema particolare. Non è questa la sede per una lista o descrizione di questi algoritmi. E' chiaro comunque che essi vengono sviluppati appositamente per risolvere un problema specifico. Di solito si hanno quindi due possibilità, dato un problema è possibile risolverlo

1. sviluppando un protocollo che utilizza uno o più algoritmi crittografici
2. sviluppando un particolare algoritmo che risolve esclusivamente questo problema.

Lo sviluppo di questi algoritmi è molto legato quindi a quali sono i principali problemi aperti che potrebbero essere risolti con l'utilizzo della crittografia. Ne riparleremo quindi nelle prossime sezioni.

18 Si noti, tra l'altro, il sito http://passcracking.com/, apparso nel Giugno del 2004, ove dato un Hash MD5 è possibile ricostruire la stringa originaria se questa è composta di 8 caratteri minuscoli o numeri.

2.3 Protocolli ed applicazioni, da oggi a domani

2.3.1 Algoritmi e Protocolli

Sinora ci siamo occupati prevalentemente di algoritmi crittografici. Questi però singolarmente non bastano nelle applicazioni pratiche. Vediamo di illustrare questo punto con gli esempi più semplici e comuni.

Possiamo combinare l'uso di un algoritmo Asimmetrico e di un Hash per creare una Digital Signature, ovvero una firma digitale di un documento.[19] Una firma digitale garantisce l'autenticità e la integrità del documento a cui è associata. Per crearla il procedimento è il seguente

1. dato il documento si calcola il suo Hash
2. usando la **propria chiave Privata** e l'algoritmo Asimmetrico si cifra l'Hash del documento

La firma digitale è costituita dall'Hash del documento cifrato con la propria chiave Privata. Si noti che in questo caso si usa la propria chiave Privata, mentre nei casi considerati precedentemente di cifratura di dati abbiamo usato la chiave Pubblica del corrispondente. Il motivo è che nei casi precedenti volevamo garantire la confidenzialità e quindi solo il possessore della chiave Privata era in grado di decifrare il documento. In questo caso invece vogliamo garantire la autenticità, quindi solo il possessore della chiave Privata può aver cifrato l'Hash così identificandosi, mentre chiunque può verificarlo utilizzando la chiave Pubblica. Per convincersi di questo, descriviamo il procedimento di verifica della firma digitale:

1. chi riceve il documento e l'associata firma digitale per prima cosa si procura la chiave Pubblica del mittente
2. con la **chiave Pubblica del mittente** decifra l'Hash
3. calcola indipendentemente l'Hash del documento
4. confronta l'Hash che ha calcolato con quello decifrato, se coincidono ha verificato l'integrità del documento e la sua origine poiché solo chi è in possesso della chiave Privata può aver creato la firma digitale.

Ci sono due punti estremamente importanti, anche se sottili, da sottolineare

1. *la firma digitale identifica chi ha firmato il documento, questa persona può essere diversa da chi ha creato il documento, in altre parole la firma digitale è solo un modo di associare una chiave Privata ad un documento*
2. *la firma digitale non associa una persona ad un documento ma solo una chiave Privata, chiunque può essere il possessore della chiave Privata, non vi è modo di scoprire il possessore della chiave Privata utilizzando solo la firma digitale appena descritta.*

Questo punto è molto importante in pratica, e da qui sono nati ad esempio i **Certifi-**

19 Il termine Firma Digitale è qui inteso nel puro senso tecnico e senza alcuna rilevanza legale.

cati Digitali e le associate strutture per gestirli (Certificate Authorities, Public Key Infrastructures ecc.). Comunque il problema dell'associazione di una persona od ente ad una chiave Privata non è risolvibile solo con strumenti elettronici, richiede anche l'intervento di procedure umane. Purtroppo questo è un problema molto complicato che non ha ancora trovato una soluzione del tutto soddisfacente.

Continuando in questo esempio, è facile utilizzare tutti i tipi di algoritmi descritti sinora per realizzare una comunicazione che garantisce **confidenzialità, integrità e autenticità** (nelle forme appena descritte). Supponiamo ancora di dover scambiare un documento, sia questo un messaggio di posta elettronica od un pacchetto da inviare sulla rete di comunicazione o qualche altro formato elettronico. Possiamo procedere come segue

1. generare una firma digitale del documento e concatenarla allo stesso
2. generare una chiave (casuale) segreta di sessione
3. utilizzare la chiave di sessione ed un algoritmo simmetrico per cifrare il documento e la sua firma digitale
4. utilizzare un algoritmo Asimmetrico e la chiave Pubblica del nostro corrispondente per cifrare la chiave di sessione
5. inviare il documento+firma digitale cifrati con l'algoritmo simmetrico e la chiave di sessione cifrata con l'algoritmo Asimmetrico al nostro corrispondente.

Il nostro corrispondente, in possesso della propria chiave Privata, decifra la chiave di sessione (solo lui può farlo) e con questa decifra il documento e la firma digitale. Poi con la chiave Pubblica del mittente, il ricevente verifica la firma digitale. In questo modo confidenzialità, integrità e autenticità sono garantite.

L'implementazione nei dettagli di questa procedura dipende dal protocollo, ma la maggior parte dei protocolli utilizzati oggi, da PGP, a IPSEC a SSL ecc., sono basati su di un procedimento simile. Ad esempio, molti protocolli utilizzati per cifrare le comunicazioni in rete, quali IPSEC, svolgono una procedura simile in due fasi. Nella prima fase, utilizzando un algoritmo Asimmetrico viene scambiata, o generata, una chiave segreta casuale di sessione. Questa fase viene ripetuta periodicamente per cambiare la chiave segreta di sessione in modo da non utilizzarla per lungo tempo o per molti dati. Nella seconda fase i dati vengono cifrati con un algoritmo simmetrico e la chiave segreta di sessione, e poi vengono scambiati. Inoltre in molti casi la firma digitale è sostituita con un protocollo H-MAC, descritto nella sezione 1.3.6, che in molte situazioni si rivela più efficiente e veloce.

Nell'Appendice A diamo una breve descrizione di OpenPGP che è, a nostro parere, indicativa di molti dei problemi che si incontrano nel tradurre un protocollo teorico ad alto livello in un programma che implementa delle procedure crittografiche.

2.3.2 Certificati Digitali, Certification-Authorities e Web

Dopo aver descritto gli algoritmi crittografici e dato alcune indicazioni su come combinarli per creare dei protocolli, in questa sezione passiamo direttamente a considerare un caso pratico dal punto di vista dell'utente finale. Con questo vorremmo evidenziare al lettore come il combinare gli elementi di base della crittografia per ottenere un prodotto finale, sia un processo lungo, complesso e irto di difficoltà. Come abbiamo già indicato varie volte, le difficoltà provengono per lo più dall'interazione tra l'uomo, od i processi ad alto livello, e gli algoritmi crittografici, e spesso i punti critici risiedono nelle procedure ad alto livello che permettono di utilizzare in pratica la crittografia.

Il problema di cui vogliamo discutere è quello della sicurezza della navigazione web restringendosi in particolare a quello degli acquisti con carta di credito. Ci poniamo quindi nella posizione di un utente non esperto di informatica né tanto meno di sicurezza informatica che vuole usare la propria carta di credito per fare un acquisto tramite un portale web. Si pongono come minimo due problemi:

1. i dati della carta di credito non devono essere intercettati nel loro tragitto tra il browser dell'utente ed il server su cui è ospitato il portale web
2. l'utente vuole essere sicuro che il server a cui invia i dati della propria carta di credito è proprio quello dell'azienda presso la quale vuole fare l'acquisto.

Se il primo punto può essere risolto cifrando la connessione con le tecniche descritte nella sezione precedente, il secondo punto è molto problematico. Il problema sorge dal fatto che per creare il tunnel cifrato il primo passo da parte dell'utente è quello di utilizzare la chiave pubblica del server per verificare che il server è in possesso della corrispondente chiave privata, e poi per scambiarsi le chiavi di sessione usate per cifrare i dati con gli algoritmi simmetrici. In questo modo l'utente ha ottenuto di creare un tunnel cifrato con il possessore della chiave privata corrispondente alla chiave pubblica di cui è in possesso.

> *Ma **chi è** il possessore di questa coppia di chiavi Asimmetriche? E' il server dell'azienda presso cui l'utente vuole fare l'acquisto o qualcun altro che vuole conoscere i dati della carta di credito?*

E' necessario quindi associare un ente od una persona alla chiave pubblica come il possessore della corrispondente chiave privata. La soluzione utilizzata oggi si basa sull'utilizzo di

1. un ente certificatore
2. dei certificati digitali.

In pratica l'ente certificatore, detto comunemente CA (Certification Authority), verifica che un certo ente o persona sia in possesso di una chiave privata, ed in questo caso prepara un certificato pubblico che è formato principalmente da:

1. la chiave pubblica dell'ente o persona
2. i dati dell'ente o persona (nome, indirizzo, URL se si tratta di sito web

ecc.)

3. il tutto firmato digitalmente con la chiave privata dell'ente certificatore.

Quindi l'ente certificatore garantisce l'associazione della chiave pubblica, e della relativa chiave privata, ad un ente o persona. Chiunque è in possesso della chiave pubblica dell'ente certificatore può verificare la firma digitale di un certificato e quindi accettare come autentico il certificato. Basta quindi che il nostro utente *si fidi* della CA e verifichi che il certificato sia autentico per poter essere sicuro di chi è il proprietario della chiave pubblica.

Sembrerebbe dunque che il problema sia risolto. Ci sono degli enti, le CA, di cui ci fidiamo che garantiscono l'identità dei possessori delle chiavi pubbliche. Purtroppo in pratica questa soluzione ha finora funzionato potremmo dire a metà.

Già il fatto che le CA siano degli enti commerciali, che possano essere in qualunque nazione del mondo, e che chiunque possa volendo creare la propria CA, rende il fatto di *fidarsi* di una CA non così banale come potrebbe sembrare.

A parte ciò, vediamo cosa succede in un browser qualunque configurato in maniera standard come potrebbe essere quello del nostro utente comune ed inesperto. All'interno dei Browser vi sono le chiavi pubbliche delle CA più note.[20] E' il produttore del software quindi che decide di quali CA inserire le chiavi pubbliche nel proprio prodotto. Come vedremo in dettaglio, questo inserisce un altro anello nella catena della fiducia, il nostro utente si deve fidare anche del produttore del browser che ha effettuato questa scelta per lui.

Quando il browser si connette ad un server utilizzando il protocollo SSL o TLS, per prima cosa il server invia al browser il proprio certificato digitale emesso da una CA. Il browser, se configurato in modo normale, verifica se possiede la chiave pubblica della CA che ha firmato il certificato del server e prosegue così

- se il browser possiede la chiave pubblica della CA, e quindi vuol dire che il browser conosce la CA e si fida della CA,[21] verifica il certificato del sito web e se la verifica è positiva crea il tunnel cifrato e mostra le pagine all'utente; nessuna informazione viene data all'utente eccetto l'apparire del simbolo del lucchetto chiuso in un angolo della finestra; se la verifica del certificato fallisce il browser informa l'utente e chiede come proseguire;
- se il browser non possiede la chiave pubblica della CA che ha firmato il certificato, mostra all'utente i dati contenuti nel certificato e chiede all'utente se vuole proseguire lo stesso o meno; l'utente può anche memorizzare nel browser il certificato del server in modo che la prossima volta che verrà stabilito il tunnel, il browser riconoscerà direttamente il certificato e non chiederà più nulla all'utente.

Di nuovo questa procedura sembra essere del tutto ragionevole e corretta. Purtroppo entrambe le possibilità sono facilmente aggirabili usando tecniche non crittografiche ma bensì umane. Nel primo caso il malfattore può acquistare legalmente presso una CA (ad esempio di un altro paese) un certificato per un sito con un nome simile

20 In realtà all'interno dei browser vi sono i certificati self-signed delle CA, e questi contengono le chiavi pubbliche.

21 Ovviamente chi si fida della CA è chi ha inserito la chiave pubblica della CA nel browser, ovvero il suo produttore; per semplicità parleremo nel proseguio semplicemente del browser.

a quello del sito che vuole impersonare. Costruisce poi un sito identico nell'aspetto a quello della vera azienda e con qualche trucco od email convince l'utente a connettersi al proprio sito. Visto che il server fornisce un certificato firmato da una CA riconosciuta dal browser, l'utente non viene informato di nulla e può rendersi conto della truffa solo dal fatto che l'URL è simile ma non identica a quella usuale. Vista la complicazione delle URL dei siti di commercio elettronico è molto difficile se non quasi impossibile che un utente normale si possa accorgere della differenza.

In alternativa il truffatore può creare la propria CA e generare il certificato per il proprio server. Quando il browser dell'utente prova a connettersi al server del truffatore, riceve un certificato di una CA di cui non ha la chiave pubblica. Il browser chiede allora all'utente cosa fare. Ovviamente nel certificato il truffatore ha messo i dati esatti dell'azienda che vuole imitare, così che agli occhi del nostro utente i dati presenti nel certificato risultano corretti, sono proprio quelli dell'azienda presso la quale vuole fare acquisti! A questo punto alcuni utenti saranno in dubbio se proseguire o meno, visto che di solito quando fanno acquisti non compare questa richiesta di conferma da parte del browser, corredata da avvertimenti che dicono che il browser non conosce e non si fida della CA che ha creato il certificato. Vi sono quindi utenti che nel dubbio non si fidano del certificato e rinunciano, ed altri che invece proseguono senza pensarci su troppo.

In conclusione, il fatto che il browser si fidi delle CA che ha in memoria non garantisce la sicurezza delle connessioni ai siti di commercio elettronico. Il problema infatti è che è il browser a fidarsi delle CA e non il nostro utente che per lo più è all'oscuro di tutto.

Purtroppo questi attacchi sono all'ordine del giorno, il cosiddetto *Phishing* è basato su tecniche simili che portano all'accesso da parte dei malfattori agli account di Internet Banking dei nostri ignari ed a questo punto indifesi utenti.

Chiaramente in questa sezione non siamo entrati nei dettagli dei protocolli per la generazione, gestione ecc. dei certificati digitali, né per la creazione di comunicazioni cifrate tramite SSL/TLS. Il nostro scopo era dare un esempio di come sia difficile non solo creare algoritmi crittografici sicuri, ma poi anche usarli in modo da risolvere problemi pratici. Ovviamente più ci si sposta nella scala uomo-macchina verso l'uomo, più le difficoltà da risolvere diventano anche di carattere organizzativo e gestionale, ed introducono spesso dei problemi che la matematica non può risolvere da sola.

2.3.3 Problemi Aperti ed Applicazioni

Questa sezione potrebbe sembrare un po' una lista della spesa, vogliamo però indicare succintamente alcune problematiche di sicurezza la cui soluzione potrebbe richiedere l'utilizzo della crittografia. La possibilità che la crittografia possa aiutare a risolvere un problema di sicurezza ovviamente spinge i crittografi a ideare nuovi algoritmi o protocolli.

Come abbiamo ripetuto più volte, molto spesso la crittografia da sola non risolve il problema ma è solo uno strumento spesso molto efficace per la soluzione. Quindi il

fatto che esista un algoritmo od un protocollo che possa essere usato per risolvere un problema di sicurezza non vuol dire che il problema sia risolto, anzi come vedremo ci sono casi ove la complicazione maggiore non è nella parte che può essere risolta dalla crittografia. In pratica spesso se un problema di sicurezza può essere risolto quasi esclusivamente con la crittografia, vuol dire che questo problema è *semplice*. Per i problemi *difficili* la crittografia risolve la parte facile, ma lascia insoluta la parte difficile.

L'intenzione di questa lista è di dare una idea di dove possa andare la crittografia nei prossimi anni. E' chiaro che alcuni problemi rimarranno insoluti, altri saranno tralasciati per mancanza di interesse (cambio di tecnologie o altro motivo), altri troveranno una soluzione soddisfacente o anche solo passabile. E' ovvio anche che questa lista non ha nessuna presunzione di essere completa, anzi ne abbiamo escluso tutti i problemi tecnici direttamente crittografici ed abbiamo omesso del tutto la crittoanalisi. Pensiamo che comunque queste indicazioni possano risultare utili a fornire un panorama della crittografia oggi e probabilmente anche nel prossimo futuro.

- **Instant Messaging (chat)** La comunicazione attraverso questi canali elettronici si sta diffondendo molto, ma come molte altre volte per quanto riguarda le tecnologie informatiche e quelle connesse ad internet in particolare, all'inizio non si era prevista nessuna forma di sicurezza; i problemi aperti sono principalmente quello della identificazione dei partecipanti e quello della confidenzialità dei contenuti; il problema della identificazione dei partecipanti non è semplice perché in alcuni casi si vuole al contempo proteggere l'anonimato ma garantire allo stesso tempo l'autenticità degli stessi, in altre parole che le persone siano proprio quelle che ci si aspetta; le tecnologie in questo campo stanno evolvendo velocemente ed a breve la situazione potrebbe cambiare.
- **Peer-2-Peer (P2P)** Le cronache dei giornali riportano spesso dell'uso fatto del P2P per distribuire musica o film piratati, ma i problemi di sicurezza del P2P non concernono solo la protezione dei dati scambiati in questi canali che in realtà possono essere usati per usi leciti con molto profitto; si delinea la creazione di due tipi di reti parallele di P2P, quelle per il business nelle quali l'autenticazione è di fondamentale importanza e quelle private ove l'anonimato è prevalente; in entrambi i casi già oggi è necessaria la protezione dei canali e dell'accesso agli stessi.
- **VPN e comunicazioni dati** lo sviluppo dei tunnel cifrati che collegano diversi corrispondenti in modo sicuro attraverso una rete non sicura (si pensi ad esempio al wireless) è in aumento; al momento i due protocolli utilizzati principalmente sono IPSEC e SSL/TLS, entrambi hanno i propri problemi e quindi è auspicabile che compaia uno o più successori che li integrino e migliorino al tempo stesso; una direzione di sviluppo è quella di IPv6 ove la cifratura del canale è già inclusa direttamente nel protocollo di comunicazione; questo infatti sembra essere il trend per il futuro perché permetterebbe di connettere in modo sicuro qualunque tipo di apparecchio, dagli elaboratori ai telefonini, PDA ecc. basando la sicurezza unicamente sul protocollo di comunicazione; d'altra parte questo introduce un punto critico unico sul quale si basa la sicurezza; la recente esperienza negativa del WEP per le reti WiFi genera molti dubbi sul puntare ad un unico protocollo sia per la trasmissione dei dati che per la loro sicurezza, pertanto non è chiaro quale sia la direzione giusta da percorrere.

- **VoIP e comunicazioni voce** I problemi legati alla convergenza voce e dati sullo stesso protocollo, in particolare IP, sono molteplici, quello che ha sicuramente la prevalenza è la qualità del servizio (QoS) che sebbene è parte della sicurezza informatica di solito non ha nulla a che fare con la crittografia; con l'esclusione di qualche ambiente particolare, le esigenze di confidenzialità delle comunicazioni voce oggi non sono molto sentite; quello che però potrebbe riscuotere maggiore interesse è l'autenticazione (almeno dei device all'estremo delle comunicazioni) e l'integrità delle comunicazioni, ove la crittografia dovrebbe giocare un ruolo rilevante; per quanto invece riguarda i device mobili wireless, la crittografia già gioca un ruolo sia per l'autenticazione dei device che per la cifratura delle comunicazioni che in questo caso vuole prevenire la possibilità di facili intercettazioni; comunque tra gli esperti non vi è concordia su quanto importante sarà il ruolo della crittografia in questo campo.
- **Email** La sicurezza della posta elettronica è un tema caldo del momento, anche se più che i problemi di confidenzialità sono quelli legati allo spam ed ai Virus ad interessare; in realtà le soluzioni crittografiche per la confidenzialità esistono da tempo, quali PGP e S/MIME, ma la loro implementazione procede molto a rilento principalmente per due motivi: il primo è quello della infrastruttura per la distribuzione dei certificati, che soffre degli stessi problemi descritti nella precedente sezione sulla navigazione Web, il secondo è che la posta elettronica si sta configurando come una modalità diversa ma parallela di comunicazione voce, e come non vi è vero interesse a cifrare le telefonate se non in rari casi, così non vi è un grande interesse a cifrare i messaggi di posta elettronica; un problema più interessante ed attuale è quello invece della certificazione del mittente e del ricevente, ovvero di come realizzare la comune raccomandata con ricevuta di ritorno utilizzando la posta elettronica; l'Italia ha appena introdotto un protocollo di posta certificata che si propone proprio di risolvere questo punto, ma è sicuramente un inizio e ci si può aspettare che prossimamente ci siano molti sviluppi.
- **Spam** Il problema dello spam, o messaggi di posta non voluti, è un problema molto sentito ma anche molto complesso; esso nasce dal sommarsi di vari fattori quali: l'economicità dell'invio dei messaggi di posta elettronica, la possibilità di falsificare il mittente, l'apertura delle caselle elettroniche a ricevere qualunque messaggio ecc. ecc.; ben pensandoci queste sono anche caratteristiche delle normali lettere e caselle postali, a parte il fatto che il francobollo costa notevolmente di più dell'invio elettronico; da un punto di vista tecnico lo spam è posta *non voluta* ma non è posta *non autorizzata*, la differenza fra questi due punti può sembrare sottile, ma dal punto di vista della crittografia è possibile aiutare a risolvere il problema dell'autorizzazione ma molto meno quello del gradimento dei messaggi; vi è quindi molto scetticismo fra i crittografi che la crittografia sia l'arma giusta per combattere lo spam; certo la crittografia può giocare un ruolo nell'autenticazione dei server smtp tra di loro e così via, ma rimane difficile immaginare come possa fornire La soluzione al problema spam.
- **Virus** Analogamente al problema spam, la posta elettronica è anche il principale canale di diffusione dei virus elettronici, e di nuovo non è chiaro come la crittografia possa aiutare a risolvere il problema; possiamo pensare a ciò da due punti di vista, il primo è evitare che il messaggio che porta il virus arrivi o venga letto dal destinatario, il secondo di sostituire i programmi che usiamo al momento con altri più sicuri; nel primo caso la soluzione è più negli anti-virus che ormai tutti usiamo, anche obbligati dalla legge, ma la crittografia potrebbe aiutarci ad identificare i

mittenti; se tutti i messaggi che riceviamo dai nostri corrispondenti fossero firmati digitalmente, potrebbero sorgerci dei dubbi se ne ricevessimo uno non firmato, ... oppure no ??? Di nuovo è difficile immaginare come la crittografia possa fornire La soluzione al problema virus, mentre sicuramente può essere utile nel costruire questa soluzione.

- **Phishing e Web-surfing** Il problema delle credenziali e dell'autenticazione ha la sua massima espressione attualmente nel Phishing; in pratica si tratta di truffatori che, nella versione più semplice, mandano messaggi di posta elettronica facendo finta di essere la banca del malcapitato utente, nei quali chiedono di connettersi al sito di Internet Banking con qualche scusa; peccato che il sito sia invece una replica del vero sito della banca fatto dai truffatori che in questo modo riescono a ottenere le chiavi di accesso al conto corrente dal quale sottraggono di solito modiche cifre; questa è fondamentalmente la combinazione di un messaggio di spam con la debolezza dell'autenticazione nella navigazione Web, navigazione che è sì protetta per quanto riguarda la confidenzialità dai protocolli quali SSL/TLS ma che ha il suo punto debole nei certificati digitali che dovrebbero garantire l'autenticità; qui sicuramente i protocolli crittografici potranno essere d'aiuto, anche se finora le soluzioni trovate ed adottate si sono rilevate non all'altezza della situazione.
- **Sistemi Operativi *Sicuri*** [22] A prima vista potrebbe sembrare che la sicurezza dei Sistemi Operativi abbia poco a che fare con la crittografia: che ruolo potrebbe giocare la crittografia nel prevenire ad esempio i buffer-overflow ? In realtà il problema è molto complesso, e questo non è il luogo per discuterne in dettaglio, ma possiamo accennare alcuni spunti che indicano come la crittografia sarà un costituente fondamentale di un futuro Sistema Operativo *Sicuro*; in primo luogo, il processo di autenticazione degli utenti richiede la crittografia, oggi le password sono cifrate ma domani utilizzeremo token, smartcard, sistemi biometrici che in futuro saranno in grado di riconoscere la nostra voce od il nostro viso, ed in tutto questo la crittografia gioca e giocherà un ruolo importante; ma non solo, i sistemi operativi sono troppo complessi per essere veramente sicuri nella loro interezza, pertanto l'idea è che vi sarà una piccola parte *sicura* del SO che dovrà giudicare se un programma che deve andare in esecuzione o è in esecuzione è sicuro o no; la soluzione più banale è quella di permettere l'esecuzione solo di programmi che portano la firma digitale di qualcuno di *fidato* ed impedire l'esecuzione di programmi senza firma o con firme non riconosciute; sembrerebbe l'uovo di Colombo per evitare che worm, virus, rootkit ecc. possano fare danni, ma in realtà vi sono molti problemi con questo approccio alcuni dei quali sono: così facendo si finisce per non risolvere le falle dei programmi, che prima o poi verranno sfruttate in altro modo, la sicurezza sarebbe affidata totalmente a questa entità fidata, ma di chi ci possiamo fidare? se solo i programmi con la firma della persona/ente fidato possono essere eseguiti come sarà possibile far eseguire i *propri* programmi o i programmi di un altro fornitore ? che ne è della Privacy sul nostro elaboratore se demandiamo a qualcun altro la verifica di tutto ciò che viene eseguito ? ecc. ecc. Comunque è un argomento molto interessante, molto dibattuto ed in grande sviluppo, basta guardare a progetti quali Palladium/NGSCB, TCPA ecc.
- **Digital-Rights-Management (DRM)** Il problema della protezione dei diritti d'autore nelle opere distribuite digitalmente è molto sentito e richiede sicuramente la crittografia come elemento essenziale per la sua soluzione; è anche legato alla si-

22 Il termine **Trusted Operating System** è di solito riservato a Sistemi Operativi con certificazione di sicurezza quali ITSEC o Common-Criteria.

curezza dei Sistemi Operativi ed alcune proposte di SO Sicuri includono infatti anche la gestione dei diritti d'autore (alcuni dicono che sono fatte quasi esclusivamente per questo scopo); questo è comunque un campo molto vasto e difficile, perché non è limitato agli elaboratori ma anche a tutti gli apparecchi sui quali vengono e verranno distribuiti i contenuti multimediali, lettori portatili ecc.; la storia recente tra l'altro porta esempi di mezzi fiaschi, come ad esempio il sistema adottato per i DVD (CSS) che è risultato facilmente aggirabile. Si noti comunque, che per poter essere fruita, un'opera, si pensi al tipico brano musicale, deve essere decifrata sul device di riproduzione ove le protezioni possono risultare più facili da aggirare.

- **Public Key Infrastructure (PKI) e metodi di autenticazione** Come abbiamo già indicato più volte uno dei problemi principali che ricorre in molte situazioni è quello dell'autenticazione, in questo contesto più precisamente dell'associazione di una chiave Privata o comunque di un elemento crittografico ad una persona od ente; in linea di principio sembrerebbe un problema di facile soluzione, ma in pratica non lo è poiché richiede comunque l'intervento umano e diventa perciò non solo un problema crittografico sia dal punto di vista di algoritmi che protocolli, ma anche un problema organizzativo e procedurale; la struttura odierna delle PKI per la gestione dei certificati elettronici non ha dato i risultati sperati, anche solo per la navigazione Web; essa soffre di parecchi problemi sia dal punto di gestione, che di scala che di sicurezza effettiva per l'utente finale non esperto; attualmente non è chiaro come si svilupperanno o quali saranno i successori delle PKI, ma di sicuro è necessario progettare nuovi sistemi di autenticazione distribuiti su larga scala.
- **Biometria e Token di autenticazione** Gli strumenti biometrici ed i token di autenticazione avranno sicuramente un grande sviluppo nel prossimo futuro; è chiaro che l'uso capillare ed in un numero crescente a dismisura di apparecchi elettronici sta rendendo il paradigma *autenticazione = username + password* del tutto obsoleto ed inadatto; molti di questi strumenti sono sul mercato da anni, ma non hanno sinora ricevuto l'interesse che forse meritavano; certamente molto rimane ancora da fare, soprattutto nel campo della compatibilità, degli standard ecc.; per quanto riguarda la biometria vi sono ancora molti problemi, sia per quanto riguarda la Privacy che per la reale efficacia delle varie tecniche, sono entrambi problemi non strettamente crittografici, ma non è detto che la crittografia non possa dare una aiuto anche per la loro soluzione.
- **Carta d'identità elettronica, passaporto elettronico e simili** Queste carte arriveranno sicuramente, alcuni di noi le stanno già sperimentando ed i primi problemi sono già sorti; ideare e gestire questi strumenti non è per nulla facile sia per il livello di sicurezza che devono garantire, sia per la difficile unione tra disponibilità dei dati e Privacy; non ultimo devono essere strumenti semplici da usare; ci sarà quindi molto da lavorare e la crittografia avrà sicuramente un ruolo fondamentale.
- **Carta di credito elettronica e denaro elettronico** In modo molto simile alle carte d'identificazione, le carte economiche quali le carte di credito e i bancomat, sono già entrate nella nostra vita quotidiana; anche se molto comode, siamo tutti consci (soprattutto banche ed istituzioni finanziarie che alla fine sono quelli che hanno le maggiori perdite economiche) dei rischi che comportano; è chiaro quindi che avremo presto una nuova generazione di carte di credito anche elettroniche e che compariranno anche nuove versioni di denaro elettronico di vario tipo; ed è da

aspettarsi che a breve nasceranno varie generazioni di carte economiche, per le quali la crittografia offrirà ovviamente la base di costruzione.

- **Voto Elettronico** Questo è un altro argomento di grandissimo interesse, soprattutto negli Stati Uniti la sperimentazione ed implementazione del voto elettronico, in alcuni casi addirittura via internet, è già da tempo avviata, ma le opinioni sono molto discordanti; in linea generale molti politici ed amministratori sono favorevoli, mentre soprattutto i crittografi sono contrari ritenendo che la tecnologia odierna sia ancora molto immatura per offrire le sufficienti garanzie di sicurezza, privacy ecc. richieste per un momento così delicato della nostra vita civile; questo è sicuramente un campo ove nuovi protocolli ed algoritmi crittografici potrebbero aiutare grandemente allo sviluppo della tecnologia, anche se vi sono alcuni che ritengono che la tecnologia non sarà mai sufficiente perché è predominante il problema umano.
- **USB-Sticks, portable hard disks, distributed file systems, distributed computing ecc.** In quest'ultimo punto riassumiamo alcune recenti tecnologie di raccolta e distribuzione di dati ed informazioni; nel prossimo futuro sarà sempre più facile accedere da apparecchi diversi, mobili o fissi, a dati di varia natura, privati e pubblici; sarà pertanto necessario che la crittografia fornisca algoritmi, metodi e protocolli per garantire la sicurezza nell'accesso ai dati, in qualunque modo questi avvengano e da qualunque sorgente, ma anche per la loro protezione ed al contempo disponibilità; questo è chiedere molto, e purtroppo per il momento non sembra esserci molta sensibilità da parte di chi sta sviluppando queste tecnologie ai problemi di sicurezza, in linea di principio difficili, che essi comportano.

Per concludere questa sezione, è forse opportuno riportare quello che Adi Shamir, la S in RSA, ha detto nel suo discorso per il ricevimento del *Turing Award* della *Association for Computing Machinery* nel giugno del 2003. Shamir indicò quelle che a suo parere sono le 3 leggi fondamentali della crittografia:

1. *Absolutely secure systems do not exist*
2. *To halve your vulnerability, you have to double your expenditure*
3. *Cryptography is typically bypassed, not penetrated.*

Inoltre Shamir indicò quelli che, sempre a suo vedere, saranno i prossimi sviluppi della crittografia:

1. AES will remain secure for the foreseeable future
2. Some public-key schemes and key sizes will be successfully attacked in the next few years
3. Cryptography will be invisibly everywhere
4. Vulnerabilities will be visibly everywhere
5. Crypto research will remain vigorous, but only its simplest ideas will become practically useful
6. Non-crypto security will remain a mess.

PARTE 3 – LA CRITTOGRAFIA QUANTISTICA

In questa terza parte vogliamo introdurre un argomento certamente avanzato, ma di ovvio interesse. Nella sezione precedente abbiamo guardato alla crittografia rimanendo nell'ambito delle tecnologie e dell'approccio moderno ma al contempo classico, ovvero basate sulla evoluzione delle tecniche matematiche tradizionali.

Non è detto però che la crittografia in futuro debba basarsi unicamente su questo tipo di approccio, anzi è molto probabile che la confluenza della informatica con la crittografia porti a considerare altre discipline ed altre possibilità per lo sviluppo della crittografia oltre la matematica. Con questo non vogliamo dire che la matematica avanzata potrebbe in futuro non essere più la base della crittografia, ma solamente che vi potrebbero essere altre discipline che contribuiranno alla crittografia al pari della matematica.

Come sarà chiaro dalle prossime sezioni, la crittografia quantistica non si propone al giorno d'oggi come una rivoluzione nel campo della crittografia. Anche se già oggi vi sono i primi prototipi commerciali, solo il tempo potrà dirci se la crittografia quantistica diventerà un domani un elemento portante della crittografia o rimarrà un buon esperimento con però una piccola base di implementazione reale.

Quello che è importante a nostro parere, è la strada che la crittografia quantistica sta aprendo. Lo scopo principale di questa terza parte, più che descrivere nei dettagli cosa è la crittografia quantistica, è quindi quello di indicare al lettore una possibile rivoluzione alle porte della crittografia classica.

3.1 Perché la Fisica Quantistica

Prima di illustrare la crittografia quantistica, ci sembra giusto fare due passi indietro per cercare di chiarire l'ambito nel quale essa si colloca. Per fare questo ci sentiamo obbligati ad iniziare cercando di rispondere ad una difficile domanda:

cosa è la Meccanica Quantistica ?

La Meccanica Quantistica è al contempo una disciplina della Fisica ed una teoria formale. Lo scopo della Meccanica Quantistica è di rappresentare alcuni fenomeni fisici tramite un modello formale in modo da poterli descrivere ed essere in grado di fare predizioni utilizzando delle equazioni matematiche. In particolare la Meccanica Quantistica formula le leggi fondamentali delle particelle elementari.[23] Ma l'aspetto principale della Meccanica Quantistica che ci interessa è il suo ruolo nella fisica contemporanea. Nella prima metà dello scorso secolo si è capito che le leggi che regolano il comportamento delle particelle elementari sub-atomiche sono molto diverse da quelle a cui siamo abituati nella nostra esperienza quotidiana. La Meccanica Quantistica è una rivoluzione ben più fondamentale rispetto ad esempio alla sco-

23 Per essere precisi dovremmo specificare che la Meccanica Quantistica propriamente detta si occupa delle particelle elementari non-relativistiche, ovvero quando i fenomeni che si vogliono descrivere non hanno "*velocità*" vicine a quelle della luce. Altre teorie, quali la Teoria dei Campi, si occupano di fenomeni al contempo quantistici e relativistici, ma in queste note non le prenderemo in considerazione.

perta della Relatività da parte di Einstein. Ricordiamo che la teoria della Relatività di Einstein ha introdotto il concetto di una velocità massima in natura, la velocità della luce.

Le particelle elementari, così piccole che sono al di fuori della nostra esperienza quotidiana, possono essere rilevate solo attraverso speciali strumenti in modo indiretto. Non solo, esse seguono delle leggi che spesso vanno contro la nostra intuizione.[24] Infatti a tutt'oggi vi sono ancora degli scettici sulla correttezza della Meccanica Quantistica, ma i risultati di più di mezzo secolo di esperimenti non hanno fatto altro che continuare a confermare la fondatezza ed esattezza di questa teoria in maniera incontrovertibile.

Perché allora ci interessa in questa sede la Meccanica Quantistica? Fondamentalmente per due ragioni:

1. poiché le regole che governano le particelle elementari sono molto diverse da quelle *normali*, ne consegue che nell'ambito delle particelle elementari si possono fare delle cose che nel mondo macroscopico non sono possibili; si aprono quindi nuove possibilità per realizzare cose altrimenti non possibili;
2. gli elaboratori oggi sono costruiti con circuiti stampati sempre più piccoli, prima o poi arriveremo al punto in cui le leggi della Meccanica Quantistica avranno un effetto rilevante sul loro funzionamento (chi progetta i nuovi circuiti oggi è conscio di questo problema); inoltre molto più spesso le informazioni sono trasmesse tramite fotoni, ovvero codificate in raggi di luce ed inviate di solito in fibre ottiche, ma i fotoni sono particelle intrinsecamente quantistiche.

La Crittografia Quantistica può essere considerata come il primo risultato commerciale delle conseguenze di queste due osservazioni.

3.2 Gli Elaboratori Quantistici e la Sicurezza Informatica

Prima di buttarci nella Crittografia Quantistica e vedere come dai due punti appena indicati possa nascere qualche cosa di nuovo, facciamo il secondo passo indietro.

Nel 1982 Richard Feynman propose l'idea di un Elaboratore Quantistico che venne poi sviluppata nel 1985 nel fondamentale lavoro di David Deutsch che per primo descrisse dettagliatamente il suo funzionamento.[25] Precisiamo immediatamente che al giorno d'oggi non esiste alcun vero elaboratore quantistico funzionante e che le stime più ottimistiche dicono che ci vorranno almeno altri 20 anni prima di poterne costruire uno (ovviamente i pessimisti dicono che non sarà mai possibile).

Il più grande prototipo di elaboratore quantistico costruito sino ad oggi, è stato realizzato nei laboratori IBM del Almaden Research Center nel 2001. E' un elaboratore a 7 qubit (quantum-bit) e gli scienziati sono stati in grado di eseguire su di esso l'algoritmo di Shor, di cui parleremo più avanti, fattorizzando il numero 15 in 5 e 3.

24 Un esempio notevole di questo fu proprio Einstein che non si convinse mai completamente della correttezza della Meccanica Quantistica.

25 Si noti che gli elaboratori quantistici sono in realtà nati qualche anno dopo la crittografia quantistica, ma per semplicità di esposizione noi preferiamo invertire l'ordine storico.

L'idea di base di un elaboratore quantistico è quella di codificare i bit informatici in proprietà di particelle elementari quali gli elettroni o i fotoni. Una volta che i bit informatici sono codificati nelle particelle elementari, un elaboratore quantistico è in grado di eseguire delle trasformazioni sulle particelle secondo le leggi della meccanica quantistica. Facendo ciò, le proprietà delle particelle cambiano e questi cambiamenti possono essere letti come il risultato dell'operazione effettuata. In Appendice B diamo alcuni ulteriori dettagli sui principi di funzionamento di un elaboratore quantistico.

Gli elaboratori quantistici funzionano (o meglio funzioneranno) seguendo la logica delle leggi della Meccanica Quantistica e quindi sono (potenzialmente) in grado di fare i conti in modo molto diverso da quello noto a tutti noi. In particolare gli elaboratori quantistici saranno in grado di velocizzare alcuni calcoli ove è possibile in un *certo senso* parallelizzare l'esecuzione, più precisamente eseguire lo stesso calcolo su un grande numero di dati allo stesso tempo e automaticamente selezionare il risultato corretto. Non bisogna meravigliarsi se questo concetto così espresso non è molto chiaro, è certo colpa nostra ma al contempo, come abbiamo sottolineato più volte, è la logica stessa degli elaboratori quantistici che è diversa dalla logica a cui siamo abituati, e quindi il loro modo di procedere ci risulta sicuramente difficile da comprendere. In ogni caso, il modo con cui gli elaboratori quantistici faranno i conti sarà tale da permettere di risolvere alcuni difficili problemi matematici istantaneamente. Tra questi problemi vi sono quelli su cui si basano molti degli algoritmi crittografici moderni, quali ad esempio la fattorizzazione in numeri primi che è alla base di RSA.

Gli elaboratori quantistici sono macchine potenzialmente in grado di risolvere problemi *difficili* in tempi molto rapidi, in alcuni casi quasi istantaneamente in modo indipendente dalla lunghezza dell'input al problema. Ovviamente per poter risolvere dei problemi matematici così velocemente, oltre ad avere un vero elaboratore quantistico, è necessario avere un algoritmo di soluzione, non basta sapere che questo esiste. Però già oggi si conoscono due algoritmi ideati nel 1994 da Peter Shor per fattorizzare in numeri primi e risolvere i logaritmi discreti. Quindi per rompere quasi istantaneamente RSA, ElGamal, DSA ecc. non ci manca che costruire un vero elaboratore quantistico! In altre parole, se fosse possibile costruire oggi un elaboratore quantistico, questo sarebbe in grado quasi istantaneamente di ottenere da una chiave pubblica di qualunque lunghezza, la corrispondente chiave privata utilizzata dai comuni algoritmi Asimmetrici quali RSA.

Sorge naturale la domanda se gli elaboratori quantistici saranno in grado di rompere anche gli algoritmi simmetrici (e gli Hash) altrettanto rapidamente. La risposta ad oggi è data dall'algoritmo proposto da Lov Grover nel 1995 per effettuare ricerche in spazi non strutturati. Questo algoritmo permette di fare un attacco di forza bruta (equivalente a provare tutte le chiavi possibili) in un tempo che cresce come la radice del numero di chiavi da provare ($\sqrt{m}$) mentre gli algoritmi classici crescono come il numero di chiavi da provare (ovvero m). Visto che il numero di chiavi è dato da $m=2^n$ ove n è la lunghezza della chiave in bit, ne segue che per avere la stessa resistenza di un algoritmo simmetrico ad un attacco di un elaboratore quantistico rispetto a quello di un elaboratore tradizionale, dobbiamo usare chiavi lunghe il doppio, e questo in pratica potrebbe non essere un grave problema.

In ogni caso l'unico algoritmo crittografico noto oggi che non è attaccabile è il One-Time-Pad, contro il quale nulla potrà mai nessun elaboratore quantistico o tradizio-

nale.

Gli elaboratori quantistici sono sicuramente una spada di Damocle sulla crittografia moderna, ed il loro possibile prossimo avvento rivoluzionerà la crittografia. Se da un lato non vogliamo qui minimizzare i rischi per la crittografia moderna dovuti agli elaboratori quantistici, al contempo non vogliamo nascondere che ancora molto rimane da fare prima di poter costruire un vero elaboratore quantistico. Non solo vi sono difficili problemi tecnologici, ma vi sono anche problemi teorici ed in particolare il peggiore è legato al fenomeno della **de-coerenza**. Come abbiamo già indicato, un elaboratore quantistico può fare dei calcoli allo stesso tempo su moltissimi dati, banalizzando molto potremmo dire che il problema della de-coerenza è il fatto che l'elaboratore può essere indotto da fattori esterni a fare confusione tra i dati che sta trattando ed alla peggio addirittura dimenticarsi di cosa sta facendo, dando alla fine dei risultati assolutamente casuali.[26] Questo ovviamente sarebbe un totale disastro! Un altro problema degli elaboratori quantistici è quello dell'immagazzinamento dei dati ovvero della realizzazione di memorie (quantistiche) stabili e grandi a sufficienza per contenere tutti i dati necessari allo stesso tempo.

Dal punto di vista tecnologico, sono state proposte e realizzate nei laboratori di ricerca, varie tecniche per costruire un elaboratore quantistico. Tra queste citiamo quelle che sino ad oggi si sono dimostrate più promettenti:

- *Linear Ion Trap*: sono atomi con elettroni in eccesso od in difetto, mantenuti stabili con campi elettromagnetici, le operazioni sono realizzate con sottilissimi raggi laser;
- *Nuclear magnetic resonance (NMR)*: nuclei di molecole sono posti in forti campi magnetici e le operazioni sono realizzate utilizzando radiofrequenze che ne modificano il momento magnetico; il più grande prototipo di 7 qubit costruito nel 2001 nei laboratori IBM adottava questa tecnologia;
- *Quantum Dots*: è possibile creare delle piccole strutture all'interno di superconduttori nelle quali intrappolare singoli elettroni sui quali si può poi agire con campi elettrici o sottilissimi raggi laser.

Se gli elaboratori quantistici minacciano di far crollare la crittografia moderna, od almeno quella basata sugli algoritmi Asimmetrici, la crittografia quantistica si propone di salvare almeno una parte di essa offrendo un nuovo modo di generare e scambiare le chiavi segrete da usare ad esempio con l'OTP. Nella prossima sezione descriveremo quindi le basi fondamentali di funzionamento della Crittografia Quantistica.

3.3 La Crittografia Quantistica

Già negli anni '70 i fisici teorici si chiedevano se fosse possibile utilizzare le teorie che descrivono le particelle elementari, cioè la Meccanica Quantistica (e la Teoria dei Campi), per realizzare direttamente qualche cosa di veramente nuovo. In particolare la prima proposta di un protocollo di questo tipo fu formulata intorno al 1970

26 Questo è dovuto principalmente alla interferenza di altre particelle elementari, quelle dell'ambiente in cui è l'elaboratore quantistico, che potrebbero interagire anche contro la nostra volontà con le particelle utilizzate per il calcolo.

da Stephen Wiesner; si trattava di un protocollo per realizzare banconote quantistiche (il lavoro scientifico fu pubblicato solo nel 1983). Non vi furono molti sviluppi immediati, ma l'idea non fu mai del tutto abbandonata. Nel 1979 Bennett e Brassard ripresero questa idea e cominciarono a lavorare all'idea della Crittografia Quantistica, il loro primo lavoro su questo argomento fu pubblicato nel 1982. Cerchiamo di spiegare nella prossima sezione i principi di base su cui si fondano questi risultati.

3.3.1 I Principi Generali

Prima di tutto, il nome corretto della Crittografia Quantistica è **Quantum Key Distribution** (**QKD**), che possiamo riassumere come:

> *un protocollo per generare e scambiare in assoluta sicurezza delle chiavi segrete tra due corrispondenti per usi crittografici, mediante particelle elementari e sfruttando le leggi della meccanica quantistica.*

Quindi la QKD non è usata per proteggere le informazioni né per garantire confidenzialità, integrità o autenticità. La QKD offre un nuovo modo per risolvere il problema della generazione e distribuzione delle chiavi segrete. Come abbiamo ampiamente discusso, la crittografia moderna risolve questo problema con l'utilizzo degli algoritmi Asimmetrici (a chiavi Pubbliche), i quali però sono i più a rischio di un attacco da parte degli elaboratori quantistici e sono basati comunque su delle assunzioni matematiche ad oggi non provate.

Supponiamo di avere due corrispondenti, di solito nella letteratura scientifica chiamati Alice e Bob con Eve l'attaccante, che vogliono creare e scambiarsi delle chiavi segrete utilizzando QKD. Per fare ciò oggi hanno bisogno:

1. un canale di comunicazione Quantistico attraverso il quale si possono scambiare delle particelle elementari;
2. un canale di comunicazione Classico con il quale Alice e Bob possono comunicare in modo autenticato ed integro.

Al giorno d'oggi solo due canali Quantistici sono stati implementati:

1. *fibre ottiche*
2. *spazio libero.*

In questi canali le particelle elementari scambiate sono **Fotoni**. I fotoni sono le particelle elementari che costituiscono la luce ed in generale le onde elettromagnetiche. Le implementazioni pratiche di QKD oggi utilizzano fibre ottiche, quindi Alice e Bob devono avere a disposizione una fibra ottica che li connetta sulla quale inviare i fotoni. In futuro si potranno avere QKD sia con fotoni in spazio libero che con l'utilizzo di altre particelle quali gli elettroni.

Il canale Classico può essere anche una comunissima linea telefonica, ma avendo già a disposizione una fibra ottica di solito la si utilizza anche come canale classico. La cosa importante dal punto di vista di QKD è che il canale classico garantisca l'autenticità e l'integrità delle comunicazioni fra Alice e Bob.

Infatti possiamo riassume le richieste sui canali di comunicazione specificando cosa

Eve può fare su di essi:

1. Eve può fare tutto quello che la fisica le permette sul canale Quantistico
2. Eve può solo intercettare ma non modificare le comunicazioni sul canale Classico, ovvero può solo fare Eavesdropping.

Il protocollo ha la seguente struttura:

1. *Alice e Bob si scambiano fotoni preparati appositamente attraverso la fibra ottica che li connette*
2. *dopo di che si scambiano alcune informazioni sul canale di comunicazione classico*
3. *alla fine di questo processo Alice e Bob hanno generato e si sono scambiati una chiave segreta casuale; se Eve è stata in grado di intercettare la chiave segreta Alice e Bob ne sono sicuramente a conoscenza, interrompono le comunicazioni e si mettono alla ricerca di Eve.*

Il terzo punto è in realtà molto particolare se lo confrontiamo con le proprietà della crittografia classica. La crittografia classica non è in grado di rendersi conto se qualcuno può intercettare i messaggi, mentre quello che la crittografia quantistica fa è proprio di segnalare se qualcuno sta ascoltando sul canale di comunicazione. E' chiaro che non possiamo utilizzare QKD per cifrare e scambiare informazioni poiché il protocollo informa Alice e Bob della presenza di Eve solo dopo che la trasmissione dei dati è avvenuta! Se i dati trasmessi sono una chiave segreta non ancora utilizzata, non vi è un grande problema per la sicurezza delle informazioni, ma se invece di una chiave segreta non ancora utilizzata fossero scambiate delle informazioni sarebbero guai seri. Il ruolo fondamentale della meccanica quantistica è quello appunto di segnalare ad Alice e Bob qualunque tentativo di Eve di interferire nelle loro comunicazioni.

Una volta che Alice e Bob hanno generato e si sono scambiati una chiave segreta usando QKD, possono utilizzare il OTP per cifrare i dati e scambiarseli attraverso un qualunque canale di comunicazione non sicuro. L'accoppiata QKD ed OTP fornisce un sistema di comunicazioni assolutamente sicuro. In alternativa, la chiave prodotta da QKD può essere usata con algoritmi quali AES, il che permette sicuramente di aumentare la velocità di comunicazione a scapito ovviamente del livello di sicurezza.

Nella seguente tavola riassumiamo alcuni punti di paragone tra QKD e gli algoritmi Asimmetrici a chiavi Pubbliche/Private limitatamente al loro utilizzo per creare e scambiare chiavi segrete di sessione:

	QDK	***Public/Private Key***	
CON	Richiede HW e linee di comunicazione dedicate	Può essere implementato in software ed è molto portabile	**PRO**
PRO	Assolutamente sicuro basato su leggi fondamentali della fisica	Matematicamente non deciso, basato su problemi matematici per i quali non è nota pubblicamente una soluzione semplice (che potrebbe esistere)	**CON**
PRO	La sicurezza è basata su principi generali e non richiede alcuna modifica in futuro	La sicurezza è basata sull'uso di chiavi P/P sempre più lunghe a seconda della potenza degli elaboratori	**CON**
PRO	Sarà sicuro anche se vi saranno Elaboratori Quantistici	Gli Elaboratori Quantistici saranno in grado di romperlo facilmente	**CON**
CON	Oggi molto costoso	A disposizione di chiunque	**PRO**
CON	Molto nuovo ed in grande sviluppo	Ben sperimentato e di grande distribuzione	**PRO**
CON	Ad oggi funziona solo a distanze limitate e con connessioni dirette in fibra ottica	Funziona a qualunque distanza ed attraverso qualunque tipo di network	**PRO**
?	La velocità di creazione della chiave è ancora bassa, ma sta crescendo molto velocemente (problema tecnico)	Richiede parecchie risorse computazionali soprattutto con chiavi lunghe e dati lunghi, di solito questo non è un problema per lo scambio di chiavi segrete	**?**
PRO	Può essere usato facilmente con il One-Time-Pad, l'unico algoritmo di cifratura matematicamente sicuro	Non può essere usato semplicemente con il One-Time-Pad	**CON**

3.3.2 La Fisica di Base

Senza addentrarci qui nei principi fondamentali della meccanica quantistica ne tanto meno nei suoi dettagli tecnici, indichiamo alcune caratteristiche fondamentali delle leggi che governano le particelle elementari per dare una idea al lettore del perché e come possa funzionare la QKD. I concetti rilevanti sono

1. la Meccanica Quantistica è inerentemente *statistica*, perciò molto spesso non siamo in grado di predire con certezza il risultato di un esperimento ma solo di indicare la statistica dei risultati quando l'esperimento è ripetuto molte volte
2. non è possibile duplicare uno stato *sconosciuto*, in altre parole una fotocopiatrice quantistica non esiste
3. ogni misura *perturba* il sistema misurato (a meno che il sistema sia in uno stato particolare *compatibile* con la misura effettuata).

Queste proprietà sono sicuramente diverse dalle nostre esperienze quotidiane, ad esempio noi siamo in grado di osservare e misurare un oggetto a noi sino a quel momento sconosciuto senza modificarlo affatto, cosa invece impossibile in meccanica quantistica.

Nella implementazione pratica di QKD che descriveremo nella prossima sezioni, ci sarà utile una conseguenza di questi punti che possiamo enunciare come segue:

> *non è possibile misurare la polarizzazione di un fotone nella base verticale-orizzontale ed al contempo in quella diagonale.*

Visto che questa affermazione può risultare molto poco chiara al lettore, cerchiamo di chiarire il suo significato. La polarizzazione è una delle caratteristiche dei fotoni in cui può essere codificato il bit informatico. Ad esempio possiamo decidere che se un fotone ha polarizzazione verticale allora il valore codificato è 1, mentre se la polarizzazione è orizzontale il valore è 0. Analogamente se la polarizzazione è diagonale a destra il valore è 1 mentre se è diagonale a sinistra il valore è 0. Le polarizzazioni verticali ed orizzontali sono dette essere ortogonali fra di loro, analogamente le polarizzazioni diagonali destre e sinistre sono ortogonali fra loro. La meccanica quantistica ci dice che è possibile costruire uno strumento con il quale è possibile distinguere esattamente se una particella elementare è in uno di due stati ortogonali. Ma se una particella è in uno stato diverso da quelli per i quali abbiamo costruito o preparato lo strumento, il risultato della misura avrà una certa probabilità di dare un qualunque valore. Tornando ai nostri fotoni polarizzati, se prepariamo un filtro di polarizzazione per misurare fotoni polarizzati verticalmente/orizzontalmente e inviamo al filtro un fotone polarizzato verticalmente o orizzontalmente, sapremo esattamente quale è la polarizzazione del fotone. Se invece vi inviamo un fotone polarizzato diagonalmente, avremo il 50% di possibilità che il risultato della misura sia 1 ed il 50% che sia 0.

Il punto fondamentale di QKD è esattamente in questa proprietà della meccanica quantistica.

3.3.3 Il Protocollo BB84

Per mostrare come possiamo sfruttare queste particolari proprietà della Meccanica Quantistica per realizzare un sistema di QKD, illustriamo qui il protocollo più semplice, più famoso ed anche il primo ad essere enunciato. Il BB84 fu proposto da Bennett e Brassard appunto nel 1984 e malgrado l'anzianità rimane il protocollo di riferimento di QKD. E' anche il protocollo più semplice, e quindi più chiaro per esporre i principi fondamentali di funzionamento. Ovviamente molti altri protocolli sono stati proposti, e molte varianti del BB84 sono state introdotte. In realtà nelle implementazioni pratiche non viene di norma adottato il BB84 di base per vari motivi tecnici di cui non discuteremo in questa sede, ma il principio di funzionamento è esattamente lo stesso.

Il BB84 è composto da 3 fasi, la prima quantistica e le ultime due classiche. In questa sezione descriveremo brevemente la fase quantistica mentre nella sezione 3.3.5 descriveremo quelle classiche.

Il Protocollo. Alice e Bob eseguono le seguenti operazioni:

1. Usando il canale di comunicazione quantistico, Alice sceglie in modo *casuale* una delle 4 polarizzazioni, prepara un fotone con questa polarizzazione e lo invia a Bob, infine si annota la polarizzazione che ha scelto e la tiene rigorosamente segreta;
2. Bob sceglie in modo *casuale* (ed indipendente da Alice) di preparare il proprio filtro delle polarizzazioni o nella direzione verticale/orizzontale od in quella diagonale, misura il fotone ricevuto da Alice e si annota il risultato della misura e la direzione scelta per misurare tenendole entrambe segrete;
3. Alice e Bob ripetono i punti 1 e 2 un numero sufficiente di volte, questo numero dipende dalla lunghezza della chiave segreta che vogliono generare (di norma dovrebbe essere almeno più del doppio del numero di bit della chiave da generare);
4. Usando il canale di comunicazione classico, Bob informa Alice della scelta di direzione che ha usato per misurare ogni fotone ricevuto, ma non dice il risultato della misura;
5. Per ogni fotone Alice risponde a Bob dicendogli se la scelta di misura che Bob ha fatto era giusta o meno, ovvero se Bob aveva scelto la direzione corretta per misurare esattamente la polarizzazione; Alice e Bob scartano tutte le misure per cui Bob aveva scelto la direzione sbagliata, quella che dava un risultato casuale, ed i bit che rimangono formano la *sifted key*, ovvero la prima proposta per la chiave segreta.

Questo protocollo è descritto nella figura 1 e la sua configurazione sperimentale nella figura 2.

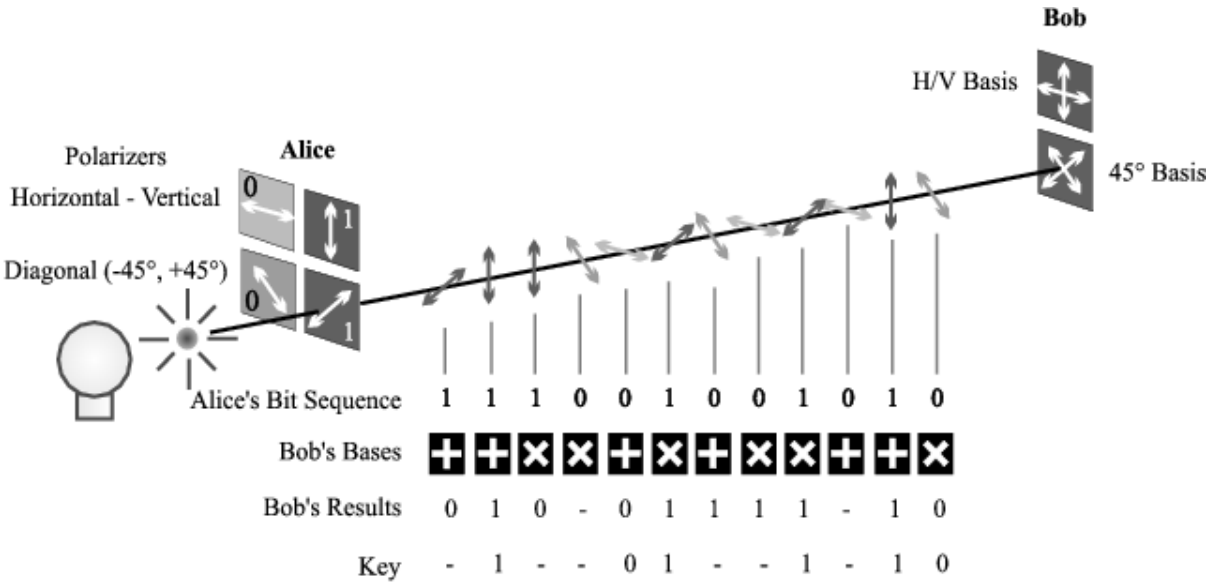


Figura 1. Il protocollo BB84

Descriviamo in breve questa figura. Alice invia i fotoni numero 1, 4, 6, 7, 9 e 12 (contando da destra a sinistra ovvero nell'ordine di invio di Alice) con polarizzazione nella base diagonale e gli altri nella base verticale-orizzontale. Bob sceglie casualmente la base di misura come indicato, quindi Bob usa la stessa base di Alice per i fotoni 1, 2, 3, 4, 7, 8, 9 e 11, mentre i risultati degli altri fotoni sono scartati nell'ultima fase del protocollo. Si noti ad esempio come Alice abbia inviato il fotone 10 con polarizzazione verticale, corrispondente al valore 1 del bit, ma Bob misurando nella base diagonale abbia ottenuto il valore 0. Inoltre per i fotoni numero 3 e 9 Bob non ha nessun risultato, questo vuol dire che vi è stato un errore sperimentale, ad esempio questi fotoni potrebbero essere stati persi lungo la fibra ottica (il problema degli errori sperimentali è discusso nella sezione 3.3.5). In conclusione, Alice e Bob formano una sifted-key di 6 bit corrispondente ai fotoni 1, 2, 4, 7, 8 e 11.

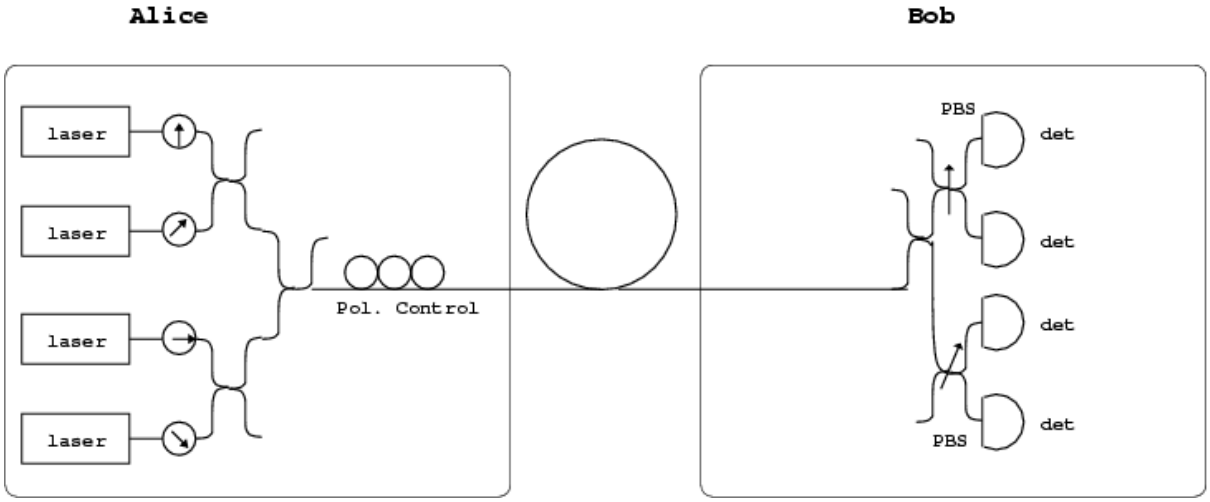


Figura 2. La configurazione sperimentale per il BB84

In questa figura è rappresentata in modo diagrammatico la configurazione sperimentale di un sistema QKD che realizza il protocollo BB84. Dal lato di Alice abbiamo una apparecchiatura realizzata con laser sulle usuali frequenze adottate nelle fibre di telecomunicazione, che produce un fotone con una delle quattro polarizzazioni possibili. Per semplicità abbiamo indicato 4 laser ognuno che produce una delle 4 polarizzazioni, ed un sistema di controllo delle polarizzazioni che sceglie in maniera casuale quale inviare a Bob. Ovviamente le realizzazioni pratiche sono molto più complesse di quanto indicato. Dal lato di Bob vi è prima di tutto un deviatore che in modo casuale invia il fotone ricevuto verso il rivelatore di polarizzazione verticale-orizzontale o quello diagonale. Il Polarization Beam Splitter (PBS) poi distingue tra le due possibili polarizzazioni nella propria base ed attiva il relativo rivelatore di fotoni (detector).

Si noti come dopo i punti 1 e 2 del protocollo, la chiave che ha ottenuto Bob ha in media il 25% di bit sbagliati, questo perché il 50% delle volte Bob sceglie la direzione sbagliata di misura e quando Bob sbaglia la scelta, il 50% di volte ottiene il risultato sbagliato, ovvero il bit sbagliato. Per questo, con i punti 4 e 5 Alice e Bob eliminano tutti i fotoni per i quali Bob ha scelto la direzione sbagliata, e questi sono in media la metà dei fotoni inviati.

Si noti inoltre come la chiave segreta è indipendente dalle scelte di Alice e Bob ma è una combinazione delle scelte casuali di entrambi.

Se tutto fosse perfetto a questo punto Alice e Bob avrebbero una chiave segreta, dobbiamo però considerare cosa può fare Eve e la possibilità che vi siano errori sperimentali e di trasmissione.

3.3.4 Eavesdropping

Supponiamo ora che Eve voglia intercettare i fotoni inviati da Alice a Bob. Vi sono molti possibili attacchi a disposizione di Eve, ma il più semplice è un attacco di tipo Man-in-the-Middle. In questo caso Eve riceve il fotone inviato da Alice, lo misura, ne prepara uno nuovo e lo invia a Bob. Anche se altri attacchi possono essere diversi, le caratteristiche principali sono alla fine simili a questo. Il punto fondamentale per la sicurezza è che Eve, come Bob, non sa quale è la polarizzazione scelta da Alice né quale direzione scegliere per fare una misura esatta. Quindi Eve fa degli errori esattamente come Bob. Infatti, come detto inizialmente, in meccanica quantistica è impossibile fare fotocopie, e quindi Eve non può fare una copia esatta del fotone che Alice ha inviato. Bisogna notare inoltre che gli errori fatti da Eve sono diversi ed indipendenti da quelli fatti da Bob. In conclusione, qualunque cosa Eve faccia, non conoscendo il fotone inviato da Alice, deve introdurre delle modifiche, deve disturbare il fotone in modo che Bob riceva dei fotoni diversi da quelli inviati da Alice.

Qualunque attacco faccia, Eve disturba e modifica i fotoni inviati da Alice.

La conseguenza delle interferenze che Eve fa, garantite dalla meccanica quantistica, è che vi saranno errori nella sifted-key in possesso di Bob; ad esempio nel caso del semplice attacco descritto, il 25% dei bit della sifted-key di Bob saranno diversi da quelli di Alice. Altri attacchi possono dare altre percentuali di errori, che però di solito non si discostano di molto dal 25%.

E' chiaro quindi che per capire se Eve ha effettuato un attacco alla trasmissione dei fotoni basta verificare se la sifted-key in possesso di Bob è diversa da quella di Alice. La seconda e la terza fase del protocollo BB84, descritte nella prossima sezione, affrontano anche questo problema.

3.3.5 Error Correction e Privacy Amplification

Sin qui non abbiamo considerato la possibilità che vi siano degli errori sperimentali, ovvero negli strumenti di creazione e misura dei fotoni, o degli errori di trasmissione per cui i fotoni siano modificati durante il tragitto per puri motivi accidentali. Come fare a distinguere questi errori da quelli introdotti da Eve ? La risposta è semplice: non si può.

Sembrerebbe quindi che siamo giunti ad un punto morto: vi sono sempre errori, ma se ci sono errori vuol dire che Eve ha intercettato la chiave poiché è praticamente impossibile distinguere con sicurezza tra errori sperimentali ed errori dovuti ad Eve.

La soluzione a questo apparente insolubile problema, è in realtà relativamente semplice. Prima di tutto si assume che tutti gli errori siano sempre dovuti a Eve. Poi Alice e Bob debbono applicare alla sifted-key due ulteriori fasi del protocollo. La prima si chiama "Reconciliation" od "Error Correction" e permette ad Alice e Bob di eliminare tutti gli errori nella sifted-key di Bob ed al contempo stimare la percentuale di errori trovati. Se questa percentuale è inferiore all'11%, allora si può passare alla fase seguente detta "Privacy Amplification". In questa fase la chiave segreta viene modificata secondo una procedura tale che l'informazione che nel caso Eve ha sulla chiave segreta viene ridotta praticamente a zero. Questo è possibile perché se Eve ha in-

trodotto errori solo per al più l'11%, vuol dire che la sua conoscenza della sifted-key è sufficientemente ridotta, conosce pochi bit della chiave, e quindi modificando appropriatamente la chiave segreta Alice e Bob possono eliminare i bit a conoscenza di Eve.

Queste ultime due fasi possono essere realizzate anche pubblicamente poiché le informazioni scambiate tra Alice e Bob non aiutano Eve a fare lo stesso. Bisogna inoltre notare che in queste due fasi la lunghezza della chiave viene ulteriormente ridotta. A seconda delle procedure utilizzate la chiave segreta finale può anche essere lunga solo 1/8 del numero di fotoni inizialmente inviato da Alice.

Indichiamo ora come sia possibile realizzare in pratica queste due fasi. Gli algoritmi che presentiamo qui sono molto banali ed altamente inefficienti, ma forniscono l'idea di come funzionano le procedure indicate. Per realizzare la Error Correction ed al contempo valutare la percentuale di errori presenti nella sifted-key, Alice e Bob possono procedere come segue. Alice sceglie 2 bit della sua chiave a caso e ne calcola l'XOR. Alice dice a Bob quali bit ha scelto e il valore dell'XOR ma non il valore dei singoli bit. Bob calcola l'XOR sugli stessi bit della sua sifted-key e dice ad Alice se il risultato è lo stesso o no. Se il risultato è lo stesso, Alice e Bob tengono il primo bit e scartano il secondo, altrimenti scartano entrambi i bit e registrano un errore.

Il lettore attento si sarà immediatamente reso conto che questa procedura non funziona correttamente quando entrambi i bit di Bob sono errati, in questo caso l'XOR valutato da Bob ha lo stesso valore di quello di Alice, e quindi l'errore non è segnalato. Ovviamente il protocollo indicato è troppo semplice e deve essere sostituito con altri più avanzati. Spesso nei sistemi QKD viene adottato un protocollo proposto da Brassard e Salvail nel 1993. Questo è un protocollo di correzione degli errori iterativo che, sfruttando il teorema di Coding di Shannon, riduce al minimo l'informazione scambiata tra Alice e Bob ed al contempo permette di trovare praticamente tutti gli errori nella sifted key.

Visto che Eve può ascoltare sul canale classico, anche Eve può applicare il protocollo di correzione degli errori, ma se gli errori introdotti da Eve sono meno dell'11% vuol dire che la sifted-key di Eve è diversa da quella di Bob e da quella di Alice, ed alla fine della fase di Error Correction la sifted-key di Eve sarà ancora più diversa da quella di Alice che è ora uguale a quella di Bob.

Dopo la fase di Error Correction, Eve può essere ancora a conoscenza di alcuni bit della chiave. Alice e Bob possono procedere come segue per ridurre l'informazione a conoscenza di Eve. Di nuovo Alice sceglie a caso 2 bit e ne calcola l'XOR. Alice informa Bob dei 2 bit scelti (ma non del loro valore) e Bob calcola anch'esso l'XOR. Sia Alice che Bob cancellano i due bit e li sostituiscono con il valore dell'XOR. Alice e Bob ripetono un numero sufficiente di volte questa procedura. Vediamo cosa succede quando anche Eve esegue questa procedura. Visto che la sifted-key di Eve è diversa da quella di Alice e Bob, qualche volta Eve farà l'XOR di un bit giusto con uno sbagliato, e li sostituirà con l'XOR che ovviamente è sbagliato. In conclusione il numero di errori nella sifted-key di Eve aumenta, sino a che praticamente tutti i bit sono sbagliati. Anche in questo caso, vi sono protocolli più efficienti e teoricamente corretti rispetto a quello appena indicato che vengono di norma implementati nei si-

stemi QKD.

3.4 Problemi di Gioventù

Questa breve introduzione potrebbe aver lasciato dei dubbi sulla vera sicurezza della crittografia quantistica, ed ovviamente posto il problema della convenienza economica della sua implementazione. Per quanto riguarda i costi, e quindi la valutazione di chi può essere interessato a questa tecnologia oggi, possiamo dire che governi, militari ed agenzie di sicurezza, banche ed istituzioni finanziarie hanno espresso interesse a valutare QKD. Ad esempio, Visa International, l'azienda internazionale di carte di credito, sta sperimentando questa tecnologia, ed altre banche e istituzioni finanziare hanno annunciato il loro interesse. Quello che ovviamente va valutato è il rapporto tra il costo per implementare un sistema di comunicazioni basato su QKD ed il livello di sicurezza che si ritiene essere necessario per i propri scopi. Bisogna poi ovviamente che tutto il sistema sia sicuro allo stesso livello, e non solo la generazione delle chiavi!

Per quanto riguarda invece la sicurezza delle implementazioni di QKD che sono oggi sul mercato, vi sono ancora alcuni punti da approfondire e migliorare. Da un punto di vista formale e teorico il protocollo è sicuro in modo assoluto grazie alle leggi della meccanica quantistica. Vi sono però vari aspetti che vanno sottolineati e che diventano cruciali nelle implementazioni pratiche.

Ad esempio sino ad oggi tutti i protocolli teorici richiedono la presenza di un canale classico autenticato ed integro ma non specificano come questo possa essere realizzato. Ovviamente possiamo utilizzare algoritmi classici, e questi sono stati studiati ed implementati. Quello che si ottiene alla fine è in pratica un protocollo di allungamento delle chiavi segrete. In altre parole, Alice e Bob al momento della configurazione iniziale del sistema di QKD si scambiano delle chiavi segrete che poi adoperano per mettere in sicurezza il canale classico di comunicazione. Una volta che hanno attivato QKD possono utilizzare parte delle chiavi segrete generate con questo per sostituire le chiavi iniziali e garantire la sicurezza del canale classico. Ovviamente questa soluzione non è bella né da un punto di estetica teorica né per l'analisi della sicurezza in quanto il punto debole diventa proprio il canale classico di comunicazione.

Vi sono poi tutti i problemi tecnici dovuti al fatto di dover produrre e rilevare singoli fotoni che ovviamente contribuiscono agli errori sperimentali. Anche se la tecnica ha fatto dei passi avanti incredibili negli ultimi anni, gli apparecchi sono ancora molto poco affidabili se confrontati con quelli che usiamo di solito in informatica. Non solo, ad oggi QKD è implementabile solo su pezzi singoli di fibre ottiche senza alcun ripetitore, ed infatti il record di distanza (nell'anno 2004) è di 150 chilometri. Sono già in sviluppo dei ripetitori quantistici che fra qualche anno, ma non meno di 5 o 10, permetteranno di allungare senza problemi le distanze. Un'altra possibilità è la trasmissione in aria dei fotoni, vari esperimenti sono già stati fatti dimostrando la possibile fattibilità di questa tecnologia. Lo scopo finale è quello di inviare i fotoni su satelliti dai quali sarebbero ri-inviati a terra riuscendo in questo modo a coprire tutta la superficie terreste. Ovviamente anche in questo caso dobbiamo attendere ancora qualche anno.

I risultati ottenuti dalla crittografia quantistica, ed il fatto che siano già sul mercato i primi modelli, ha ovviamente interessato molto i ricercatori ed i finanziatori. Ora la ricerca si sta ampliando a molti altri interessanti argomenti, ad esempio se è possibile utilizzare la meccanica quantistica per realizzare protocolli di cifratura, o firme digitali ecc. A titolo puramente di esempio possiamo citare gli studi in corso sulla possibilità di generare dei sigilli quantistici su informazioni codificate utilizzando particelle elementari, come un tempo si faceva apponendo i sigilli di ceralacca sulle buste. Chiaramente ancora molto può essere fatto sia per la implementazione di QKD che per lo sviluppo di tecniche simili. Basti dire che l'Unione Europea ha appena finanziato il progetto SECOQC, iniziato il 1 Aprile 2004, per lo sviluppo sia della ricerca che della implementazione tecnologica e commerciale della Crittografia Quantistica. Il progetto ha un budget di 11,4 Milioni di Euro in 4 anni, vi partecipano 41 partner in 12 paesi europei, e per l'Italia vi sono l'Università` di Pavia, il CNR, la Scuola Normale Superiore di Pisa ed il Politecnico di Milano.

Per concludere possiamo cercare di guardare ancora più avanti. Come abbiamo già detto, gli elaboratori diventano sempre più piccoli e più veloci, e prima o poi sarà possibile realizzare circuiti in cui fluiscono direttamente ad esempio fotoni (già oggi si lavora alla realizzazione di switch completamente ottici). Non è detto che questi debbano essere per forza elaboratori quantistici, ma sicuramente potrebbero interagire direttamente con la crittografia quantistica e altri simili protocolli basati sulla meccanica quantistica.

Appendice A: Breve introduzione a OpenPGP

In questa appendice indichiamo brevemente come è costruito un protocollo molto noto. Lo scopo di questa appendice è di dare una idea più precisa al lettore di come non sia semplice mettere insieme gli algoritmi crittografici per realizzare un protocollo utilizzabile direttamente per una applicazione pratica. In questa sezione ci limitiamo comunque ad una descrizione sommaria di OpenPGP, rimandando il lettore più interessato ai relativi documenti tecnici quali gli RFC-2440 e RFC-3156, ma anche RFC-1991 e RFC-2015. Va notato che PGP ha avuto una evoluzione abbastanza tormentata durante la quale sono apparse varie versioni anche abbastanza differenti fra di loro nei dettagli di funzionamento. In questa appendice faremo riferimento in particolare al RFC-2440 intitolato *OpenPGP Message Format.*

PGP (Pretty Good Privacy) ha come scopo di fornire confidenzialità, autenticità ed integrità ai messaggi di posta elettronica utilizzando lo schema indicato nella sezione 2.3.1. In particolare OpenPGP permette di apporre ad un messaggio una firma digitale, che garantisce autenticità ed integrità, e/o di cifrare il messaggio. Le due operazioni possono essere fatte entrambe o l'utente può scegliere di farne una soltanto. Vediamo come possiamo in pratica realizzare questo protocollo. Per completezza ripetiamo la procedura di creazione di un messaggio firmato e cifrato:

1. dato il messaggio per prima cosa se ne calcola l'Hash
2. l'Hash viene cifrato con un algoritmo Asimmetrico e la chiave Privata del mittente
3. l'Hash cifrato viene pre/post-messo al messaggio ottenendo così un messaggio+firma-digitale
4. viene generato un numero casuale da usarsi come chiave segreta di sessione per l'algoritmo simmetrico
5. si cifra il messaggio+firma-digitale con l'algoritmo simmetrico e la chiave casuale segreta di sessione
6. la chiave casuale segreta di sessione viene cifrata con un algoritmo Asimmetrico e la chiave Pubblica del ricevente
7. la chiave di sessione così cifrata viene premessa al messaggio+firma-digitale cifrato
8. il tutto viene inviato al corrispondente.

OpenPGP deve quindi usare almeno un algoritmo Asimmetrico, un algoritmo simmetrico ed un Hash. L'algoritmo Asimmetrico e simmetrico hanno delle chiavi segrete. Come abbiamo già visto, la chiave segreta di sessione dell'algoritmo simmetrico è un numero casuale generato direttamente dal programma ed usato una volta sola. Invece per l'algoritmo Asimmetrico abbiamo una chiave Privata del mittente e le chiavi Pubbliche del mittente e del ricevente. La chiave Privata del mittente deve essere tenuta per lungo tempo e protetta, mentre per cifrare le chiavi di sessione[27] è

27 Ed anche per verificare le firme digitali dei messaggi ricevuti.

necessario ottenere e memorizzare le chiavi Pubbliche dei destinatari. Per memorizzare e gestire le chiavi, OpenPGP utilizza due *keyring* (portachiavi), in pratica due file in uno dei quali vi sono le chiavi Pubbliche e nell'altro quelle Private. Ma le chiavi Private devono essere protette, e per questo vengono cifrate con un algoritmo simmetrico ed una chiave questa volta decisa dall'utente. Quando un programma basato su OpenPGP ha necessità di accedere alla chiave Privata dell'utente, chiede all'utente la chiave segreta con cui la chiave Privata è cifrata, decifra la chiave Privata e la utilizza.

Ma non è così semplice. L'utente non è in grado di solito di generare una chiave segreta con le proprietà corrette (numero di bit, casualità ecc.) per essere usata come chiave per l'algoritmo simmetrico usato per cifrare la chiave Privata. Per questo l'utente fornisce invece una *Passphrase* di lunghezza arbitraria. Utilizzando una procedura denominata *String-to-key* (S2K), la Passphrase è trasformata nella chiave utilizzata per cifrare la chiave Privata. In pratica nel keyring delle chiavi Private, prima di ogni chiave Privata vi è un header che specifica la procedura S2K da applicare alla passphrase fornita dall'utente. In particolare questo header contiene il codice che indica l'algoritmo di Hash da applicare alla passphrase, un numero casuale detto *salt* da anteporre alla passphrase per prevenire attacchi di dizionario, il numero di volte che la procedura di Hash deve essere ripetuta per ottenere la chiave con cui è cifrata la chiave Privata. La procedura S2K in pratica consiste nel prendere l'Hash del salt+passphrase una o più volte, e poi selezionare gli n bit più importanti (leftmost, OpenPGP è big-endian) se la chiave da ottenere è di n bit. Nel caso in cui la lunghezza dell'Hash sia inferiore alla lunghezza della chiave da ottenere, la procedura viene ripetuta premettendo al salt+passphrase un ottetto di zeri, ed il nuovo risultato viene aggiunto a destra a quello precedente. Se questo non fosse ancora sufficiente, viene premesso un altro ottetto di zeri e si ripete ancora la procedura sino a che non si ottiene una chiave della lunghezza richiesta.

Nel keyring delle chiavi Private, dopo l'header S2K, segue la chiave Privata cifrata con relativo header. In realtà dopo l'header è presente un Initial Vector (IV) e poi la chiave Privata cifrata. La chiave Privata è cifrata con l'algoritmo scelto e l'IV indicato, utilizzando l'algoritmo simmetrico in Cipher Feedback Mode (CFB).[28] Segue a ciò una semplice checksum a 16bit della chiave Privata, e questa checksum è cifrata anch'essa come la chiave Privata. La checksum permette di verificare che la passphrase data dall'utente è corretta con l'usuale procedura: si decifrano sia la chiave Privata che la checksum e si ricalcola la checksum sulla chiave Privata decifrata, se le due checksum sono uguali vuol dire che la passphrase fornita dall'utente è corretta.

Lo scopo principale di tutta questa procedura è quello di proteggere la chiave Privata principalmente da attacchi di dizionario, ovvero provare molte semplici passphrase.

Da un punto di vista storico, può essere interessante anche indicare come questa procedura è stata modificata nel tempo. Inizialmente la passphrase era trasformata nella chiave di cifratura usando MD5 senza alcun salt, e poi la chiave Privata era decifrata usando questa chiave segreta e l'algoritmo simmetrico IDEA. La prima osservazione su questa procedura, è che la scelta di usare algoritmi fissi è sicuramente problematica. Infatti come abbiamo visto, nel tempo gli algoritmi devono essere

28 Il Cipher Feedback Mode (CFB) e l'Output Feedback Mode (OFB) sono due ulteriori modi con cui utilizzare cifrari a blocchi, essi permettono di realizzare algoritmi di tipo Stream a partire da algoritmi a blocchi.

cambiati, per motivi di sicurezza come nel caso di MD5, o di licenze come per IDEA. Per risolvere questo problema, nella struttura dei dati di OpenPGP sono stati quindi introdotti gli header che specificano gli algoritmi utilizzati. Dal punto di vista della sicurezza, l'assenza di salt nella procedura di generazione della chiave segreta rendeva molto più facili gli attacchi di dizionario. Ad esempio, la stessa passphrase utilizzata per più chiavi in assenza di salt generava sempre la stessa chiave segreta. Utilizzando invece il salt, la stessa passphrase anche se riutilizzata genera differenti chiavi segrete. Quindi la prima modifica del protocollo fu quella di anteporre il salt alla passphrase. Successivamente ci si rese conto che è possibile rendere ancora più difficili gli attacchi di dizionario se oltre ad aggiungere il salt, si ripete varie volte l'operazione di Hash. Così è stata introdotta una terza versione della procedura S2K, che è quella attualmente in vigore.

Il lettore a questo punto sarà un po' preoccupato del fatto che in una pagina siamo riusciti solo a descrivere sommariamente come le chiavi Private sono scritte nel loro keyring. Abbiamo voluto con questo indicare al lettore come non sia per nulla facile passare da un algoritmo crittografico ad un protocollo implementato in un programma che effettivamente garantisca la sicurezza che ci aspettiamo. Moltissimi aspetti che possiamo dire pratici, quali quello della gestione delle chiavi Private appena descritto, devono essere risolti nella maniera opportuna per poter soddisfare le richieste di sicurezza. Detto ciò, passiamo ora a descrivere la parte principale di OpenPGP in modo un po' più riassuntivo.

Prima di tutto, qualunque dato o messaggio di PGP è organizzato in blocchi detti *Pacchetti,* un Pacchetto è formato da un header che come primo dato ha un Tag (un numero) che indica il tipo di Pacchetto, e dopo l'header seguono i dati specifici di quel pacchetto. Esistono molti tipi di Pacchetti, ad esempio:

Tag 1: Publick-key encrypted session key

Tag 2: Signature

Tag 3: Symmetric-key encrypted session key

Tag 4: One-pass signature

Tag 5: Secret key

Tag 6: Public key

Tag 7: Secret subkey

Tag 8: Compressed data

Tag 9: Symmetrically encrypted data

e così via. Un Pacchetto può essere incluso all'interno di un altro Pacchetto. Ad esempio, prima di cifrare un messaggio, OpenPGP lo comprime utilizzando di solito ZIP, questo anche per motivi di sicurezza e non solo di lunghezza dei messaggi in quanto un messaggio compresso è più *casuale* rispetto alla sua versione normale e rende più difficili molti attacchi. Quindi un programma che implementa OpenPGP prima crea un Pacchetto di tipo 8 contenente il messaggio compresso, e poi crea un Pacchetto di tipo 9 nel quale è cifrato il Pacchetto precedente.

Un'altra nota di interesse è che le chiavi Pubbliche/Private sono identificate attraverso un key-ID di 64bit oppure una fingerprint. La fingerprint è data dal Hash SHA-1 di 160 bit dell'intero Pacchetto della chiave Pubblica più alcuni header, mentre le key-ID sono i 64 bit di destra (low-order) della fingerprint. La key-ID e la fingerprint sono usati per identificare velocemente la chiave Pubblica o Privata da utilizzare sia da parte del programma che degli utenti, anche se ovviamente più chiavi potrebbero avere lo stesso key-ID (più difficilmente la stessa fingerprint).

Passiamo ora a descrivere brevemente il procedimento di creazione di un messaggio di posta elettronica firmato e cifrato, indicando come i vari Pacchetti sono costruiti. Dato il messaggio che assumiamo per semplicità in UTF-8, per prima cosa vi apponiamo una firma digitale. Per questo creiamo un pacchetto di tipo 2.[29] Questo Pacchetto ha un header contenente tra altre le seguenti informazioni:

- la key-ID della chiave Privata con la quale è fatta la firma digitale
- l'algoritmo Asimmetrico utilizzato
- la data di creazione della firma
- l'algoritmo di Hash utilizzato
- i primi due ottetti di sinistra (high-order) dell'Hash (non cifrato).

La firma è realizzata calcolando per prima cosa l'Hash del messaggio con l'algoritmo indicato, all'Hash viene poi concatenata la versione del tipo di firma e la data di creazione della firma e tutto questo è cifrato con l'algoritmo Asimmetrico e la chiave Privata indicata (la chiave Privata è prelevata dal keyring e decifrata con la procedura descritta precedentemente). La presenza dei primi 2 ottetti di sinistra dell'Hash non cifrato nell'header del Pacchetto permette di verificare velocemente se il Pacchetto e la procedura di verifica della firma, ad esempio la chiave Pubblica utilizzata, sono validi. Ad esempio, il ricevente del messaggio firmato vuole verificare la firma utilizzando la chiave Pubblica con la key-ID indicata. Con questa chiave Pubblica decifra l'Hash e per prima cosa verifica che i primi due ottetti, la data di creazione e la versione sono uguali a quelli presenti nell'header; in caso negativo la firma non è valida ancora prima di confrontare l'Hash appena decifrato con quello calcolato indipendentemente dal ricevente del messaggio.

Prima di proseguire dobbiamo notare che la firma digitale così realizzata può essere connessa al relativo messaggio in modi diversi, ad esempio utilizzando MIME, od in un ASCII Armor, o creando un *PGP Signed Message*. Nei vari casi la posizione ed il modo in cui la firma è legata al messaggio sono differenti. Nel caso che stiamo illustrando formiamo un PGP Signed Message, che non è altro che la concatenazione del Pacchetto di tipo 2 appena creato con un Pacchetto di tipo 11 (Literal Data) che contiene null'altro che il messaggio originale. A questo punto il PGP Signed Message (Pacchetto 2 + Pacchetto 11) viene compresso formando un Pacchetto di tipo 8. Questo Pacchetto ha un formato molto semplice, l'Header contiene l'algoritmo di compressione, di norma ZIP, ed il Body contiene i dati compressi.

Il passo successivo è generare in maniera opportuna un numero casuale con le proprietà necessarie per essere una buona chiave segreta di sessione per l'algoritmo

29 Per semplicità descriviamo la versione 3 del Tag 2; la versione 4 differisce principalmente in una maggiore elasticità ed ampiezza del formato del Pacchetto distribuendo molte informazioni in sotto-Pacchetti.

simmetrico scelto. Non entriamo in questa sede nella discussione di questo difficile problema e delle tecniche adottate per risolverlo. Si forma poi un Pacchetto di tipo 9 (Symmetrically Encrypted Data) nel quale l'header non contiene altro che il valore del Tag. Infatti l'algoritmo usato per cifrare e le altre informazioni necessarie per la decifrazione sono incluse nel pacchetto di tipo 1 (o 3) che deve obbligatoriamente precedere ogni Pacchetto di tipo 9. I dati sono cifrati con la chiave di sessione appena generata e l'algoritmo simmetrico scelto con una modalità CFB modificata apposta per OpenPGP. In questa modalità l'IV è nullo (tutti gli ottetti sono zero) ma ai dati viene preposto un blocco contenente un numero casuale. Gli ultimi due ottetti del numero casuale sono ripetuti anche all'inizio del secondo blocco (dopo di che avviene un riallineamento del CFB). La ripetizione dei due ultimi ottetti del numero casuale permette in decifrazione di verificare che la chiave usata sia quella corretta senza dover neanche decifrare il messaggio vero e proprio.

Il passo successivo è creare un Pacchetto di tipo 1 (Public-Key Encrypted Session Key) da preporre al pacchetto precedente. Questo Pacchetto contiene la key-ID della chiave Pubblica usata, l'algoritmo Asimmetrico utilizzato e la chiave segreta di sessione cifrata. Prima di essere cifrata, alla chiave di sessione viene preposto il codice dell'algoritmo simmetrico utilizzato per cifrare il Pacchetto di tipo 9 che segue e che abbiamo appena descritto, e viene postposta una checksum di 16 bit della stessa chiave di sessione. La presenza della checksum è per una verifica veloce di errori o dell'uso di una chiave Privata sbagliata nella decifrazione della chiave di sessione. La procedura di cifratura con l'algoritmo Asimmetrico e la chiave Pubblica indicata, ad esempio la divisione in blocchi, il padding dell'ultimo blocco ecc., segue lo standard PKCS-1.[30]

La concatenazione degli ultimi due Pacchetti descritti, di tipo 1 e tipo 9, forma il messaggio OpenPGP firmato e cifrato finale, a meno di un ultimissimo passo, la conversione RADIX-64 (detta anche ASCII Armor). Il problema è che il messaggio così costruito è formato da uno stream di ottetti binari puri, e spesso non è possibile trasportare questo tipo di dati con i normali programmi di comunicazione poiché alcuni ottetti binari potrebbero essere interpretati come caratteri di controllo da parte dei programmi di trasmissione. Questo è sicuramente il caso di smtp ad esempio, e pertanto è necessario trasformare lo stream di ottetti binari in uno stream di ottetti formato solo da 64 caratteri ASCII stampabili, a costo di un aumento di 1/3 della lunghezza del messaggio.

30 Lo standard Public-Key Cryptography Standard 1 (PKCS1), disponibile su http://www.rsasecurity.-com/rsalabs/node.asp?id=2125, descrive le procedure fondamentali per l'utilizzo pratico di RSA.

Appendice B: Principi di funzionamento di un elaboratore quantistico

In questa sezione daremo alcune ulteriori indicazioni su come dovrebbe funzionare un elaboratore quantistico. Ricordiamo che ad oggi sono stati realizzati solo piccoli prototipi sperimentali di elaboratori quantistici, e gli scienziati non sono assolutamente sicuri di riuscire a costruire veri elaboratori quantistici, almeno nei prossimi anni. Vogliamo qui dare al lettore una idea di quali siano le basi logiche utilizzate da un elaboratore quantistico e mostrare come questo sia molto diverso da un grande elaboratore parallelo. Non vogliamo qui fare una introduzione alla meccanica quantistica, ed il lettore più interessato ma che non intende fare un corso di meccanica quantistica universitario, può ad esempio leggere le lezioni in ref. [Mermin] per alcune utili informazioni al proposito.

Cominciamo con il rappresentare il processo di elaborazione di dati in un modo abbastanza generale. Supponiamo di avere dei dati in input, che possiamo genericamente considerare essere un numero, che viene inviato dentro una unità di elaborazione, appunto il nostro elaboratore. All'interno di questa scatola, vengono eseguite delle trasformazioni sul dato in ingresso, una dopo l'altra. Alla fine di questo procedimento la scatola produce in output il risultato in forma di un numero.

Un elaboratore classico agisce su numeri scritti in forma binaria, ed ogni operazione effettuata al suo interno può essere scomposta in operazioni effettuate sui singoli bit. Poiché sui dati effettuiamo una successione di operazioni, tra una operazione e l'altra i dati avranno un certo valore. Possiamo pensare ad un ciclo della CPU come indicatore della successione delle operazioni effettuate. Diciamo in generale che tra una operazione e l'altra, il nostro elaboratore è in uno *STATO* descritto, tra gli altri parametri, dal valore del dato in elaborazione. Ogni bit in uno stato ha il valore 0 od 1. Indichiamo per convenzione lo stato di un bit con il simbolo |0> se esso ha valore 0, e |1> se esso ha valore 1.[31]

In un elaboratore classico ogni singolo bit può essere nello stato |0> o |1>, e niente altro. Ad esempio non ha senso che l'elaboratore abbia un bit nello *stato* |1/2>, e sarebbe un disastro se il nostro PC decidesse di usare la logica ternaria tutto di un colpo.[32]

Vogliamo sottolineare la differenza fra il concetto di stato e quello di valore del bit. Lo *stato* è ad esempio la *configurazione fisica* in cui si trova un circuito, e descriviamo questo con il simbolo |0> o |1>. Il *valore* di un bit associato ad uno stato è invece un numero che per convenzione associamo ad una certa configurazione fisica. In un elaboratore classico, tra una operazione e l'altra ogni bit è o nello stato |0> o nello stato |1>, ed ha valore rispettivamente 0 o 1.

Analogamente in un elaboratore quantistico il valore del bit è codificato nello stato, una proprietà fisica, di una particella elementare. Come nell'esempio della crittogra-

31 Queste notazioni provengono dalla meccanica quantistica.

32 Ovviamente c'è un momento di transizione da uno stato all'altro in cui i potenziali elettrici dei circuiti possono avere un valore intermedio tra quello interpretato come 0 e quello interpretato come 1, ma questo non è uno *stato*, è un momento brevissimo di una transizione.

fia quantistica, si usano particelle elementari con proprietà che formano due stati ortogonali ai quali si assegnano per convenzione il valore 0 e 1. Di solito questi stati sono chiamati qubit (Quantum bit) ed indicati come prima con |0> e |1>.

Dopo tutta questa noiosa premessa, giungiamo al punto fondamentale. All'interno di un elaboratore quantistico lo stato di un qubit può generalmente essere rappresentato nella forma:

$$a\,|0\rangle + b\,|1\rangle$$

ove $|a|^2 + |b|^2 = 1$ con a e b numeri complessi. Quindi se a=1 o b=1 siamo in uno stato quale quelli presenti negli elaboratori classici, ma altrimenti siamo davanti a qualche cosa di completamente nuovo.

Non ha molto senso chiedersi quale sia il significato numerico di tale stato, poiché è qualche cosa che esiste solo in meccanica quantistica, ovvero solo a livello di particelle elementari e non ha un senso definito nella fisica macroscopica, quella a cui i nostri sensi e la nostra ragione sono abituati.[33]

Cosa succede quindi se all'interno di un elaboratore quantistico possiamo fare trasformazioni su stati di questo tipo? Se cerchiamo di interpretare una operazione su di uno stato di questo tipo dal punto di vista del valore numerico ad esso associato, potremmo dire che siamo in grado di operare contemporaneamente sia sul valore 0 che sul valore 1 del qubit. In una maniera *molto* particolare siamo in grado di fare un calcolo parallelo, trasformando al contempo entrambi i valori di un singolo bit! La cosa sembra a prima vista abbastanza paradossale, ma come abbiamo ripetuto più volte il punto fondamentale della meccanica quantistica è proprio quello di adottare regole di ragionamento del tutto diverse da quelle a cui siamo abituati. Per questo dobbiamo accettare, alle volte senza capire bene quello che sta succedendo, e vedere cosa possiamo fare seguendo queste logiche.

Notiamo inoltre che un elaboratore quantistico, come uno classico, agirà di solito sul prodotto tensoriale di n qubit, in termini classici un numero a n bit. Pertanto il grado di *parallelismo*, se possiamo chiamarlo così, di un elaboratore quantistico potrebbe idealmente corrispondere ad eseguire una singola istruzione contemporaneamente su 2^n stati classici. In realtà non è proprio così, poiché le regole di calcolo non sono identiche a quelle di un corrispondente ipotetico enorme elaboratore classico parallelo.

Per capire in pochino meglio la situazione, limitiamoci a considerare uno o due qubit. Quello che è possibile fare, è disegnare delle trasformazioni (quantum gate) come su di un elaboratore classico, che agiscono sui qubit nella forma appena indicata. Come in un elaboratore classico esistono delle operazioni elementari che agiscono su singoli bit o coppie di bit come AND, NOT, OR, XOR ecc., anche per gli elaboratori quantistici sono stati disegnati dei circuiti quantistici, in termini della meccanica quantistica questi sono chiamati delle *trasformazioni unitarie nello spazio degli stati dei qubit*. E' importante sottolineare che in pratica una trasformazione quantistica modifica per ogni qubit i coefficienti a e b mantenendo la relazione $|a|^2 + |b|^2 = 1$.[34]

33 Vi è una discussione preminentemente filosofica sul fatto che si possa dire che questi stati *esistono* o sono solo una rappresentazione formale della realtà subatomica a nostro uso; in ogni caso i fisici li hanno usati con incredibile successo e senza alcuna controindicazione negli ultimi 50 anni.

34 Ogni operazione è una trasformazione in U(2) e può essere rappresentata come una matrice 2x2

Usando queste regole è possibile ideare degli algoritmi per effettuare dei calcoli su numeri rappresentati in qubit. In questo spazio limitato non possiamo addentrarci nella descrizione di un algoritmo quantistico, basti dire che una volta accettate le regole della meccanica quantistica, il processo di generazione di un algoritmo quantistico non è molto diverso dal caso classico. Vogliamo inoltre ricordare che alcuni di questi algoritmi sono stati già implementati sui prototipi di elaboratori quantistici costruiti.

Ci importa però qui sottolineare un altro aspetto fondamentale di un elaboratore quantistico che lo distingue da un elaboratore classico parallelo. Abbiamo detto che un elaboratore quantistico a n-qubit agisce contemporaneamente in qualche senso su quelli che sarebbero 2^n stati classici. Se questo fosse un vero elaboratore parallelo ci aspetteremmo di poter fare 2^n calcoli parallelamente ed ottenere 2^n risultati. Invece un elaboratore quantistico non funziona così. Consideriamo un attimo in pratica come funzionerebbe un elaboratore quantistico. Come prima cosa dobbiamo preparare gli n-qubit del numero in input, ovvero avere ad esempio n particelle elementari preparate ognuna nello stato che rappresenta un bit del numero iniziale dell'elaborazione. Gli n-qubit vengono poi trasformati dall'elaboratore quantistico secondo i modi che abbiamo indicato, ed alla fine otteniamo in output di nuovo n-qubit in un particolare stato. Questo vuol dire in pratica che in uscita da un elaboratore quantistico avremo n particelle elementari. A questo punto dobbiamo fare qualche cosa per leggere lo stato in cui sono queste particelle e ricavarne il numero risultato. Il lettore più attento e che ha già letto la sezione sulla crittografia quantistica, avrà indovinato cosa succede. Per sapere il risultato del calcolo bisogna fare una misura sullo stato quantistico risultante, ma facendo la misura su di uno stato ignoto si modifica lo stato stesso. Questo però non conduce ad un disastro, invece quello che succede è che la modifica dello stato è tale che se si provasse a fare una seconda misura sullo stato si otterrebbe sempre lo stesso valore. In altre parole, dagli n-qubit in output da un elaboratore quantistico possiamo estrarre con una misura quantistica solo 1 risultato numerico, un solo numero di n bit. Il *parallelismo*, se così possiamo chiamarlo, di un elaboratore quantistico è solo una proprietà interna alla elaborazione che ci permette di disegnare algoritmi molto efficienti per risolvere alcuni problemi, ma che non aiuta per nulla in altri casi, ad esempio se vogliamo ripetere lo stesso calcolo su n input diversi, ovvero fare un calcolo classicamente parallelo.

ad elementi complessi che agisce sul vettore (a b); nel caso di n qubit, l'operazione è un elemento di $U(2^n)$.

Bibliografia Essenziale

Alcuni testi generali sulla crittografia sia di introduzione tecnica che storica:

[HAC] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997 (*HAC*)

[Schneier] B. Schneier, *Applied Cryptography*, John Wiley & Sons, New York 1997

[Welsh] D. Welsh, *Codes and Cryptography,* Oxford Science Pub., New York, 1988

[Cover] T.M. Cover, J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 1991

[Kahn] D. Kahn, *The Codebreakers*, McMillan Pub., New York, 1967

[Meltzer] C. Meltzer, D. Baker, *Cryptography Decrypted*, Addison-Wesley, 2001

Riferimenti ai principali standard citati nel testo:

[3DES] 3DES FIPS: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[AES-1] AES home page: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

[AES-2] AES NIST: http://csrc.nist.gov/CryptoToolkit/aes/

[AES-3] AES FIPS: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[AES-4] AES WiKi:http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[AES-5] Attacchi a AES: http://www.cryptosystem.net/aes/

[AES-6] Breve introduzione a AES on-line: http://home.ecn.ab.ca/~ jsavard/crypto/co040401.htm

[RSA] R. River, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM 21 (1978) 120

[MD5] R. Rivest, *The MD5 Message-Digest Algorithm*, RFC-1321

[SHA] SHA-1 e varianti FIPS: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

[Shannon] I due principali lavori scientifici di Shannon sulla Informazione e la crittografia:
http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
http://www.cs.ucla.edu/~jkong/research/security/shannon1949.pdf

Alcuni testi introduttivi e storici in Italiano:

[Singh] Simon Singh, *Codici e segreti*, Rizzoli, Milano, 1999

[Berardi] L. Berardi, A. Beutelspacher, *Crittologia*, FrancoAngeli, Milano, 1996

[Sgarro] A. Sgarro, *Crittografia*, Franco Muzzio Editore, Padova, 1993

[Zanotti] M. Zanotti, *Crittografia*, Hoepli Editore, Milano, 1976

Alcuni riferimenti sulla crittografia quantistica e gli elaboratori quantistici:

[UCCI] Alcuni White Papers ed altre informazioni sulla Crittografia Quantistica sono disponibili sul sito http://www.ucci.it/

[SECOQC] Il sito ufficiale del progetto SECOQC dell'Unione Europea è http://www.secoqc.net/

[QUBIT] Il sito http://www.qubit.org/ è dedicato alla ricerca di base nei campi degli elaboratori quantistici e della crittografia quantistica

[Mermin] Una introduzione alla fisica quantistica per informatici per avvicinarsi ai computer quantistici è in David Mermin, *From Cbits to Qbits*, http://lanl.arxiv.org/abs/quant-ph/0207118

[BEZ] Una introduzione alla fisica della *Informazione Quantistica* che presuppone alcune conoscenze di base di meccanica quantistica è in D. Bouwmeester, A. Ekert, A. Zeilinger, *The Physics of Quantum Information*, Springer-Verlag 2000

[Preskill] John Preskill tiene un corso per fisici ed informatici su *Quantum Computation*, informazioni, links e lectures-notes (400 pagine) sono sul sito http://www.theory.caltech.edu/~preskill/ph219/index.html

[Gisin] Una introduzione molto tecnica alla crittografia quantistica è in N. Gisin, G. Ribordy, W. Tittel, H. Zbinden, *Quantum Cryptography*, *Reviews of Modern Physics,* Vol. 74, p. 145 (2002), http://arXiv.org/abs/quant-ph/0101098

[OnLine] Altri riferimenti ed introduzioni alla Crittografia Quantistica on-line si trovano ad esempio in http://www.csa.com/hottopics/crypt/overview.html e http://jfi.uchicago.edu/~pelton/reading.html

CLUSIT Associazione Italiana per la Sicurezza Informatica
Dipartimento di Informatica e Comunicazione
Università degli Studi di Milano
Via Comelico 39 - 20135 MILANO

GRAFICA DI COPERTINA: Giorgio Giudice