This is the last preprint. The final paper appeared in "Matematica, Arte, Tecnologia, Cinema", Bologna 2000, ed. da M.Emmer e M.Manaresi, Springer Verlag Italia, 2002, 99-106.

CRITTOGRAFIA E FIRMA DIGITALE

A. LANGUASCO A. PERELLI

La sicurezza nella trasmissione dell'informazione è una necessità da sempre sentita dall'umanità. Nel corso dei secoli sono state utilizzate varie idee per questo scopo; esempi famosi sono il metodo di trasposizione di Giulio Cesare e la macchina di cifratura Enigma (utilizzata dalle forze armate tedesche durante la seconda guerra mondiale). I tentativi fatti precedentemente agli anni '70 non furono completamente soddisfacenti (si ricordi ad esempio la storia della forzatura del codice Enigma operata dal matematico A. Turing e dal suo gruppo negli anni '40); la possibilità di costruire un sistema crittografico rispondente a requisiti di sicurezza e autenticità fu provata a livello teorico da Diffie e Hellman nel 1976, mediante un rivoluzionario metodo detto a chiave pubblica. Tale idea trovò applicazione pratica due anni dopo, quando Rivest, Shamir e Adleman, utilizzando le proprietà dei numeri primi, concretizzarono l'idea di Diffie e Hellman. La crittografia moderna nasce quindi negli anni '70, e soppianta quasi completamente i metodi ottenuti come evoluzione delle idee classiche.

1. Crittografia

La crittografia studia i metodi che possono essere usati per inviare informazioni in forma celata, in modo tale che solamente il destinatario autorizzato possa rimuovere l'impedimento e leggere il messaggio in forma chiara. Denotando con

$$\mathcal{M} = \{\text{messaggi in chiaro}\} \quad \text{e} \quad \mathcal{C} = \{\text{messaggi cifrati}\}$$

gli insiemi dei messaggi in forma chiara e in forma celata, una *trasformazione crittografica* è una funzione iniettiva $f : \mathcal{M} \to \mathcal{C}$. È quindi definita la funzione inversa f^{-1}, il cui dominio è $f(\mathcal{M})$:

$$\mathcal{M} \xrightarrow{f} f(\mathcal{M}) \xrightarrow{f^{-1}} \mathcal{M}.$$

Si noti che è importante supporre l'iniettività di f perchè vogliamo evitare ogni possibile ambiguità nella decifrazione dei messaggi. Chiameremo

$$f : \text{ funzione di } \textit{cifratura} \quad \text{e} \quad f^{-1} : \text{ funzione di } \textit{decifratura}.$$

Un *crittosistema* è costituito da una quaterna $(\mathcal{M}, \mathcal{C}, f, f^{-1})$.

Una breve storia.

Uno dei primi metodi crittografici fu usato da Giulio Cesare. Con terminologia moderna, esso era basato sull'aritmetica modulo n (dove n è il numero di lettere contenute nell'alfabeto utilizzato). Fissata una corrispondenza bigettiva tra l'alfabeto e $\mathbb{Z}_n$, la cifratura veniva realizzata utilizzando uno "spostamento" delle lettere di una fissata quantità m; ad esempio, se $n = 21$ e $m = 5$, per cifrare il messaggio M (che supponiamo formato da una sola lettera) veniva utilizzata la funzione di cifratura $C \equiv M + 5 \,(\mathrm{mod}\ 21)$ e la funzione di decifratura $M \equiv C - 5 \,(\mathrm{mod}\ 21)$. È chiaro che un'accurata analisi della frequenza con cui ricorrono le lettere nel messaggio consente di violare il sistema (cioè consente di ottenere M a partire da C, ricostruendo la funzione di decifratura).

Nel sedicesimo secolo Vigenère sviluppò una variazione di tale metodo, che è risultata più difficile da violare. Si considerino blocchi di k lettere (ogni parola è identificata con un vettore di $(\mathbb{Z}_n)^k$) e si definisca, come funzione di cifratura, lo "spostamento" contemporaneo di tutte le lettere del blocco mediante una "parola-chiave" di k lettere (ovvero si aggiunge al messaggio $M \in (\mathbb{Z}_n)^k$ un fissato vettore $m \in (\mathbb{Z}_n)^k$, addizionandolo componente per componente). Come nel caso precedente, anche questo sistema può essere violato per mezzo di un'opportuna analisi di frequenza.

Nel 1931 Hill utilizzò come funzione di cifratura f il prodotto per una matrice invertibile. Posto $\mathcal{M} = \mathcal{C} = (\mathbb{Z}_n)^k$, egli considerò una matrice invertibile A i cui coefficienti stanno in $\mathbb{Z}_n$. La funzione di cifratura si ottiene allora moltiplicando A per $M \in \mathcal{M}$, mentre quella di decifratura è ottenuta mediante il prodotto di $C \in \mathcal{C}$ per A^{-1}. Il metodo di Hill può essere violato utilizzando metodi di algebra lineare modulo n.

Negli anni '20 un imprenditore tedesco, A. Scherbius, sviluppò la prima versione di una macchina di cifratura che sarebbe poi divenuta famosa con il nome di Enigma. Essenzialmente, il metodo di cifratura era basato su un'ingegnosa iterazione del metodo di Giulio Cesare. Ogni singolo spostamento era realizzato mediante un cilindro che collegava in modo elettromeccanico la lettera in input con quella in output; l'output di tale cilindro veniva poi passato come input ad un successivo cilindro che realizzava un secondo spostamento, e cosí via fino al risultato finale. Le prime versioni di Enigma erano dotate di tre cilindri collegati in sequenza, mentre in seguito vennero introdotte ulteriori complicazioni aumentando i cilindri a cinque, consentendone di variare la posizione di partenza ed anche la sequenza in cui essi operavano. Il governo tedesco del tempo fu talmente convinto della inviolabilità di Enigma che su di essa basò tutto il traffico di informazioni riservate, sia di tipo militare che commerciale. I punti deboli di Enigma vennero intuiti dal matematico polacco M. Rejewski che, svolgendo i calcoli a mano, riuscí per un certo tempo a decifrare i dispacci tedeschi. In seguito, i servizi segreti britannici istituirono un ufficio cifratura guidato dal matematico A. Turing; sebbene nel frattempo la complessità di Enigma fosse stata aumentata introducendo le suddette varianti, il gruppo di Turing riuscí a svilup-

pare alcuni calcolatori elettromeccanici (progenitori degli odierni computers) con i quali analizzare e, in molti casi, decifrare i messaggi di Enigma.

In definitiva, i primi crittosistemi utilizzavano solamente concetti elementari di algebra e teoria dei numeri; essenzialmente, tale situazione perdurò fino agli anni '70.

Crittografia classica.

Gli esempi fino ad ora trattati appartengono alla categoria della *crittografia classica* (detta anche *a chiave segreta* oppure *simmetrica*). Chiunque abbia abbastanza informazioni per cifrare un messaggio può, eventualmente con un piccolo sforzo, decifrarlo. Inoltre, gli utenti che desiderano comunicare confidenzialmente devono accordarsi sulle chiavi di cifratura e decifratura, e scambiarsele tramite un canale sicuro. Osserviamo che i metodi classici possiedono la seguente caratteristica: la *cifratura e la decifratura sono computazionalmente equivalenti* (ossia le loro complessità computazionali sono uguali o dello stesso ordine). Tale caratteristica consente di operare una classificazione basata sulla complessità computazionale.

Crittografia classica: ad essa appartengono i crittosistemi in cui (nota la chiave di cifratura) la funzione di decifratura può essere implementata con una complessità computazionale equivalente a quella della cifratura.

Alla *crittografia a chiave pubblica* appartengono i crittosistemi in cui la funzione di decifratura f^{-1} presenta una complessità computazionale di ordine maggiore rispetto a quella di f.

Nei prossimi paragrafi svilupperemo il concetto di crittografia a chiave pubblica e illustreremo alcuni esempi.

Crittografia a chiave pubblica.

La crittografia a chiave pubblica (detta anche *asimmetrica*) fu introdotta da Diffie e Hellman [6] nel 1976. Per chiarire la definizione data nel paragrafo precedente, osserviamo che in un crittosistema a chiave pubblica chi conosce solamente come cifrare non è in grado di usare la chiave di cifratura per ricostruire quella di decifratura senza compiere calcoli di lunghezza proibitiva. In altre parole, $f : \mathcal{M} \to \mathcal{C}$ può essere calcolata con facilità conoscendo K_E (chiave di cifratura), ma calcolare $f^{-1} : f(\mathcal{M}) \to \mathcal{M}$ senza avere informazioni addizionali (ossia senza conoscere la chiave di decifratura K_D) è un problema computazionalmente molto difficile. Le funzioni f di questo tipo vengono dette *one-way functions* (o *trap-door functions*).

Allo stato attuale non si conoscono funzioni di cui sia stato *provato* l'essere one-way (anche se esistono funzioni, usate nella pratica, che si congettura essere one-way). Notiamo inoltre che, a causa dello sviluppo delle prestazioni dei microprocessori, funzioni che oggi possono essere considerate one-way potrebbero perdere tale caratteristica nel prossimo futuro.

Osserviamo infine come un sistema a chiave pubblica possa essere usato come metodo sicuro per lo scambio delle chiavi di un sistema classico. Ciò può essere utile perchè generalmente i sistemi classici utilizzano chiavi più semplici di quelle dei sistemi a chiave pubblica, consentendo quindi un risparmio di tempo.

Il crittosistema R.S.A.

Uno dei più importanti esempi di funzione congetturalmente one-way (ed anche il primo crittosistema a chiave pubblica ad essere sviluppato) fu proposto da Rivest, Shamir e Adleman [2] nel 1978. La loro idea fu quella di sfruttare la forte differenza (congetturale) tra la complessità computazionale degli algoritmi di primalità e quella degli algoritmi di fattorizzazione.

Il sistema R.S.A. si può descrivere nel seguente modo: ogni utente deve

- scegliere in modo casuale due numeri primi grandi p, q (attualmente 300 cifre decimali sono considerate una scelta sicura) e calcolare $n = pq$;
- calcolare $\varphi(n) = (p-1)(q-1) = \#\mathbb{Z}_n^*$;
- scegliere in modo casuale un intero e, $1 \leq e \leq \varphi(n)$, coprimo con $\varphi(n)$;
- calcolare $d \equiv e^{-1} \pmod{\varphi(n)}$.

Osserviamo che il fatto che le suddette scelte siano *casuali* è un requisito fondamentale per ottenere un crittosistema sicuro. Osserviamo inoltre che tutti i calcoli coinvolti hanno bassa complessità computazionale. Per ogni utente X del crittosistema sono quindi definite le seguenti quantità:

chiave pubblica $K_E = (n, e)$: deve essere nota a tutti gli utenti del sistema, in modo che ognuno possa codificare i messaggi da inviare a X;

chiave privata $K_D = d$: è la chiave privata di X e deve essere tenuta segreta poichè essa consente a chi la conosce di decifrare i messaggi inviati a X;

funzione di cifratura: $f : \mathbb{Z}_n \to \mathbb{Z}_n$ data da $f(M) \equiv M^e \pmod{n}$;

funzione di decifratura: $f^{-1} : \mathbb{Z}_n \to \mathbb{Z}_n$ data da $f^{-1}(C) \equiv C^d \pmod{n}$.

Un utente A che desideri inviare un messaggio M all'utente B utilizza la chiave pubblica (n_B, e_B) di B per calcolare

$$C \equiv M^{e_B} \pmod{n_B}$$

e invia C a B. Ricevuto C, B usa la propria chiave segreta d_B e calcola

$$C^{d_B} \pmod{n_B} \equiv M^{e_B d_B} \pmod{n_B};$$

poichè $e_B d_B \equiv 1 \pmod{\varphi(n_B)}$, il teorema di Fermat-Eulero assicura (in questo caso) che

$$C^{d_B} \pmod{n_B} \equiv M \pmod{n_B}$$

e quindi B può leggere il messaggio M in chiaro.

Osserviamo che per violare il sistema R.S.A. (ovvero per calcolare f^{-1} senza conoscere K_D) bisogna essere in grado di calcolare d conoscendo solo e e n; noto e, per calcolare d è necessario conoscere $\varphi(n)$; ma, noto n, calcolare $\varphi(n)$ è *computazionalmente equivalente* a fattorizzare n. Quindi per violare il sistema R.S.A. occorre essere in grado

di fattorizzare n efficientemente (ovvero in un tempo ragionevole); ma, come vedremo nel paragrafo seguente, attualmente la complessità computazionale degli algoritmi di primalità (per provare la primalità di p e q) è decisamente più bassa di quella degli algoritmi di fattorizzazione (per calcolare, a partire da n, i fattori primi p e q). Per questo motivo il sistema R.S.A. è, in pratica, considerato sicuro.

Cenni sugli algoritmi di primalità e fattorizzazione.

Al momento della scelta di $n = pq$ nel sistema R.S.A. è necessario evitare scelte di p e q di "forma speciale" (ad esempio, classiche forme speciali sono del tipo $2^k - 1$ o $2^k + 1$); in tali casi esistono infatti algoritmi di fattorizzazione molto veloci. Esaminiamo ora alcune caratteristiche degli *algoritmi* (o *test*) *di primalità*, ovvero degli algoritmi atti a determinare se un dato intero n è primo. Ricordiamo preliminarmente che il numero di cifre decimali di un intero n è circa $\log n$, che la quantità $\log n$ viene presa come "unità di misura" per la complessità computazionale degli algoritmi riguardanti il numero n e che un algoritmo è *polinomiale* se la sua complessità è $O(\log^c n)$ per qualche $c > 0$.

Alcuni tra i più noti test di primalità sono basati su proprietà delle congruenze, ed in particolare sul Piccolo Teorema di Fermat: se n è primo allora $a^{n-1} \equiv 1 \pmod{n}$ per ogni $a \in \mathbb{Z}_n^*$. Osserviamo però che tale teorema non caratterizza i numeri primi; esistono infatti i *numeri di Carmichael*, ovvero gli interi *compositi* n che soddisfano $a^{n-1} \equiv 1 \pmod{n}$ per ogni $a \in \mathbb{Z}_n^*$. Ad esempio, $561 = 3 * 11 * 17$ è un numero di Carmichael, e nel 1994 è stato dimostrato da Alford, Granville e Pomerance [3] che tali numeri sono infiniti. Pertanto, i test più semplici sono in grado di fornire solamente una stima *probabilistica* della primalità di n (ovvero: n è primo con una certa probabilità, molto vicina a 1), anche se l'algoritmo di Miller-Rabin (vedi [8]) può provare la primalità di n con complessità computazionale $O(\log^5 n)$. Tale test assume però la validità dell'Ipotesi di Riemann generalizzata, una famosa congettura in teoria dei numeri. Senza assumere alcuna ipotesi, il più veloce test di primalità oggi noto è stato introdotto nel 1983 da Adleman, Pomerance e Rumely [1] ed ha una complessità computazionale (provata con tecniche di teoria analitica dei numeri)

$$O((\log n)^{c \log\log\log n}),$$

dove $c > 0$ è una certa costante; in altre parole, tale test è "quasi-polinomiale".

Consideriamo ora gli *algoritmi di fattorizzazione*. Negli ultimi vent'anni molti studiosi hanno tentato di risolvere il problema di fattorizzare gli interi in modo veloce ma, sebbene siano stati tentati anche approcci basati su tecniche molto sofisticate, allo stato attuale questo problema sembra essere intrinsecamente più difficile di quello della primalità. Infatti, anche il migliore algoritmo di fattorizzazione attualmente noto (Pollard [11] ebbe l'idea base nel 1993, ed altri ricercatori successivamente la migliorarono; si veda il libro di Lenstra-Lenstra [10]) non ha una complessità computazionale comparabile con quella dei test di primalità. Tale metodo è basato sulle proprietà dei campi di numeri algebrici, ed

ha complessità computazionale (congetturale)

$$O(e^{c\sqrt[3]{\log n(\log\log n)^2}})$$

dove $c > 0$ è una data costante, che è notevolmente maggiore di quella dei test di primalità.

Come esempio pratico, notiamo che un computer reperibile in commercio con poca spesa può costruire un intero (prodotto di due primi di forma "generale") di 140 cifre decimali in pochi secondi. Per fattorizzare lo stesso numero, usando computer paralleli, è necessario circa un mese di tempo! È quindi buona norma usare numeri sufficientemente grandi; attualmente un intero (prodotto di due primi di forma generale) di 220 cifre decimali è considerato una scelta sicura per un uso personale, ma per scopi professionali è consigliato l'utilizzo di interi ancora più grandi. Osserviamo infine che, poichè l'esistenza di minorazioni sufficientemente grandi per la complessità computazionale degli algoritmi di fattorizzazione è un problema tuttora aperto, il crittosistema R.S.A. può essere considerato un sistema a chiave pubblica solamente da un punto di vista congetturale, anche se viene comunemente utilizzato nalla pratica.

2. La firma digitale

Nella vita di tutti i giorni accade frequentemente di dover firmare un documento, un titolo di credito oppure di dover autenticare una fotocopia. In tutti questi casi è richiesta l'apposizione della propria firma a testimonianza della propria identità e, in alcuni casi, per assicurare la non ripudiabilità delle obbligazioni sottoscritte (si pensi ad un assegno bancario). La nostra identità viene quindi certificata attraverso la nostra grafia, che viene ritenuta un elemento identificativo della persona. Tutto ciò può essere realizzato solamente su un supporto cartaceo e quindi si pone il problema di come creare un concetto equivalente ed altrettanto efficace per supporti immateriali quali sono i bit. A tale scopo si è pensato di introdurre il concetto di firma digitale per autenticare una qualunque sequenza di cifre binarie, indipendentemente dal loro significato. Nella ricerca di un metodo che fornisca una firma digitale con i requisiti di cui sopra, è chiaro che la crittografia classica non è di aiuto; dobbiamo quindi rivolgerci alla crittografia a chiave pubblica, che permette di ottenere un semplice algoritmo di firma digitale descrivibile con notazioni generali nel modo seguente.

Siano A e B due utenti di un sistema a chiave pubblica. Sappiamo allora che le funzioni di cifratura f_A e f_B sono pubbliche e che le corrispondenti funzioni di decifratura f_A^{-1} e f_B^{-1} sono segrete; assumiamo inoltre che f_A e f_B siano funzioni surgettive (oltre che iniettive). Se A desidera inviare un messaggio M a B, deve inviare il messaggio cifrato $f_B(M)$ e, per certificare la propria identità, basta che egli invii anche la quantità $f_B(f_A^{-1}(nome_A))$. $f_A^{-1}(nome_A)$ rappresenta la firma digitale di A e $nome_A$ un nome convenzionale di A. B decifra il messaggio usando f_B^{-1} (che solo lui conosce) e ottiene $f_B^{-1}(f_B(M)) = M$. Per controllare che il mittente sia proprio A, B controlla l'identificazione allegata al messaggio applicando $f_A f_B^{-1}$; egli ottiene cosí $f_A f_B^{-1}(f_B f_A^{-1}(nome_A)) = nome_A$. Il sistema funziona

bene perchè solamente A può aver firmato in tale modo il messaggio (perchè nessun altro conosce f_A^{-1}).

Resta però il problema di essere veramente sicuri che sia proprio A a possedere f_A^{-1} e che invece non sia C che, rendendo pubblica una chiave, la abbia attribuita ad A divenendo automaticamente capace di spacciarsi per lui. Ciò può essere ovviato introducendo una figura super-partes: un *Ente Certificatore* delle chiavi pubbliche a cui ogni utente può rivolgersi per confrontare se la firma ottenuta tramite il processo sopra descritto effettivamente coincide con quella certificata (e quindi controllata) e depositata all'Ente Certificatore.

Inoltre, è sempre possibile che un intruso riesca ad identificare $f_A^{-1}(nome_A)$ utilizzando un numero opportuno di messaggi intercettati. Una strategia può allora essere quella di variare la posizione della firma digitale all'interno del messaggio inviato; al momento della decifratura il destinatario si ritroverà con una parte del messaggio priva di senso e capirà che è in quel punto che si trova la firma digitale del mittente. Un'idea sicuramente migliore è quella di far dipendere la firma digitale dal messaggio stesso; la firma digitale diviene allora $f_A^{-1}(impronta_M)$, dove $impronta_M$ è una sequenza di bit di lunghezza fissata (usualmente 160 cifre binarie) che viene ottenuta da M mediante un'opportuna funzione che ha la caratteristica di non consentire di risalire a M conoscendo solamente $impronta_M$, e di avere una buona probabilità di non generare collisioni (sono generalmente accettate funzioni che assicurino che la probabilità che $impronta_M = impronta_{M'}$ con $M \neq M'$ sia minore di 10^{-50}). Tali funzioni sono denominate *funzioni hash* e vengono messe a disposizione degli utenti dall'Ente Certificatore; ne viene utilizzata una sola per tutti gli utenti del crittosistema. B controllerà che il messaggio sia mandato da A e otterrà $impronta_M$; per accertarsi della probabile assenza di manomissioni, B ricalcolerà poi l'impronta di M per mezzo della funzione hash del crittosistema.

Chiaramente ciò introduce una certa debolezza in un metodo che altrimenti parrebbe inattaccabile; perchè allora non si utilizza come impronta del messaggio il messaggio stesso? Certamente questo assicurerebbe completamente l'identità del mittente ed anche la totale integrità del messaggio, ma ha lo svantaggio pratico di richiedere un più lungo tempo di cifratura e decifratura. Come sempre in questi casi, bisogna scegliere tra la sicurezza assoluta a un maggior costo in tempo e una sicurezza relativa (ma comunque alta) a un minor costo in tempo. Tale scelta dipenderà essenzialmente dal tipo di informazioni contenute in M, ed anche da fattori personali.

Esistono anche altre forme di firma digitale, che privilegiano ancora di più il fattore tempo rispetto alla sicurezza. Per esempio, alcuni metodi di firma digitale utilizzano $f_A^{-1}(impronta_M)$ allegata al messaggio M inviato in chiaro. Altri metodi assicurano solamente la provenienza del mittente, ma non l'integrità del messaggio (si firma con $f_A^{-1}(nome_A)$ allegato al messaggio M inviato in chiaro).

Osserviamo infine che l'introduzione dell'Ente Certificatore consente di apporre una *marcatura temporale* al messaggio stesso; tale possibilità è importante poichè è un fatto

risaputo che le date indicate da un computer non sono significative. Nel caso in cui A abbia la necessità di datare un messaggio, invia all'Ente Certificatore $f_B(f_A^{-1}(impronta_M))$ (da cui non è possibile risalire al testo in chiaro) a cui, dopo aver aggiunto la data, l'Ente Certificatore applica la propria chiave privata. Fatto questo, la quantità cosí marcata viene inviata ad A, che la allega al messaggio M e spedisce il tutto al destinatario B cifrando secondo le regole della crittografia a chiave pubblica. In un certo senso, la marcatura temporale è analoga all'autenticazione di un documento e serve a certificare che esso non possa essere in seguito sostituito con uno diverso.

Bibliografia

Per quanto riguarda la teoria dei numeri segnaliamo l'eccellente testo di Davenport [5] (in lingua italiana), che contiene anche un capitolo sulla crittografia. Per maggiori dettagli sugli aspetti storici dello sviluppo della crittografia si consiglia il libro di Kahn [7] ed il recente libro di Singh [13]. Eccellenti referenze sulla modellizzazione matematica della crittografia a chiave pubblica sono i testi di Koblitz [8] e [9]. Per quanto concerne gli algoritmi di fattorizzazione e di primalità, si consigliano i libri di Koblitz [8], di Cohen [4] e di Riesel [12].

[1] L.M. Adleman, C. Pomerance, R. Rumely - *On distinguishing prime numbers from composite numbers* - Annals of Math. **117** (1983), 173-206.

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

[3] W.R. Alford, A. Granville , C. Pomerance - *There are infinitely many Carmichael numbers* - Ann. Math. **140** (1994), 703-722.

[4] H. Cohen - *A Course in Computational Algebraic Number Theory* - Springer 1994.

[5] H. Davenport - *Aritmetica Superiore* - Zanichelli 1994.

[6] W. Diffie, M.E. Hellman - *New directions in cryptography* - IEEE Trans. Information Th. **22** (1976), 644-654.

[7] D. Kahn - *The Codebreakers, the Story of Secret Writing* - Macmillan 1967.

[8] N. Koblitz - *A Course in Number Theory and Cryptography* - Springer 1987.

[9] N. Koblitz - *Algebraic Aspects of Cryptography* - Springer 1998.

[10] A.K. Lenstra, H.W. Lenstra (eds) - *The development of the Number Field Sieve* - Springer L.N. 1993.

[11] J.M. Pollard - *Factoring with cubic integers* - in "The development of the Number Field Sieve", A.K. Lenstra and H.W. Lenstra (eds), Springer L.N. 1993, 4-10.

[12] H. Riesel - *Prime Numbers and Computer Method for factorization* - Birkhäuser 1994.

[13] S. Singh - *Codici & segreti* - Rizzoli 1999.

Alessandro Languasco
Dipartimento di Matematica Pura e Applicata
Università di Padova
Via Belzoni 7
35131 Padova, Italy
e-mail: languasco@math.unipd.it

Alberto Perelli
Dipartimento di Matematica
Università di Genova
Via Dodecaneso 35
16146 Genova, Italy
e-mail: perelli@dima.unige.it