Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

CRITTOGRAFIA E RETICOLI GEOMETRICI

Tesi di Laurea in Crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Matteo Candita

II Sessione
A.A. 2012/2013

Indice

Prefazione

Già dall'antichità l'uomo si è reso conto, specialmente in ambito politico e militare, dell'importanza di riuscire a comunicare segretamente: da Giulio Cesare, con l'utilizzo dell'omonimo cifrario, fino a Enigma, la sofisticata macchina di cifratura utilizzata dai tedeschi durante la seconda guerra mondiale, i metodi crittografici si sono evoluti per cercare di dare una risposta a questa esigenza. Con il recente sviluppo dell'informatica e delle telecomunicazioni il problema della crittografia è ritornato più che mai alla ribalta: oltre alla questione della riservatezza nella cifratura del messaggio, bisogna infatti rispondere anche a esigenze di autenticazione, verifica di integrità dei dati e non ripudiabilità del messaggio.
Affronteremo qui la posizione del problema crittografico, daremo una panoramica su quali sono i metodi principali utilizzati attualmente e quindi, finalmente, il problema relativamente nuovo dei reticoli geometrici.

Capitolo 1

Introduzione alla crittografia moderna

Nello scenario classico ci sono due parti che chiameremo Alice e Bob che vogliono comunicare tra loro. Per farlo hanno a disposizione solamente un canale non sicuro perché un canale sicuro è generalmente troppo costoso, se non proprio impossibile, da realizzare. Comunicare su un canale non sicuro vuol dire che il messaggio può essere intercettato da una terza parte che chiameremo Eva (E come Enemy, avversario; di fatto nel riferirsi a Eva si utilizza il genere maschile), che supporremo malintenzionato e pronto a fare qualsiasi cosa, come leggere il messaggio, modificarlo o impersonare una delle parti (vedi Figura 1.1).

Dobbiamo quindi trovare un modo per risolvere questi problemi.
Innanzitutto bisogna supporre che l'avversario conosca il metodo di cifratura utilizzato, assunzione nota come *principio di Kerckhoffs*, e che l'unico segreto sia la **chiave** utilizzata durante la cifratura. Puntare sulla segretezza del metodo si rivela poco sicuro poiché quest'ultimo potrebbe essere scoperto grazie ad un approccio basato su reverse engineering di dispositivi funzionanti, come anche a fughe d'informazioni da parte di addetti ai lavori; d'altra parte riducendo il numero di elementi da tenere nascosti aumentano le probabilità

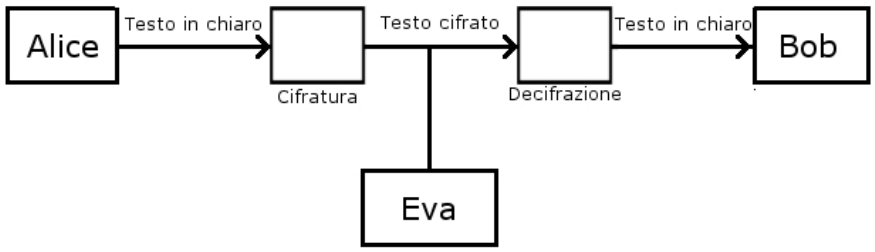

Figura 1.1: Schema di riferimento. Alice e Bob comunicano su un canale insicuro, Eva intercetta un testo cifrato

di successo.
L'algoritmo di cifratura dei messaggi è quindi noto (nella pratica attualmente quello utilizzato è AES) e l'unica cosa che si tiene segreta è la chiave che permette di cifrare e decifrare. Il problema si sposta quindi sul cercare il modo di condividere la chiave pur non avendo a disposizione un canale sicuro. Questo problema è stato brillantemente risolto nel 1976, quando Whitfield Diffie e Martin Hellman introdussero nell'articolo *New Directions in Cryptography* l'idea della cifratura a chiave pubblica: ogni utente dispone di due chiavi, una pubblica, nota a tutti, e una privata di sua esclusiva conoscenza; un messaggio criptato con la chiave pubblica può essere decriptato *soltanto* conoscendo la relativa chiave privata. La sicurezza di questo sistema richiede di trovare un problema matematico che faccia uso di algoritmi computazionalmente economici e che siano *difficili* (per una definizione più formale del termine *difficile* consultare Appendice A) da risolvere senza la conoscenza di specifiche informazioni, appunto la chiave privata.

Più formalmente si tratta di trovare una funzione f unidirezionale *con trapdoor* tale che

1. Trovare $f(x) = y$ sia facile per tutte le x.
2. Sia *unidirezionale*, cioè fissato y, trovare x t.c. $y = f(x)$ sia un problema computazionalmente intrattabile.

3. Esiste una *trapdoor*, ovvero esiste un modo computazionalmente semplice di invertire la funzione, ma solo conoscendo delle informazioni in più.

1.1 I problemi matematici

I problemi matematici attualmente utilizzati sono quelli della fattorizzazione in numeri primi e del logaritmo discreto.

1.1.1 Fattorizzazione - RSA

L'RSA è la prima implementazione del sistema a cifratura asimmetrica teorizzato da Diffie-Hellman nonché la più diffusa ed utilizzata; fu proposto nel 1978 da Ronald Rivest, Adi Shamir e Leonard Adleman, tre ricercatori del MIT, le iniziali dei quali compongono proprio l'acronimo RSA.
Questo sistema fonda la propria sicurezza sulla supposta difficoltà della fattorizzazione in numeri primi. Difficoltà *supposta* perché per quanto non sia stato trovato un algoritmo efficiente per fattorizzare un numero, non è neanche stato provato che un tale algoritmo non esista; per l'appunto si suppone che sia un problema computazionalmente intrattabile. Tuttavia è noto un algoritmo efficiente per computer quantistici, motivo per cui si ritiene opportuna la ricerca di un problema alternativo alla fattorizzazione e al logaritmo discreto.

Descrizione

Il procedimento per la generazione delle chiavi è il seguente:

- Si prendano $\boldsymbol{p}$ e $\boldsymbol{q}$, numeri primi nell'ordine delle centinaia di cifre in base alla sicurezza desiderata (RSA-2048 utilizza numeri primi lunghi più di 600 cifre decimali) e si considerano $\boldsymbol{n = pq}$ e $\boldsymbol{\varphi(n) = (p-1)(q-1)}$;

- Si sceglie $\boldsymbol{e}$ (esponente pubblico) coprimo con $\varphi(n)$, con $\boldsymbol{e} < \varphi(n)$;

- Si trova $\boldsymbol{d}$ tale che $\boldsymbol{de} \equiv 1(mod\varphi(n))$, ovvero l'inverso di $\boldsymbol{e}$ modulo $\varphi(n)$.

La *chiave pubblica* è **(n,e)**, la *chiave privata* è **(n,d)**.

Cifratura e decifrazione

Alice vuole mandare un messaggio **m** a Bob e ha a disposizione **(n,e)**, chiave pubblica di Bob. Alice manda il messaggio cifrato $\boldsymbol{c = m^e}$ **(mod n)**;
Bob riceve c e riottiene m calcolando $\boldsymbol{m = c^d}$ **(mod n)**; questo accade soltanto se $de \equiv 1(mod\varphi(n))$, il corretto funzionamento è infatti garantito dal piccolo teorema di Fermat.

Osservazioni

Osserviamo ora che tutte le richieste della funzione con trapdoor sono effettivamente rispettate:

1. Le operazioni compiute (trovare numeri primi, moltiplicarli, trovare e numero coprimo con $\varphi(n)$ e ottenere l'inverso con l'algoritmo euclideo esteso) hanno tutte complessità polinomiale, l'algoritmo è quindi efficientemente utilizzabile dalle parti legittime.

2. L'unico modo attualmente noto di trovare m a partire da c è la conoscenza di d; questo è facile da trovare conoscendo $\varphi(n)$, ma il problema di trovare $\varphi(n)$ equivale a quello di trovare una fattorizzazione per n: infatti, avendo p e q, $\varphi(n)$ si trova facilmente come $(p-1)(q-1)$; d'altra parte conoscendo n e $\varphi(n)$, p e q si ricavano dalle relazioni $p*q=n$ e $p+q=n-\varphi(n)+1$

3. La trapdoor è proprio la conoscenza della chiave privata d. Trovare m a partire da c, avendo a disposizione la chiave privata, consiste in

una semplice operazione di esponenziazione, algoritmo di complessità polinomiale.

1.1.2 Logaritmo discreto

Sia **G** un gruppo ciclico finito di n elementi e **a** un suo generatore; ogni elemento $\boldsymbol{g}$ di $\boldsymbol{G}$ può essere scritto nella forma $\boldsymbol{g = a^k}$ con k intero.
Il problema del logaritmo discreto sta nel trovare, fissato b, l'unico, a meno di congruenze modulo n, elemento k tale che $a^k = b$. Il problema sembra avere costo computazionale dello stesso ordine di grandezza della fattorizzazione. Anche qui l' effettiva difficoltà non è stata dimostrata; si suppone che il problema sia intrattabile, ma non si ha la certezza matematica.
Vediamo alcune possibili applicazioni:

Scambio delle chiavi di Diffie-Helmann

Alice e Bob vogliono comunicare utilizzando un algoritmo a cifratura simmetrica e vogliono scambiarsi una chiave segreta. Scelgono un numero **p** dell'ordine delle centinaia di cifre e un generatore **g** del gruppo ciclico $\mathbb{Z}_p^*$. Alice sceglie segretamente un numero **a**, calcola $\mathbf{A}=g^a \pmod{\mathbf{p}}$ e lo invia a Bob; Bob sceglie segretamente un numero **b**, calcola $\mathbf{B} = g^b \pmod{\mathbf{p}}$ e lo invia ad Alice.
La chiave è $\mathbf{K}=g^{ab} \pmod{\mathbf{p}}=A^b \pmod{\mathbf{p}}=B^a \pmod{\mathbf{p}}$.

Notiamo che g, p, A e B passano in chiaro sul canale insicuro, ma K è facile da calcolare solo conoscendo a o b, che però sono segreti. Si tratterebbe quindi di risolvere un problema di logaritmo discreto.
È stato formulato il problema di Diffie-Hellman: dati $g^a \pmod{p}$ e $g^b \pmod{p}$, esiste un altro modo di conoscere $g^{ab} \pmod{p}$? Questo problema si riduce a quello del logaritmo discreto, ovvero è difficile non più di quello, in simboli

$$D.H. \leq Dlog$$

Sistema a chiave pubblica di ElGamal

ElGamal è uno schema di cifratura a chiave pubblica proposto nel 1985 da Taher ElGamal, la cui difficoltà si riduce a quella del problema di Diffie-Hellman e quindi del logaritmo discreto. Il procedimento per la generazione delle chiavi è il seguente:

Descrizione Si sceglie un numero primo **p** grande e **g** generatore del gruppo $\mathbb{Z}_p^*$; si sceglie **b** numero casuale. Si calcola $\mathbf{B}=g^b \pmod p$
La chiave pubblica è **(B,g,p)**, la chiave privata è **b**.

Cifratura e decifrazione Alice vuole mandare un messaggio **m** a Bob e dispone di **(B,g,p)**, chiave pubblica di quest'ultimo. Alice genera un numero k casuale $0 < k < \boldsymbol{p}$. Costruisce la chiave $\mathbf{K}=\boldsymbol{B}^k \pmod{\boldsymbol{p}}$.

Cifra $\mathbf{C} = (\boldsymbol{C_1},\boldsymbol{C_2})$, con:

- $\boldsymbol{C_1}=\boldsymbol{g}^k \pmod p$
- $\boldsymbol{C_2}= \boldsymbol{K} \cdot \boldsymbol{m} \pmod p$

Per decifrare Bob ricava $\boldsymbol{K}$ da $\boldsymbol{C_1}$, infatti

$$\boldsymbol{K} = \boldsymbol{B}^k \pmod{\boldsymbol{p}} = (\boldsymbol{g}^b)^k \pmod{\boldsymbol{p}} = (g^k)^b \pmod p = (\boldsymbol{C_1})^b \pmod{\boldsymbol{p}}$$

Quindi ritrova **m**, come

$$\boldsymbol{m} = \boldsymbol{K}^{-1}\boldsymbol{C_2} \pmod{\boldsymbol{p}}$$

Sostituendo in un'unica espressione:

$$\boldsymbol{m} = \boldsymbol{C_1}^{-b}\boldsymbol{C_2} \pmod{\boldsymbol{p}}$$

ricordando che $\boldsymbol{C_1}^{-b} = \boldsymbol{C_1}^{\boldsymbol{p}-1-b}$

Osservazioni In ElGamal viene scelto casualmente un numero k diverso ad ogni cifratura, rendendo il procedimento non deterministico: cifrando più volte lo stesso messaggio si ottengono testi cifrati diversi. Per contro ElGamal è più lento di RSA e produce un messaggio cifrato più lungo.

Capitolo 2

I reticoli geometrici

Definizione 2.1 (Base)**.**
Siano dati n vettori $v_1, v_2, \ldots, v_n$ dello spazio $\mathbb{R}^n$ linearmente indipendenti. Si dice **reticolo** generato da $v_1, v_2, \ldots, v_n$ l'insieme dei vettori della forma

$$a_1 v_1 + a_2 v_2 + \cdots + a_n v_n \qquad \text{con } a_1, \ldots, a_n \in \mathbb{Z}$$

I vettori $v_1, v_2, \ldots, v_n$ costituiscono una **base** del reticolo.

Osservazione. La base di un reticolo non è unica. Di più, ogni reticolo possiede infinite basi, infatti, ad esempio, in $\mathbb{R}^2$ si vede subito che se $\{v_1, v_2\}$ è una base, anche $\{w_1, w_2\}$ con $w_1 = v_1 + kv_2$, $w_2 = v_2$ e k $\in \mathbb{Z}$ è una base.

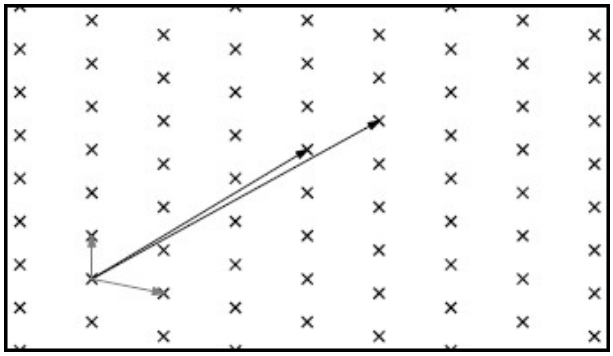

Figura 2.1: Esempio di reticolo geometrico in $\mathbb{R}^2$ con due differenti basi

Si definisce **lunghezza** di un vettore v=$(x_1, x_2, \ldots, x_n)$ la quantità:

$$||v|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Si definisce **determinante** di un reticolo geometrico di base $v_1, v_2, \ldots, v_n$ la quantità

$$D = |det(v_1, v_2, \ldots, v_n)|$$

Si osserva che il determinante è geometricamente il volume generato dai vettori della base e che questo è indipendente dalla base scelta.

Storicamente i reticoli geometrici sono stati oggetto di studio sin dalla fine del 18° secolo da parte di matematici come Lagrange, Gauss e più tardi Minkowski. Più di recente sono diventati argomento d'interesse per le applicazioni informatiche: sono infatti usati come strumenti in vari algoritmi per risolvere vari problemi matematici; sono state trovate molte applicazioni anche in criptoanalisi e solo ultimamente (anni '90) si è scoperto che è possibile usarli attivamente per creare un sistema crittografico.
Il problema computazionale su cui questi sistemi si basano, nonché il più rilevante legato ai reticoli geometrici, è **SVP** (Shortest Vector Problem), ovvero la ricerca di un vettore di norma minima nel reticolo.

2.1 Un nuovo problema

In **SVP** si cerca di trovare, data una base di un reticolo geometrico, un vettore non nullo di norma minima appartenente al reticolo; una sua approssimazione, molto utile in crittografia è $\boldsymbol{SVP_\gamma}$, ovvero trovare il vettore non nullo più corto entro un'approssimazione γ. I due problemi sono entrambi ritenuti computazionalmente molto difficili.
Prima di analizzare nel dettaglio quali siano le possibili applicazioni del problema dei vettori minimi in un reticolo, esaminiamo i risultati ottenuti finora nel risolverli.

2.1.1 Riduzione reticolare

[1] Per n molto piccolo è possibile ricorrere a un algoritmo di riduzione: si cambia la base del reticolo con una *migliore*, una base **ridotta**. Vediamo nel dettaglio il caso bidimensionale.

Definizione 2.2 (Base ridotta)**.**
Una base $\{v_1, v_2\}$ di un reticolo bidimensionale si dice **ridotta** se vale:

- $||v_1|| \leq ||v_2||$
- $-\frac{1}{2} \leq \frac{v_1 \cdot v_2}{v_1 \cdot v_1} \leq +\frac{1}{2}$

Ogni base $\{v_1, v_2\}$ può essere trasformata in una base $\{v_1^*, v_2^*\}$ ridotta grazie al seguente:

Teorema 2.1.1. *Sia $\{v_1, v_2\}$ base di un reticolo geometrico bidimensionale; si consideri il seguente algoritmo:*

1. *se $||v_1|| > ||v_2||$ scambiare v_1 con v_2*
2. *si considera $\mu = \frac{v_1 \cdot v_2}{v_1 \cdot v_1}$ e si considera t l'intero più vicino a μ*
3. *se t=0 l'algoritmo ha termine, altrimenti si pone $v_2 = v_2 - tv_1$ e si ricomincia il ciclo.*

Allora l'algoritmo ha termine in un numero finito di passi e v_1 è un vettore di lunghezza minima del reticolo.

Dimostrazione. Dimostriamo innanzitutto che l'algoritmo ha termine. Consideriamo $v_2' = v_2 - \mu v_1$ da cui

$$v_2 = v_2' + \mu v_1. \tag{2.1}$$

Allora si ha

$$v_2 - tv_1 = v_2' + (\mu - t)v_1. \tag{2.2}$$

Ora, v_1 e v_2' sono ortogonali e quindi, da 2.1 abbiamo che

$$||v_2||^2 = ||v_2'||^2 + ||\mu v_1||^2 = ||v_2'||^2 + |\mu|^2||v_1||^2$$

e, allo stesso modo, da 2.2

$$||v_2 - tv_1||^2 = ||v_2'||^2 + |\mu - t|^2||v_1||^2$$

Osserviamo adesso che se $\mu < \frac{1}{2}$ allora t=0 e $\mu - t = \mu$; se invece $\mu > \frac{1}{2}$, allora $|\mu - t| < \frac{1}{2} < |\mu|$. In definitiva, se $t \neq 0$, $|\mu - t| < |\mu|$. E quindi:

$$||v_2 - tv_1||^2 = ||v_2'||^2 + |\mu - t|^2||v_1||^2 < ||v_2'||^2 + |\mu|^2||v_1||^2 = ||v_2||^2$$

Questo vuol dire che al passo 3 dell'algoritmo il vettore v_2 viene sempre sostituito con uno di norma più piccola e, poiché all'interno di un reticolo i vettori più corti di v_2 sono in numero finito, non è possibile che l'algoritmo continui all'infinito. Si ottiene dunque una base ridotta $\{v_1^*, v_2^*\}$.

Proviamo ora che v_1^* è un vettore di norma minima. Consideriamo un vettore v non nullo appartenente al reticolo: $v = av_1^* + bv_2^*$, con $a, b \in \mathbb{Z}; a, b$ non entrambi nulli.
Allora, $||v||^2 =< av_1^* + bv_2^*, av_1^* + bv_2^* >= a^2||v_1^*||^2 + b^2||v_2^*||^2 + 2abv_1^* \cdot v_2^*$, ma, per definizione di base ridotta, vale

- $||v_1^*|| \leq ||v_2^*||$
- $-\frac{1}{2} \leq \frac{v_1^* \cdot v_2^*}{v_1^* \cdot v_1^*} \leq +\frac{1}{2}$, da cui $2v_1^* \cdot v_2^* > -||v_1^*||^2$ e quindi $2|ab|v_1^* \cdot v_2^* > -|ab|||v_1^*||^2$

Quindi

$$\begin{aligned} ||v||^2 &= a^2||v_1^*||^2 + b^2||v_2^*||^2 + 2|ab|v_1^* \cdot v_2^* > \\ &> a^2||v_1^*||^2 + b^2||v_1^*||^2 - |ab|||v_1^*||^2 = \\ &= ||v_1^*||^2(a^2 + b^2 - |ab|) \end{aligned}$$

Osserviamo che $(a^2 + b^2 - |ab|)$ può essere scritto come $(|a| - |b|)^2 + |ab|$, somma di un quadrato e di un valore assoluto e pertanto sempre positivo. In particolare, poiché a e b sono entrambi non nulli $(a^2 + b^2 - |ab|) \neq 0$, cioè essendo a valori interi, $(a^2 + b^2 - |ab|) \geq 1$.

In definitiva $||v||^2 > ||v_1^*||^2(a^2 + b^2 - |ab|) \geq ||v_1^*||^2$. □

Esempio 1. Prendiamo il reticolo generato da v_1=(3,12) e v_2=(5,10).

- Prima iterazione
  - passo 1: $||v_1||^2 = 135$, $||v_2||^2 = 125$; si scambia v_1 con v_2. v_1=(5,10) e v_2=(3,12).
  - passo 2: si considera $\mu = \frac{v_1^* \cdot v_2^*}{v_1^* \cdot v_1^*} = \frac{135}{125}$. Quindi t, intero più vicino a μ è 1.
  - passo 3: $t \neq 0$, quindi si assegna $v_2 = v_2 - tv_1 = (3, 12) - (5, 10) = (-2, 2)$
- Seconda iterazione
  - passo 1: $||v_1||^2 = 125$, $||v_2||^2 = 8$; si scambia v_1 con v_2. v_1=(-2,2) e v_2=(5,10).
  - passo 2: si considera $\mu = \frac{v_1^* \cdot v_2^*}{v_1^* \cdot v_1^*} = \frac{10}{8}$. Quindi t, intero più vicino a μ è 1.
  - passo 3: $t \neq 0$, quindi si assegna $v_2 = v_2 - tv_1 = (5, 10) - (-2, 2) = (7, 8)$
- Terza iterazione
  - passo 1: $||v_1||^2 = 8$, $||v_2||^2 = 105$; si lascia l'ordine inalterato.
  - passo 2: si considera $\mu = \frac{v_1^* \cdot v_2^*}{v_1^* \cdot v_1^*} = \frac{2}{8}$. Quindi t, intero più vicino a μ è 0.

– passo 3: $t = 0$, quindi l'algoritmo si arresta. La base ridotta è $\{(-2, 2), (7, 8)\}$ e (-2,2) è un vettore di lunghezza minima per il reticolo.

Osservazione. Notiamo che nel terzo passo dell'algoritmo di riduzione si prende il vettore $v_2 - \frac{v_1 \cdot v_2}{v_1 \cdot v_1} v_1$, che non è altro che il vettore ortogonalizzato secondo Gram-Schmidt; questo vettore però può non appartenere al reticolo, per ovviare a ciò si prende t come l'intero più vicino a μ. Allora $v_2 - tv_1$ è un vettore del reticolo e il teorema prova che l'algoritmo ha termine, portando a una base *quasi ortogonale*.

Per studiare il caso multidimensionale, richiamiamo il processo di ortogonalizzazione di Gram-Schmidt:

Gram-Schmidt in $\mathbf{R}^n$ [2] Sia $\{b_1, \ldots, b_n\}$ una base di uno spazio euclideo. Si definisce $\{b_1^*, \ldots, b_n^*\}$ base ortogonale come segue:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \qquad (1 \leq i \leq n)$$

dove

$$\mu_{i,j} = \frac{b_i \cdot b_j^*}{b_j^* \cdot b_j^*} \qquad (1 \leq j < i \leq n)$$

sono le proiezioni dell'i-simo vettore sui j vettori già trovati della base ortogonale .

Si verifica facilmente per induzione che la base così costruita è effettivamente ortogonale (N.B. Non è detto che sia ortonormale, manca il passo di normalizzazione del vettore).
In particolare, se consideriamo come B la matrice $[b_1, b_2, \ldots, b_n]$, B^* la matrice $[b_1^*, b_2^*, \ldots, b_n^*]$ e M la matrice dei coefficienti dei b_i nella nuova base b_i^*, si ha $B = MB^*$ con M triangolare superiore con 1 lungo la diagonale. Inoltre, essendo B^* costituita da vettori ortogonali, $B^* B^{*t}$ è una matrice diagonale con all'i-simo posto $||b_i^*||^2$.

Da quanto detto segue che $det(B)^2 = det(B^*)^2 = \prod_{i=1}^{n} ||b_i^*||^2$.
Osserviamo infine che det(B) è il determinante del reticolo geometrico.

Proposizione 2.1.2 (Disuguaglianza di Hadamard)**.** *Sia L un reticolo geometrico di determinante D, sia* $\{b_1, \ldots, b_n\}$ *una base di L, allora vale:*

$$D \leq \prod_{i=1}^{n} ||b_i||$$

Dimostrazione. Dal processo di ortogonalizzazione di Gram-Schmidt si trova:

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^* \Rightarrow b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$$

Quindi, dall'ortogonalità dei b_i^*:

$$||b_i||^2 = ||b_i^*||^2 + \sum_{j=1}^{i-1} |\mu_{i,j}|^2 ||b_j^*||^2 \tag{2.3}$$

da cui si ricava $||b_i^*||^2 < ||b_i||^2$.
Infine da Gram-Schmidt si ha

$$D = \prod_{i=1}^{n} ||b_i^*|| < \prod_{i=1}^{n} ||b_i||$$

□

2.1.2 Algoritmo LLL

Il procedimento visto sopra per il caso bidimensionale si estende difficilmente oltre le dimensioni 3-4 perché i tempi di esecuzione dell'algoritmo diventano presto molto lunghi. L'idea di base è però d'aiuto nella comprensione dell'algoritmo di riduzione più famoso ed utilizzato, ideato nel 1982 da A.Lenstra, H.Lenstra e L.Lovász. Nel loro lavoro viene data una nuova definizione di base ridotta (detta LLL-ridotta) e un algoritmo polinomiale di riduzione,

detto LLL dalle iniziali dei ricercatori.
Si consideri $\{b_1, b_2, \ldots, b_n\}$ una base di $\mathbf{L}$, reticolo geometrico e si prenda $\{b_1^*, \ldots, b_n^*\}$ la base ortogonalizzata con il procedimento di Gram-Schmidt.

Definizione 2.3. Utilizzando le notazioni precedenti, la base $\{b_1, b_2, \ldots, b_n\}$ è detta **LLL-ridotta** se vale:

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{con } 1 \leq j < i \leq n$$

e

$$||b_i^* + \mu_{i,i-i} b_{i-1}^*||^2 \geq \frac{3}{4}||b_{i-1}^*||^2 \qquad \text{con } 1 < i \leq n$$

o equivalentemente

$$||b_i^*||^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^2\right)||b_{i-1}^*||^2$$

Si ha il seguente

Teorema 2.1.3. *Sia* $\mathbf{L}$ *un reticolo geometrico di determinante D, sia* $\{b_1, b_2, \ldots, b_n\}$ *una base LLL-ridotta di un reticolo geometrico. Allora vale:*

(1)

$$D \leq \prod_{i=1}^{n} ||b_i|| \leq 2^{\frac{n(n-1)}{4}} D$$

(2)

$$||b_j|| \leq 2^{\frac{i-1}{2}} ||b_i^*|| \quad \text{con } 1 \leq j < i \leq n$$

(3)

$$||b_1|| \leq 2^{\frac{n-1}{4}} D^{\frac{1}{n}}$$

(4) $\forall x \in \mathbf{L}, x \neq 0$

$$||b_1|| \leq 2^{\frac{n-1}{2}} ||x||$$

Dimostrazione. La prima disuguaglianza in (1) è il Corollario 2.1.2;
d'altra parte i b_i sono LLL-ridotti, quindi $||b_i^*||^2 \geq (3/4 - \mu_{i,i-1}^2)||b_{i-1}^2|| \geq ||b_{i-1}||^2/2$, poiché $|\mu_{i,i-1}| \leq 1/2$.
Per induzione, si ha che $||b_i^*||^2 \geq 1/2^{i-j}||b_j^*||^2$ e quindi

$$||b_j^*||^2 \leq 2^{i-j}||b_i^*||^2 \text{ con } j < i \tag{2.4}$$

Proseguendo dalla 2.3 del corollario si ha:

$$||b_i||^2 = ||b_i^*||^2 + \sum_{j=1}^{i-1} |\mu_{i,j}|^2||b_j^*||^2$$

Utilizzando 2.4 e che $|\mu_{i,j}| \leq \frac{1}{2}$

$$||b_i||^2 \leq ||b_i^*||^2 + \frac{1}{4}\sum_{j=1}^{i-1} 2^{i-j}||b_i^*||^2$$

ed infine

$$||b_i||^2 \leq \frac{2^{i-1}+1}{2}||b_i^*||^2 \tag{2.5}$$

Moltiplicando sugli $i = 1 \dots n$, estraendo la radice quadrata e ricordando che $D = \prod_{i=1}^{n} ||b_i^*||$ si ottiene la seconda parte di (1), con un esponente anche leggermente migliore.

Da 2.5 e 2.4 si ha

$$||b_j||^2 \leq (2^{i-2} + 2^{i-j-1})||b_i^*||^2$$

da cui segue facilmente la (2).

Ponendo j=1 nella (2) e moltiplicando per $i = 1 \dots n$ si trova

$$\begin{aligned}(||b_1||^2)^n &\leq \prod_{i=1}^{n} 2^{\frac{i-1}{2}}||b_i^*||^2 = \prod_{i=1}^{n} 2^{\frac{i-1}{2}} \prod_{i=1}^{n} ||b_i^*||^2 \\ &= 2^{\sum_{i=1}^{n} \frac{i-1}{2}} D^2 = 2^{\frac{n(n-1)}{4}} D^2\end{aligned}$$

Ed estraendo la radice $2N$-sima si trova la (3), anche qui con un esponente migliore.

Per la (4), esiste una i tale che $x = \sum_{i \leq j \leq i} r_j b_j = \sum_{i \leq j \leq i} s_j b_j^*$, con $r_i \neq 0$, $r_j \in \mathbb{Z}$ e $s_j \in \mathbb{R}$. Dalla definizione dei b_j^* è evidente che $r_i = s_i$, da cui

$$||x||^2 \geq s_i^2 ||b_i^*||^2 = r_i^2 ||b_i^*||^2 \geq ||b_i^*||^2$$

dove, ricordiamo, $r_i \neq 0$; d'altra parte da (2) si sa che

$$||b_i^*||^2 \geq 2^{i-1} ||b_1||^2 \geq 2^{i-n} ||b_1||^2$$

e quindi, in definitiva

$$||x||^2 \geq 2^{i-n} ||b_1||^2$$

da cui estraendo la radice quadrata si ottiene l'asserto. □

Osservazione. Come visto in Gram-Schmidt, D, il determinante del reticolo, è il prodotto delle norme dei vettori di una base ortogonale; la (1) del teorema ci dice che i b_i di una base LLL-ridotta sono molto vicini ad essere ortogonali, almeno se n è piccolo: infatti il prodotto delle loro norme è maggiore di D di non più di un fattore $2^{\frac{n(n-1)}{4}}$. Si noti che durante la dimostrazione di (1) si trova un esponente più piccolo rispetto al valore dato nell'enunciato, ma utilizzare quello non porta ad alcun miglioramento nella stima ottenuta in (4).
La (4) è di fatto il punto più interessante per la nostra analisi: il primo vettore di una base LLL-ridotta è, in norma, più grande di $||x||$ non più di un fattore $2^{\frac{n-1}{2}}$ qualunque sia x non nullo del reticolo.
In particolare, questo vale anche scegliendo un vettore di norma minima (che, pur non essendo noto, esiste sicuramente), sia λ questa norma:

$$||b_1|| \leq 2^{\frac{n-1}{2}} \lambda$$

Cioè b_1, pur non essendo di norma realmente minima, è comunque un vettore

corto del reticolo.

Riportiamo adesso l'algoritmo LLL che permette, data in input una base di un reticolo geometrico, di ottenere una base LLL-ridotta

Algoritmo 2.1.4 (LLL)**.** Sia data una base $\{b_1, \ldots, b_n\}$, il seguente algoritmo ha termine e porta ad una base LLL-ridotta.

1- Inizializzazione
Si inizializza $k \leftarrow 2$, $k_{max} \leftarrow 1$, $b_1^* \leftarrow b_1$, $B_1 \leftarrow b_1 \cdot b_1$ e $H \leftarrow I_n$, con I_n matrice identica $n \times n$.

2- Gram-Schmidt
Se $k \leq k_{max}$ andare al punto 3. Altrimenti, porre $k_{max} \leftarrow k$, $b_k^* \leftarrow b_k$;
per $j = 1, \ldots, k-1$ si pone $\mu_{k,j} \leftarrow \frac{b_k \cdot b_j^*}{B_j}$ e $b_k^* \leftarrow b_k^* - \mu_{k,j} b_j^*$;
si calcola $B_k \leftarrow b_k^* \cdot b_k^*$; se $B_k = 0$ si dà un messaggio d'errore, perché i v_i non formano una base, e l'algoritmo termina.

*(**Oss**: Si tratta proprio del procedimento di ortogonalizzazione di Gram-Schmidt; i coefficienti $\mu_{k,j}$ vengono calcolati in modo incrementale, piuttosto che tutti all'inizio)*

3- Test delle LLL-condizioni
Si eseguono i sotto-algoritmi $RED(k, k-1)$. Se $B_k < (0,75 - \mu_{k,k-1}^2) B_{k-1}$, si esegue il sotto-algoritmo $SWAP(k)$, si pone quindi $k \leftarrow max(2, k-1)$ e ritorno al punto 3. Altrimenti per $l = k-2, k-3, \ldots, 1$ si esegue il sottoalgoritmo $RED(k, l)$ e si pone $k \leftarrow k+1$.

*(**Oss**: si testano le condizioni di base LLL-ridotta con l'algoritmo RED e, eventualmente, si scambiano vettori con l'algoritmo SWAP)*

4- Test di conclusione
Se $k \leq n$ si torna al punto 2; altrimenti si dà come output la base LLL-ridotta $\{b_1, \ldots, b_n\}$ e la matrice H $\in GL_n(\mathbb{Z})$ di cambiamento di base e l'algoritmo ha termine.

Nel passo 3 vengono utilizzati i seguenti sotto-algoritmi:

Algoritmo 2.1.5 (Sotto-algoritmo $RED(k,l)$)**.** Se $|\mu_{k,l}| \leq 0.5$ il sotto-algoritmo RED termina. Altrimenti, consideriamo q l'intero più vicino a $\mu_{k,l}$, cioè $q \leftarrow \lfloor \mu_{k,l} \rceil = \lfloor 0.5 + \mu_{k,l} \rfloor$, dove $\lfloor x \rfloor$ è la funzione parte intera inferiore.
Dopo di che si pone $b_k \leftarrow b_k - qb_l$, $H_k \leftarrow H_k - qH_l$, $\mu_{k,l} \leftarrow \mu_{k,l} - q$ e quindi, per ogni i tale che $1 \leq i \leq l-1$ si pone $\mu_{k,i} \leftarrow \mu_{k,i} - q\mu_{k,i}$.

Algoritmo 2.1.6 (Sotto-algoritmo $SWAP(k)$)**.** Si scambiano i vettori b_k e b_{k-1}, poi si scambiano H_k e H_{k-1} e se $k > 2$ si scambiano, per ogni j tale che $1 \leq j \leq k-2$, $\mu_{k,j}$ con $\mu_{k-1,j}$. Dunque si pone, nell'ordine, $\mu \leftarrow \mu_{k,k-1}$, $B \leftarrow B_k + \mu^2 B_{k-1}$, $\mu_{k,k-1} \leftarrow \frac{\mu B_{k-1}}{B}$, $b \leftarrow b^*_{k-1}$, $b^*_{k-1} \leftarrow b^*_k + \mu b$, $b^*_k \leftarrow -\mu_{k,k-1} b^*_k + \frac{B_k}{B} b$,$B_k \leftarrow \frac{B_{k-1} B_k}{B}$ e $B_{k-1} \leftarrow B$. Infine, per $i = k+1, k+2, \ldots, k_{max}$ porre $t \leftarrow \mu_{i,k}$,$\mu_{i,k} \leftarrow \mu_{i,k-1} \leftarrow t + \mu_{k,k-1}\mu_{i,k}$, e il sotto-algoritmo ha termine.

Nota L'indice k viene posto inizialmente uguale a 2 e l'algoritmo termina quando, al punto 4, $k > n$. Al punto 3 può però accadere sia che k venga incrementato di 1, quando è soddisfatta la condizione di Lovász $||b^*_k||^2 \geq \left(\frac{3}{4} - \mu^2_{k,k-1}\right)||b^*_{k-1}||^2$, sia che venga diminuito di 1, quando la condizione non si verifica e si richiama il sotto-algoritmo SWAP. Non è evidente che questa procedura arrivi sempre a trovare $k > n$ e che quindi l'algoritmo termini. Questo può tuttavia essere dimostrato, vedi [2, Algoritmo 2.6.3, pag.88], dove si dimostra anche che il tempo di esecuzione di LLL è $O(n^6 \log^3 B)$.

2.1.3 Un buon problema?

[3]In definitiva SVP deve essere senz'altro considerato un problema molto interessante dal punto di vista crittografico. Le ragioni sono molteplici: innanzitutto, come già detto, è un problema computazionalmente difficile, i migliori algoritmi per risolverlo hanno complessità di ordine esponenziale rispetto alla dimensione del reticolo. D'altra parte LLL gira in tempo polinomiale, ma trova un vettore corto entro un'approssimazione nell'ordine di

$2^{O(n)}$ (di recente si è riuscito a migliorarlo portandolo a $2^{O(n(\log\log n)^2/\log n)}$) che, con n molto grande può non essere sufficiente.
Un altro motivo che rende SVP e i reticoli geometrici meritevoli di attenzione è il basso costo computazionale richiesto nei calcoli. Questo permetterebbe di eseguire operazioni di cifratura anche utilizzando dispositivi con un'esigua potenza di calcolo e di conseguenza più economici.
Inoltre bisogna ricordare che attualmente non si dispone di grandi alternative ad RSA, alternative che saranno effettivamente necessarie nel caso della scoperta di un algoritmo di fattorizzazione in tempo polinomiale; d'altra parte un algoritmo di fattorizzazione in grado di operare in tempo polinomiale è già noto per computer quantistici, mentre attualmente non sono noti attacchi quantistici a SVP.
Infine la proprietà più interessante: nel 1996 Miklos Ajtai dimostrò nell'articolo *Generating hard instances of lattice problems* [4] che la difficoltà di quasi ogni istanza del problema è la stessa di quella del caso peggiore. Tutti gli altri problemi, come anche la fattorizzazione, si basano sulla supposta difficoltà del caso medio: si suppone che il problema sia *mediamente* difficile, ma questo non è normalmente provato.
In SVP invece questo problema non sorge, grazie all'equivalenza della difficoltà media al caso peggiore *(average case/worst-case equivalence)*. Formalmente ciò vuol dire che se si riesce, anche con una probabilità bassa, a risolvere una qualunque istanza del problema, allora si è in grado di risolverle *tutte*. Intuitivamente questo significa che quasi ogni istanza del problema ha la stessa difficoltà del caso peggiore.
Il valore di questa equivalenza è reso ancor più significativo dal fatto che SVP è stato dimostrato essere un problema NP-hard (cfr. Appendice A).

2.2 Applicazioni

Le possibili applicazioni matematiche dei reticoli geometriche sono molteplici tra cui, ad esempio, citiamo la fattorizzazione di polinomi a coefficienti interi

[5], la ricerca di nucleo e immagine di una matrice a coefficienti interi o la ricerca di dipendenze algebriche o lineari [2].
Noi ci occuperemo soltanto delle applicazioni relative alla crittografia: i reticoli sono stati sfruttati per portare attacchi a vari crittosistemi, tra cui RSA sotto opportune condizioni, ma soprattutto, a partire dal lavoro citato di Ajtai, per creare dei critto-sistemi a chiave pubblica.

2.2.1 Ajtai-Dwork

Miklos Ajtai, insieme con Cynthia Dwork, presentò nell'articolo *A Public-Key Cryptosystem with Worst-Case/Average-case Equivalence*[6] datato 8 novembre 1996 un sistema crittografico fondato proprio sulla difficoltà di $uSVP$, un caso particolare di SVP. Negli anni sono stati fatti vari tentativi di migliorare il crittosistema; fra questi riportiamo il lavoro di Goldreich, Goldwasser e Halevi[7] del 1997 e quello di Regev[8] del 2004. Durante l'esposizione verrà usata la formulazione del crittosistema e le notazioni date in [7].

Descrizione euristica [6] Il crittosistema si basa su una famiglia segreta di iperpiani, che formano la chiave privata. Viene poi dato un metodo per generare un punto vicino a un punto fissato su uno degli iperpiani nella famiglia, che è la chiave pubblica.
La cifratura avviene bit per bit: per cifrare uno 0 si cerca, utilizzando la chiave pubblica, un vettore casuale vicino a uno degli iperpiani. Per cifrare un 1 si prende un qualunque vettore casuale.
Per la decifrazione si verifica, avendo come chiave privata la famiglia di iperpiani, la distanza del punto dagli iperpiani: se è *vicino* viene decifrato uno 0; se non lo è si decifra un 1. C'è una possibilità di errore, ma questa cresce solo polinomialmente con l'ordine. Vediamo ora più nel dettaglio il funzionamento del sistema:

Definizioni Si denota con $\mathbb{Z} + \epsilon$ l'insieme dei numeri reali la cui distanza dall'intero più vicino sia al più ϵ, dove $0 < \epsilon < \frac{1}{2}$. Dati n vettori linearmente

indipendenti $w_1, \ldots, w_n$, il parallelepipedo formato dai w_i è l'insieme:

$$P(w_1, \ldots, w_n) := \Big\{ \sum_i \alpha_i w_i : \alpha_i \in [0,1), i = 1, \ldots, n \Big\}$$

La *larghezza* di $P(w_1, \ldots, w_n)$ è il minimo sugli i delle distanze tra i w_i e i sottospazi generati dagli altri w_j.

Dato un parallelepipedo P e un vettore v, il *ridotto di v modulo P* è un vettore $v' \in P$ tale che $v' = v + \sum_i c_i w_i$, dove $c_i \in \mathbb{Z}$

Descrizione dei parametri del sistema

Dato il parametro di sicurezza n, si considera $m := n^3$ e $\rho_n := 2^{n \log n}$. Chiamiamo B_n il cubo n-dimensionale di lato ρ_n e S_n la sfera n-dimensionale di raggio n^{-8}, ovvero:

$$B_n := \{x \in \mathbb{R}^n : 0 \leq x_i < \rho_n,\ i = 1, \ldots, n\} \quad \text{e} \quad S_n = \{x \in \mathbb{R}^n : ||x|| \leq n^{-8}\}$$

Creazione delle chiavi Le chiavi vengono generate nel modo seguente:

Chiave privata: dato il parametro di sicurezza n, la chiave privata è un vettore u scelto casualmente nella sfera unitaria n-dimensionale.

Chiave pubblica: data la chiave privata u, sia $\mathbf{H_u}$ la distribuzione di punti di B_n ottenuta dal seguente processo:

1. Si sceglie in maniera uniformemente casuale un punto a dall'insieme $\{x \in B_n \mid < x, u > \in \mathbb{Z}\}$
   (N.B. $< , >$ rappresenta l'usuale prodotto scalare.)

2. Per $i = 1, \ldots, n$, si prendono $\delta_1, \ldots \delta_n$ in maniera uniformemente casuale da S_n.

3. Si restituisce il punto $v = a + \sum_i \delta_i$.

Con queste notazioni la chiave pubblica corrispondente alla chiave privata u si ottiene prendendo indipendentemente gli $n+m$ punti $w_1, \ldots, w_n, v_1, \ldots, v_m$

a caso dalla distribuzione $\mathbf{H}_u$, con la condizione che la larghezza del parallelepipedo $P(w_1, \ldots, w_n)$ sia almeno $n^{-2}\rho_n$. Adotteremo per il seguito le notazioni $\boldsymbol{w} := (w_1, \ldots, w_n)$, $\boldsymbol{v} := (v_1, \ldots, v_m)$ e infine $\boldsymbol{e} := (\boldsymbol{w}, \boldsymbol{v})$.

Cifratura e decifrazione

Il crittosistema Ajtai-Dwork agisce bit per bit, ovvero se $m = \sigma_1\sigma_2 \ldots \sigma_l$, ogni bit σ_i è cifrato separatamente dagli altri.

Per cifrare uno '0', si scelgono uniformemente $b_1, \ldots, b_m$ in $\{0, 1\}$ e si riduce il vettore $\sum_{i=1}^{m} b_i \cdot v_i$ modulo il parallelepipedo $P(w)$. Il vettore $x = \sum_{i=1}^{m} b_i \cdot v_i \bmod P(w)$ è il testo cifrato corrispondente al bit '0'.

Per cifrare un '1' scegliamo casualmente un vettore x nel parallelepipedo $P(w)$. Questo vettore è il testo cifrato corrispondente al bit '1'.

Per decifrare, preso x messaggio cifrato e u la chiave privata, si calcola $\tau =< x, u >$. Si decifra x come 0 se τ è distante da un intero meno di $1/n$, altrimenti si decifra un 1.

Errori nella decifrazione Si vede facilmente che se x è la cifratura di un 1, la distanza di $< x, \tau >$ da un intero è distribuita quasi uniformemente in $[0, 1)$. D'altra parte, se x è la cifratura di 0, la distanza di $< x, \tau >$ dall'intero più vicino è sempre minore di $1/n$ in valore assoluto. Uno '0' verrà quindi sempre decifrato come '0',ma un '1' può essere decifrato come '0' con una probabilità di $2/n$.
Una delle variazioni proposte [7] presenta una formulazione del crittosistema che permette di evitare gli errori nella decifrazione senza perdere le caratteristiche di sicurezza del sistema.

Sicurezza

Nel loro articolo [6] i due creatori del sistema dimostrano che questo è sicuro a meno che si riesca a risolvere una qualunque istanza di *uSVP (unique Shortest Vector Problem)*. In altri termini se per un'istanza qualsiasi la cifratura di 1 può essere distinta dalla cifratura di uno 0 in tempo polinomiale con probabilità almeno $\frac{1}{2} + n^{-c_1}$, con $c_1 > 0$, allora il peggior caso di uSVP ha una soluzione probabilistica in tempo polinomiale. uSVP è una modifica di SVP: dato L, un reticolo geometrico n-dimensionale, si cerca il più corto vettore v non nullo di L, dove v è *unico* nel senso che qualunque altro vettore la cui lunghezza sia al più $n^c||v||$ è parallelo a v. Questo problema è tra i tre problemi citati da Ajtai nel suo articolo del 1996[4]. Qui viene dato un metodo casuale per generare istanze difficili di uno dei problemi sui reticoli in modo che se questa istanza ha una soluzione in tempo polinomiale allora tutti e tre i problemi (incluso uSVP) hanno soluzione. Vale quindi anche per $uSVP$ l'equivalenza al caso-peggiore, ma d'altra parte la difficoltà del problema non è ancora stata compresa bene come SVP. [9]

Limiti

Il più grosso problema di Ajtai-Dwork è quello dell'efficienza: su un reticolo geometrico di dimensione n la dimensione della chiave pubblica è un $O(n^4)$ bit e ogni bit viene cifrato in $O(n^2)$ bit!!
Ajtai ha presentato un altro crittosistema più efficiente in cui la chiave pubblica è un $O(n)$ e ogni bit viene cifrato in $O(n^2)$ bit. La sicurezza di quest'ultimo crittosistema tuttavia non sembra forte come quella di altri basati sui reticoli geometrici ed inoltre non vale la proprietà del caso-peggiore che vale per Ajtai-Dwork. [9]

2.2.2 NTRU

Nello stesso anno (1996) in cui Ajtai e Dwork pubblicarono il loro crittosistema tre matematici: J.Hoffstein, J.Pipher e J.H. Silverman pubblicarono la lo-

ro proposta nell'articolo NTRU: A Ring-Based Public Key Cryptosystem[10]. NTRU presenta tra i vantaggi il basso costo computazionale ($O(n^2)$ operazioni per un messaggio lungo n contro, ad esempio, le $O(n^3)$ operazioni di RSA, lunghezza della chiave dell'ordine di $O(n)$, contro ad esempio $O(n^2)$ di RSA) e la resistenza agli attacchi quantistici.

Descrizione

[1] NTRU lavora nell'anello quoziente $\mathbf{R}$=$\mathbb{Z}[x]/(x^N+1)$ costituito da polinomi di grado al più N-1 a coefficienti interi. Data la scrittura usuale dei polinomi $f, g \in \mathbf{R}$

$$f = a_{N-1}x^{N-1} + \cdots + a_0 \quad \text{e} \quad g = b_{N-1}x^{N-1} + \cdots + b_0$$

si definisce l'operazione $*$ in $\mathbf{R}$ come l'usuale prodotto di polinomi nel quoziente, ovvero:

$$h = f * g = c_{N-1}x^{N-1} + c_{N-2}x^{N-2} + \cdots + c_1x + c_0$$

dove

$$c_i = \sum_{j+k \equiv i \mod N} a_j b_k$$

Creazione delle chiavi NTRU è un sistema a chiave pubblica, vediamo come si generano le chiavi:
Si scelgono tre interi *N, p e q*, con $MCD(p, q) = 1$ e p molto più piccolo di q; quello della scelta ottimale di questi parametri è un problema interessante su cui si concentrano molti studi, vedremo qualche risultato più avanti.
Si scelgono quindi in segreto due polinomi f e g per cui esistano polinomi F_p e F_q tali che:

$$F_p * f \equiv 1 \pmod{p} \quad \text{e} \quad F_q * f \equiv 1 \pmod{q}$$

Questi si trovano utilizzando l'algoritmo euclideo (modificato per lavorare sui polinomi). Si calcola infine:

$$h = F_q * g \pmod{q}$$

e la chiave pubblica risulta:

$$(N, p, q, h)$$

la chiave privata è invece f; bisogna anche tenere segreto F_p, perché necessario per la decifrazione.

Cifratura e decifrazione

Alice vuole mandare un messaggio m a Bob. Il messaggio m viene rappresentato come un polinomio di grado minore di N, con coefficienti in valore assoluto minori o uguali a $\frac{p-1}{2}$; in seguito il messaggio viene convertito in un messaggio binario con un'opportuna procedura. Da notare che il modo di scegliere il polinomio, il valore di p (spesso viene scelto 3) e la procedura con cui convertire il messaggio sono concordate e note a tutti; l'unica cosa che viene tenuta segreta è la chiave, secondo il principio di Kerchoffs per cui non si può contare sulla segretezza del metodo.
Alice prende quindi (N, p, q, h), chiave pubblica di Bob, sceglie φ polinomio *piccolo* (N.B. La scelta di φ ad ogni cifratura rende il processo non deterministico), calcola e invia a Bob il testo cifrato c

$$c \equiv p\varphi * h + m \pmod{q}$$

Bob riceve c, calcola prima di tutto $a \equiv f * c \pmod{q}$ e ritrova quindi m calcolando

$$m \equiv F_p * a \pmod{p}$$

Correttezza della procedura Dimostriamo che la procedura funziona correttamente, anche se non sempre:

$$\begin{aligned} a &\equiv f * (p\varphi * h + m) && \pmod q \\ \text{(ricordando che } h &\equiv F_q * g \pmod q) \\ &\equiv f * p\varphi * F_q * g + f * m && \pmod q \\ \text{(e dato che } f * F_q &\equiv 1 \pmod q) \\ &\equiv p\varphi * g + f * m && \pmod q \end{aligned}$$

Ora, φ, g, f e m hanno coefficienti piccoli rispetto a q e p è stato scelto molto più piccolo di q. La riduzione modulo q non ha quindi effetto sull'uguaglianza. Si ottiene $a = p\varphi * g + f * m$ e, moltiplicando per F_p e riducendo modulo p:

$$\begin{aligned} F_p * a &= F_p * p\varphi * g + F_p * f * m \\ \text{(ricordando che } F_p * f &\equiv 1 \pmod p) \\ &\equiv 0 + 1 * m \pmod p \\ &= m \end{aligned}$$

L'ultimo passaggio è possibile solo quando i coefficienti di m siano minori di p; questo può non succedere, anche se solitamente la probabilità di ottenere errori è minore di $5 \cdot 10^{-5}$. Questo è un difetto di NTRU di cui va tenuto conto in fase d'implementazione.

Esempio Riportiamo quindi un esempio di applicazione di NTRU con parametri molto piccoli [1] Vengono scelti come parametri $(N, p, q) = (5, 3, 16)$ Si prendono come polinomi nell'anello $\mathbf{R} = \mathbb{Z}[N]/(x^5 - 1)$

$$f = x^4 + x - 1 \quad \text{e} \quad g = x^3 - x$$

Si cercano quindi gli inversi di f modulo 3 e di g modulo 16, rispettivamente F_p e F_q. Si trova:

$$F_p = x^3 + x^2 - 1 \quad \text{e} \quad F_q = x^3 + x^2 - 1$$

Vale infatti

$$F_p * f \equiv (x^3 + x^2 - 1) * (x^4 + x - 1) \equiv 1 \pmod 3$$

e

$$F_q * f \equiv (x^3 + x^2 - 1) * (x^4 + x - 1) \equiv 1 \pmod{16}$$

Si calcola quindi

$$h \equiv F_q * g \equiv -x^4 - 2x^3 + 2x + 1 \pmod{16}$$

La chiave pubblica risulta perciò essere:

$$(N, p, q, h) = (5, 3, 16, -x^4 - 2x^3 + 2x + 1)$$

e la chiave privata $f = x^4 + x - 1$

Alice vuole mandare un messaggio $m = x^2 - x + 1$ a Bob, prende (N, p, q, h), chiave pubblica di Bob, sceglie il polinomio

$$\varphi = x - 1$$

e quindi crea il messaggio cifrato c e lo invia a Bob:

$$c \equiv 3\varphi * h + m \equiv -3x^4 + 6x^3 + 7x^2 - 4x - 5 \pmod{16}$$

(N.B.: Nel contesto di NTRU gli interi, ad esempio, mod 16 si rappresentano con {-7,...,-1,0,1,...,8}, invece che con {0,...,15}. In questa simulazione

inoltre i coefficienti sono diversi da -1,0,+1, come dovrebbero essere, cioè in valore assoluto minori di $\frac{p-1}{2}$*. Questo è soltanto al fine di poter seguire meglio i calcoli)*

Bob decifra calcolando

$$a \equiv f * c \equiv 4x^4 - 2x^3 - 5x^2 + 6x - 2 \pmod{16}$$

e quindi

$$F_p * a \equiv x^2 - x + 1 \pmod{3}$$

riottenendo così il messaggio corretto.

Scelta dei parametri

Numerosi studi su NTRU si concentrano sulla ricerca di una scelta ottimale dei parametri (N, p, q) che garantisca la sicurezza del sistema, senza d'altra parte appesantire troppo il costo computazionale.
Gli autori stessi, all'interno del loro articolo, presentano tre diverse scelte che garantiscono tre diversi livelli di sicurezza:

- **Sicurezza moderata.** $(N, p, q) = (107, 3, 64)$ con una conseguente lunghezza di 340 bit per la chiave privata e 642 bit per la chiave pubblica.

- **Sicurezza elevata.** $(N, p, q) = (167, 3, 128)$ con lunghezza di 530 bit per la chiave privata e 1169 bit per la chiave pubblica.

- **Sicurezza molto elevata.** $(N, p, q) = (503, 3, 256)$ con lunghezza di 1595 bit per la chiave privata e 4024 bit per la chiave pubblica. Si può notare come il livello di sicurezza molto alto venga pagato in termini computazionali: la dimensione delle chiavi ne risente infatti molto.

Attacco del reticolo a NTRU

[1] Vediamo ora l'attacco più semplice che si può portare a NTRU utilizzando i reticoli geometrici, mostrando così la correlazione esistente tra i due, tutt'altro che evidente dalla formulazione del critto-sistema. Questo tipo di approccio chiarisce anche perché SVP è alla base della sicurezza di NTRU.

Dato $h = h_{N-1}x^{N-1} + \cdots + h_0$ (parte della chiave pubblica) si considera la seguente matrice $N \times N$:

$$H = \begin{pmatrix} h_0 & h_1 & \cdots & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \dots & h_0 \end{pmatrix}$$

Se i due polinomi segreti $f = f_{N-1}x^{N-1}+\cdots+f_0$ e $g = g_{N-1}x^{N-1}+\cdots+g_0$ vengono rappresentati mediante vettori riga:

$$\bar{f} = (f_0, \dots, f_{N-1}) \quad \text{e} \quad \bar{g} = (g_0, \dots, g_{N-1})$$

si ha $\bar{f}H = \bar{g} \pmod{q}$.

Si consideri ora la matrice $2N \times 2N$

$$M = \begin{pmatrix} I & H \\ 0 & qI \end{pmatrix}$$

con I matrice identità $N \times N$. Si consideri L, il reticolo geometrico generato dalle righe di M.
Poiché $g \equiv f*h \pmod{q}$ si può scrivere $g = f*h+qy$ per qualche polinomio y.
Se rappresentiamo y come vettore riga N-dimensionale $\bar{y}$, si ha che $(\bar{f}, \bar{y})$ è un vettore riga 2N-dimensionale.

Vale allora che:

$$(\bar{f}, \bar{y}) = M(\bar{f}, \bar{g})$$

e cioè $(\bar{f}, \bar{g})$ appartiene al reticolo L e, avendo f e g coefficienti piccoli, è uno dei vettori corti. I due polinomi segreti f e g, informazione tenuta nascosta in NTRU, sono quindi rappresentati in un vettore corto del reticolo. Un attaccante può quindi applicare algoritmi di riduzione reticolare e provare i vettori così ottenuti; una volta ottenuti f e g il sistema è forzato.
Notiamo adesso che $(\bar{f}, \bar{g})$ è un vettore *corto*, ma *non* necessariamente un vettore di norma minima. L'algoritmo LLL, che appunto trova questi vettori in tempo polinomiale, rappresenta un potente strumento in questo tipo di attacchi.
Per proteggere il sistema da questo tipo di approccio si aumenta N cercando di rendere impraticabile l'utilizzo di LLL.

Capitolo 3

Conclusioni

In definitiva abbiamo visto che il problema di trovare vettori di norma minima (SVP) è un problema ritenuto intrattabile. Una sua approssimazione SVP_γ trova un'ottima risposta con l'algoritmo LLL. Questo, pur avendo un fattore di approssimazione esponenziale dell'ordine di $2^{O(n)}$, nelle applicazioni funziona sorprendentemente bene, ben oltre le aspettative teoriche. Anche dal punto di vista del tempo di esecuzione l'algoritmo fornisce prestazioni particolarmente buone, mediamente molto migliori del tempo stimato in $n^6 \log^3 B$, dove n è la dimensione e B una limitazione sulle lunghezze della base originale.
Abbiamo quindi visto le applicazioni, studiando due crittosistemi basati sui reticoli: in particolare tra i due proposti, il primo, Ajtai-Dwork, è di interesse prettamente teorico perché, per quanto valide le prove matematiche su cui si basa la sua sicurezza, l'algoritmo è piuttosto inefficiente e per poter ottenere delle prestazioni accettabili richiede di tenere la dimensione del reticolo troppo bassa per garantire l'inviolabilità del sistema.
Per contro NTRU è invece molto efficiente e permette di incrementare il livello di sicurezza, aumentando le dimensioni del reticolo, mantenendo tuttavia ottime prestazioni. NTRU grazie alla sua efficienza sarebbe un ottimo candidato alla sostituzione di RSA se la sua sicurezza dovesse essere provata, o quanto meno se dovesse continuare a restare inviolata dopo ulteriori studi.

Appendice A

Complessità computazionale: Problemi P e NP

Più volte nella trattazione si è parlato di problemi *facili* e problemi *difficili*; daremo qui un'introduzione ai concetti di problemi P, NP e NP completi; per una trattazione più completa e formale si faccia riferimento al classico [11].

A.1 P, NP, NP-Completi, NP-Hard

Cominciamo con qualche definizione:
Si definisce **problema** come un generico quesito a cui rispondere per via algoritmica; generalmente ogni problema è presentato con una serie di *parametri*, ovvero variabili dal valore non specificato.
Un problema è descritto fornendo una descrizione di tutti i parametri e una richiesta su quale proprietà debba avere la risposta, ovvero la **soluzione**.
Infine un'*istanza* di un problema si ottiene specificando un valore per tutti i parametri.
Un **algoritmo** è una procedura guidata con le istruzioni passo per passo per poter risolvere un problema. Nella nostra analisi si può pensare a un algoritmo come una serie di operazioni eseguibili da un calcolatore. Un algoritmo

risolve un problema, se applicato a una qualunque *istanza* del problema l'algoritmo porta a una *soluzione*.
Nella scrittura di un algoritmo, una componente molto importante da considerare è la quantità di risorse che questo impiega durante la sua esecuzione; per i nostri scopi considereremo come unica risorsa di riferimento il *tempo*.
L'obiettivo di un programmatore nel risolvere un problema è quello di cercare l'algoritmo più efficiente, ovvero quello che utilizzi meno risorse.
L'efficienza è generalmente misurata in riferimento a un solo parametro n, che rappresenta le dimensioni totali dei dati in input. Si chiama **funzione di complessità** di un algoritmo, con riferimento al tempo, la funzione che ad ogni possibile valore di n associa il tempo massimo che l'algoritmo impiega con input di grandezza n.

Si dice che una funzione $f(n)$ è un $O(g(n))$ se esiste una costante c tale che $|f(n)| \leq c \cdot |g(n)|$ con $n \geq 0$.
Un algoritmo si dice **di ordine polinomiale** (o brevemente *polinomiale*) se la sua *funzione di complessità* è un $O(p(n))$ per qualche funzione polinomiale p. Qualsiasi algoritmo per cui questo non valga è detto **esponenziale**.
Da qui viene la classificazione tra problemi **P** e **NP**: un problema per la cui risoluzione esista un algoritmo polinomiale si dice appartenere alla classe **P**; un problema si dice appartenere alla classe **NP** se, data una candidata soluzione al problema, esiste un algoritmo per verificare in tempo polinomiale che questa è effettivamente una soluzione. Segue facilmente dalla definizione che $P \subset NP$.
ESEMPIO: la fattorizzazione di un numero intero è un problema NP che si ritiene non appartenere a P: non è infatti noto un algoritmo che fattorizzi un intero n in tempo polinomiale, ma dati dei fattori è facile verificare se questi corrispondono effettivamente alla fattorizzazione di n.

Nello studio delle complessità computazionali uno strumento spesso usato è quello della *riduzione algoritmica*. Si dice che un problema **A** viene polinomialmente *ridotto* a un altro problema **B** se si riesce a costruire una

trasformazione che porti ogni istanza del primo problema in una equivalente del secondo. Questo significa anche che qualsiasi algoritmo che risolve **B** risolve anche **A**. Si scrive spesso $\mathbf{B} \leq \mathbf{A}$.
Veniamo ora alla definizione di problema NP-Completo: un problema A è **NP-Completo** se ogni altro problema in NP è polinomialmente riducibile a A. Questo significa anche che ogni algoritmo che risolve A risolve qualsiasi altro problema in NP; in questo senso gli NP-Completi sono i problemi *più difficili* della classe NP. È sorprendente il fatto (dimostrato negli anni '70, vedi [11]) che tali problemi effettivamente esistono e sono conosciuti.
Infine un problema B viene detto **NP-hard** (o NP-difficile) se ogni problema NP-Completo è riducibile a B. I problemi NP-hard sono quindi problemi difficili *almeno quanto* gli NP-completi, ma, nota bene, non necessariamente appartenenti a NP.

A.1.1 P=NP?

Nonostante quanto detto finora non è ancora stato provato che $P \neq NP$; questo quesito è infatti uno dei famosi 7 *Problemi per il millennio* posti nel 2000 dall'istituto matematico Clay ed è tuttora aperto. In particolare, sebbene la maggioranza degli studiosi di teoria della complessità sia portata a credere che i due insiemi siano effettivamente diversi, non è dimostrato che i problemi NP non possano in effetti disporre di un algoritmo polinomiale di risoluzione.
Rivedendo le definizioni date sopra alla luce di ciò, gli NP-Completi sono quindi quelli che *più probabilmente* non stanno in P e, d'altra parte, se si trovasse un algoritmo polinomiale per un problema NP-Completo si avrebbe che $P = NP$.

A.2 NP e crittografia

Da quanto abbiamo visto si capisce facilmente come i problemi NP siano di profondo interesse per la crittografia. C'è da specificare però che la funzione di complessità di un algoritmo misura il tempo del *caso-peggiore*: ovvero, per ogni n grandezza dell'input, esiste almeno un'istanza del problema che impiega quel tempo. Questo, dal punto di vista crittografico, è molto importante: molti problemi NP-Completi o addirittura NP-hard sono a tutti gli effetti inutilizzabili, perché non tutte le istanze, o comunque non abbastanza, sono computazionalmente difficili.

D'altra parte i problemi effettivamente utilizzati in crittografia non è detto siano NP-completi o NP-hard. Il problema della fattorizzazione, ad esempio non è NP-completo [12]. In effetti si ritiene che sia un problema ad un livello intermedio tra P e NP. Naturalmente, se risultasse che P=NP, anche la fattorizzazione risulterebbe un problema di complessità polinomiale.

Ajtai, come già detto, è riuscito invece a dimostrare che SVP è un problema NP-hard [13] il che lo rende ancora più interessante come oggetto di studio.

Bibliografia

[1] Wade Trappe and Lawrence C. Washington, *Introduction to cryptography with coding theory*, second ed., Pearson Prentice Hall, Upper Saddle River, NJ, 2006. MR 2372272 (2008k:94055)

[2] Henri Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138,Springer-Verlag, Berlin, 1993. MR 1228206 (94i:11105)

[3] Oded Regev, *Lattice-based Cryptography*, da Advances in Cryptology - CRYPTO 2006, pp 131-141, serie Lecture Notes in Computer Science, volume 4117, anno 2006.

[4] M. Ajtai, *Generating Hard Instances of Lattice Problems*, In proceedings of the Twenty-Eight Annual ACM Symposium on the Theory of Computing, pp 99-108, ACM, 1996

[5] Lenstra, A.K., Lenstra, Jr., H.W., Lovász, L., *Factoring polynomials with rational coefficients*, Math. Ann. 261(4) (1982) 515–534

[6] M. Ajtai, C. Dwork, *A Public-Key Cryptosystem with Average-Case/Worst-Case Equivalence*, Electronic Colloquium on Computational Complexity TR96-065, disponibile presso http:// www.eccc.uni-trier.de/ eccc-local/Lists/TR- 1996.html

[7] O. Goldreich, S. Goldwasser, and S. Halevi. *Eliminating decryption errors in the Ajtai-Dwork cryptosystem.* In Advances in cryptology, volume 1294 of Lecture Notes in Comput. Sci., pages 105–111. Springer, 1997.

[8] O. Regev. *New lattice-based cryptographic constructions*. Journal of the ACM, 51(6):899–942, 2004. Preliminary version in STOC'03.

[9] Daniele Micciancio, Oded Regev, *Lattice-based Cryptography*, in Post-Quantum Cryptography, 7 novembre 2008

[10] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, *Ntru: A ring-based public key cryptosystem*, Proceedings of the Third International Symposium on Algorithmic Number Theory (London, UK), Springer-Verlag, 1998, First presented at the rump session of Crypto '96, pp. 267–288.

[11] Garey, M. R.; Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences Victor Klee, ed.. W. H. Freeman and Co, USA, 1979.

[12] Scott Aaronson, *PHYS771 Lecture 6: P, NP, and Friends*, disponibile presso http://www.scottaaronson.com/democritus/lec6.html

[13] Miklós Ajtai, *The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract)*, in Proceedings of the thirtieth annual ACM symposium on Theory of computing (Dallas, Texas, United States: ACM, 1998), 10–19. http://portal.acm.org/citation.cfm?id=276705