Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea in Matematica

UN SISTEMA CRITTOGRAFICO: NTRU

Tesi di Laurea in Algoritmi della teoria dei numeri e crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Gian Marco Zanoni

III Sessione
Anno Accademico 2009-2010

Indice

Capitolo 1

NTRU

1.1 Introduzione

NTRU è un crittosistema a chiave pubblica relativamente nuovo. Viene pubblicato per la prima volta nel 1996 da tre matematici: J. Hoffstein, J. Pipher e J.H Silverman, nell'articolo "NTRU: A Ring-Based Public Key Cryptosystem"[HPS98].
NTRU lavora nell'anello $R = \mathbb{Z}[X]/(X^N - 1)$. La cifratura utilizza un sistema basato sia sull'algebra polinomiale che sulla riduzione modulo due numeri, mentre la decifrazione si basa su alcuni concetti della teoria della probabilità. La sicurezza di NTRU deriva non solo dall'interazione dei calcoli fra polinomi e dalle riduzioni modulari, ma soprattutto dalla difficoltá di risolvere SVP (Shortest Vector Problem), ovvero il problema di trovare un vettore di lunghezza minima in un reticolo. Infatti, se la dimensione del reticolo è sufficientemente alta, l'algoritmo LLL, usato comunemente per risolvere SVP, diventa praticamente inutilizzabile[GN07].
NTRU è stato ampiamente studiato negli ultimi vent'anni, ed è risultato molto interessante per via del fatto che occorrono solo $O(N^2)$ operazioni per cifrare e decifrare un messaggio di lunghezza N, mentre per RSA occorrono $O(N^3)$ operazioni. Inoltre, la chiave di NTRU è lunga $O(N)$ mentre solitamente i sitemi crittografici a chiave pubblica che forniscono simili prestazioni

utilizzano chiavi lunghe $O(N^2)$.
L'algoritmo NTRU possiede un'ulteriore vantaggio: attualmente, lo si ritiene immune dagli attacchi effettuati mediante computer quantistici; tuttavia in questo ambito non sono ancora stati effettuati studi approfonditi.
A sottolineare quanto di positivo detto in precedenza, nel 2009 l'Institute of Electrical and Electronic Engineers (IEEE) ha riconosciuto all'algoritmo NTRU la certificazione standard **IEEE 1363.1**, definendolo la maggiore innovazione nell'ambito dei sistemi crittografici a chiave pubblica degli ultimi vent'anni.

1.2 Algoritmo

1.2.1 Notazioni

Il sistema NTRU dipende da tre parametri interi (N, p, q), e da quattro insiemi $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_\varphi$ e $\mathcal{L}_m$ di polinomi di grado $N-1$ a coefficienti interi. In seguito verrà approfondita l'esatta composizione di questi insiemi.
Il crittosistema lavora nell'anello $R = \mathbb{Z}[X]/(X^N - 1)$. Ogni elemento $F \in R$ si può scrivere in forma polinomiale o vettoriale nel modo seguente

$$F = \sum_{i=0}^{N-1} F_i x^i = (F_0, F_1, \ldots, F_{N-1})$$

Dati i polinomi $f, g \in R$

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

si definisce un operazione $*$ in R

$$h = f * g$$

dove

$$h = c_{N-1}X^{N-1} + \ldots + c_0$$

con

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

Si noti che l'operazione $f * g$, che è proprio il prodotto nel quoziente, richiede N^2 moltiplicazioni; tuttavia, nelle classiche implementazioni di NTRU, o f o g possiedono coeffienti piccoli, cosicchè l'operazione diventa molto veloce [TW06].

1.2.2 Creazione della chiave

Consideriamo la situazione in cui Alice vuole mandare un messaggio a Bob; pertanto quest'ultimo deve creare, e distribuire, la propria chiave pubblica. Bob sceglie tre interi N, p e q, con $MCD(p, q) = 1$ e p molto piú piccolo di q.
Bob sceglie poi due polinomi segreti f e g, controllando che esistano polinomi F_p e F_q tali che:

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

Si noti che, con i parametri scelti, gli inversi F_p e F_q esistono per quasi tutte le possibili scelte di f; inoltre, risulta semplice trovarli applicando l'algoritmo euclideo leggermente modificato. Poi Bob calcola

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

la chiave pubblica di Bob è

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

mentre la chiave privata è f, nonostante Bob debba conservare segretamente anche F_p, perché gli servirà in seguito per la decifrazione.

1.2.3 Cifratura

Una volta che Bob ha costruito la chiave, Alice puó inviare il suo messaggio. Il messaggio viene rappresentato, mediante una procedura concordata, ma non segreta, come un polinomio m di grado minore di N, con coefficienti

in valore assoluto minori o uguali a $\frac{p-1}{2}$. Siccome in quasi tutte le implementazioni di NTRU p è uguale a 3, il messaggio è rappresentato mediante coefficienti uguali a 0, +1 o −1. In seguito, con opportuni algoritmi, viene convertito in un normale messaggio binario.
A questo punto Alice sceglie un polinomio "piccolo" φ e calcola il testo cifrato e

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

poi lo invia a Bob.
Si noti che la scelta di φ viene effettuata ogni volta che viene inviato un messaggio m. Questo fa in modo che la cifratura sia non deterministica, condizione necessaria affinché un sistema sia sicuro.

1.2.4 Decifrazione

Bob inizia la decifrazione calcolando

$$a \equiv f * e \pmod{q}$$

con i coefficienti di a di valore assoluto al piú $\frac{q}{2}$, e trova m calcolando

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

.

1.2.5 Perché la procedura di decifrazione funziona quasi sempre

La decifrazione funziona poiché si ha:

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

Ora, poiché φ, g, f e m hanno coefficienti "piccoli", quasi sempre compresi tra $-\frac{q}{2}$ e $+\frac{q}{2}$, e p è molto più piccolo di q, riducendo modulo q l'equazione non cambia. Si ha quindi:

$$a = p\varphi * g + f * m$$

Infine, moltiplicando a per F_p e riducendo modulo p si ottiene proprio il messaggio originale:

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

Poiché i coefficienti di m sono minori di p. Può però succedere che i coefficienti siano più grandi di p; in questo caso la procedura non funziona correttamente. Tuttavia, con i parametri che si usano solitamente, la probabilità di ottenere errori nella decifrazione è minore di $5 \cdot 10^{-5}$. Questo può non essere accettabile nelle applicazioni, e deve essere risolto a llivello superiore, dalle applicazioni che implementano NTRU.

1.3 Scelta dei parametri

1.3.1 Scelta dei polinomi

NTRU lavora con un insieme di polinomi con coefficienti piccoli, è quindi conveniente introdurre la seguente notazione:

$$\mathcal{L}(d_1, d_2) = \begin{cases} \text{l'insieme di polinomi di grado} < N \\ \text{con } d_1 \text{ coefficienti uguali a } +1 \\ d_2 \text{ coefficienti uguali a } -1 \\ \text{e i restanti coefficienti uguali a } 0 \end{cases}$$

Con questa notazione, una volta scelti tre interi positivi d_f, d_g e d_φ, possiamo definire i tre seguenti insiemi:

$$\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1) \qquad \mathcal{L}_g = \mathcal{L}(d_g, d_g) \qquad \mathcal{L}_\varphi = \mathcal{L}(d_\varphi, d_\varphi)$$

Definiti questi insiemi scegliamo

$$f \in \mathcal{L}_f \qquad g \in \mathcal{L}_g \qquad \varphi \in \mathcal{L}_\varphi$$

f ha un numero differente di coefficienti uguali a $+1$ e -1 per permetterne l'invertibilità, sapendo che un polinomio che soddisfa $f(1) = 0$ non può essere invertibile.
Per quanto riguarda l'insieme dei messaggi possibili, invece, conviene considerare

$$\mathcal{L}_m = \{m \in R |\ m \text{ ha coefficienti compresi fra } -\frac{1}{2}(p-1) \text{ e } +\frac{1}{2}(p-1)\}$$

Siccome quasi in tutte le implementazioni di NTRU $p = 3$, i coefficienti dei polinomi appartenenti all'insieme dei messaggi $\mathcal{L}_m$ saranno solamente $+1$, -1 e 0.
Con questi parametri, f, g e φ sono sufficientemente piccoli affinché la decifrazione funzioni con probabilità molto alta.

1.3.2 Scelta dei parametri (N, p, q)

Gli studi sulla sicurezza di NTRU portano ad una costante ricerca su quale sia la miglior scelta per i parametri (N, p, q) affinché sia garantita la riservatezza del messaggio, senza incrementare troppo i tempi di calcolo, ricordando che una delle maggiori qualità di NTRU è proprio la velocità.
Nel loro articolo del 1996 Hoffstein, Pipher e Silverman presentano tre diverse scelte per i parametri, ognuna delle quali garantisce un diverso livello di sicurezza[HPS98].

Sicurezza moderata

La prima scelta dei parametri è quella che garantisce minore sicurezza:

$$(N, p, q) = (107, 3, 64)$$

Si noti che vengono rispettate le condizioni richieste nella scelta di p e q, in quanto p è molto più piccolo di q e $MCD(p, q) = 1$.
Con questa scelta di (N, p, q), la scelta di d_f, d_g e d_φ più conveniente è

$$(d_f, d_g, d_\varphi) = (15, 12, 5)$$

e quindi

$$\mathcal{L}_f = \mathcal{L}(15, 14) \qquad \mathcal{L}_g = \mathcal{L}(12, 12) \qquad \mathcal{L}_\varphi = \mathcal{L}(5, 5)$$

ovvero f viene scelto con 15 coefficienti uguali a $+1$, 14 uguali a -1 e i restanti 78 coefficienti uguali a 0, g con 12 coefficienti uguali a $+1$ e 12 uguali a -1, infine φ ha 5 coefficienti uguali a $+1$ e 5 uguali -1.
La scelta di questi parametri comporta, utilizzando la codifica dei coefficienti prevista dalle implementazioni che utilizzano NTRU, i seguenti costi in termini computazionali

$$\text{Chiave privata} = 340\ bits \qquad \text{Chiave pubblica} = 642\ bits$$

Sicurezza elevata

La seconda scelta di parametri ci garantisce maggiore sicurezza:

$$(N, p, q) = (167, 3, 128)$$

inoltre si sceglie

$$(d_f, d_g, d_\varphi) = (61, 20, 18)$$

e di conseguenza sarà

$$\mathcal{L}_f = \mathcal{L}(61, 60) \qquad \mathcal{L}_g = \mathcal{L}(20, 20) \qquad \mathcal{L}_\varphi = \mathcal{L}(18, 18)$$

Questo comporta che le chiavi avranno le seguenti dimensioni

$$\text{Chiave privata} = 530\ bits \qquad \text{Chiave pubblica} = 1169\ bits$$

Sicurezza molto elevata

L'ultima scelta di parametri proposta dai tre autori è quella che prevede la massima sicurezza

$$(N, p, q) = (503, 3, 256)$$

$$(d_f, d_g, d_\varphi) = (216, 72, 55)$$

e quindi

$$\mathcal{L}_f = \mathcal{L}(216, 215) \qquad \mathcal{L}_g = \mathcal{L}(72, 72) \qquad \mathcal{L}_\varphi = \mathcal{L}(55, 55)$$

La sicurezza con questi valori dei parametri ï¿½ molto elevata, ma questo ovviamente incide sulla dimensione delle chiavi

$$\text{Chiave privata} = 1595\ bits \qquad \text{Chiave pubblica} = 4024\ bits$$

Ricerca nella scelta dei parametri

Negli ultimi anni, gli articoli di ricerca sul sistema NTRU si stanno concentrando sulla miglior scelta possibile per i parametri; in particolar modo, la ricerca dei parametri prosegue di pari passo con la scoperta di nuovi attacchi

da parte dei crittoanalisti. Gli studi più recenti si limitano a considerare i parametri N e d_f dai quali è possibile ricavare tutti gli altri [HGHPW05]. La scelta dei parametri proposta per ottenere un livello di sicurezza di $256-bit$ è

$$(N, d_f) = (1087, 120)$$

1.3.3 Un criterio per la decifrazione

Definiamo *lunghezza* di un elemento $F \in R$ la seguente norma

$$\|F\|_\infty = \max_{0\leq i\leq N-1}\{F_i\} - \min_{0\leq i\leq N-1}\{F_i\}$$

Inoltre defininiamo su R una norma centrata L^2 nel modo seguente:

$$\|F\|_2 = (\sum_{i=0}^{N-1}(F_i - \overline{F})^2)^{\frac{1}{2}} \qquad \text{con} \qquad \overline{F} = \frac{1}{N}\sum_{i=0}^{N-1} F_i$$

Definito ciò, posso enunciare la seguente proposizione:

Proposizione 1.3.1.
$\forall\, \epsilon > 0\ \exists\, \gamma_1,\ \gamma_2 > 0$, *dipendenti da* ϵ *e* N, *tali che presi* $F, G \in R$ *la probabilità che essi soddisfino*

$$\gamma_1\|F\|_2\|G\|_2 \leq \|F * G\|_\infty \leq \gamma_2\|F\|_2\|G\|_2$$

è maggiore di $1-\epsilon$

Questa proposizione è stata verificata sperimentalmente per un ampio numero di parametri[HPS98], ma, a quanto sembra, non è stata dimostrata. Ora, si noti che per far funzionare la decifrazione occorre che

$$\|f * m + p\varphi * g\|_\infty < q$$

Ciò è praticamente sempre vero se i parametri vengono scelti in modo tale che sia:

$$\|f * m\|_\infty \leq \frac{q}{4} \qquad \text{e} \qquad \|p\varphi * g\|_\infty \leq \frac{q}{4}$$

e quindi per la proposizione, 1.3.1, dovremo scegliere

$$\|f\|_2\|m\|_2 \approx \frac{q}{4\gamma_2} \qquad \text{e} \qquad \|\varphi\|_2\|g\|_2 \approx \frac{p}{4\gamma_2}$$

con γ_2 corrispondente ad un valore piccolo di ϵ.
Ad esempio, sperimentalmente, si ottiene che per $N = 107$, $N = 167$ e $N = 503$ i valori corrispondenti di γ_2 sono rispettivamente 0.35, 0.27 e 0.17 [HPS98].

1.4 Esempio

Mostro ora un esempio di cifratura e decifrazione con NTRU utilizzando parametri molto piccoli che non garantiscono, ovviamente, la sicurezza, ma permettono di seguire esplicitamente i calcoli effettuati[TW06].
Si scelgono innanzittutto i parametri

$$(N, p, q) = (5, 3, 16)$$

poi Bob sceglie i suoi due polinomi segreti dall'anello $R = \mathbb{Z}[N]/(X^5 - 1)$

$$f = X^4 + X - 1 \qquad \text{e} \qquad g = X^3 - X$$

Bob deve ora trovare i polinomi F_p e F_q, che sono

$$F_p = X^3 + X^2 - 1 \qquad F_q = X^3 + X^2 - 1$$

poichè

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

e

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

Bob calcola ora h

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

quindi la chiave pubblica di Bob è

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

invece la chiave privata è

$$f = X^4 + X - 1$$

Ora tocca ad Alice, che deve inviare il messaggio

$$m = X^2 - X + 1$$

Alice sceglie il polinomio

$$\varphi = X - 1$$

e crea quindi il messaggio cifrato e

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

e lo invia a Bob. Si noti qui che e ha coefficienti diversi da 0 e ± 1. A questo punto Bob inizia la decifrazione calcolando dapprima

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

e poi

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

quindi Bob ha ottenuto il messaggio corretto.

Capitolo 2

Reticoli

2.1 Introduzione ai reticoli

Prima di introdurre la definizione di reticolo, bisogna ricordare che presi $v_1, \ldots, v_n$ vettori linearmente indipendenti di $\mathbb{R}^n$, possiamo scrivere ogni vettore $v \in \mathbb{R}^n$ nella forma:

$$v = a_1 v_1 + \ldots + a_n v_n$$

per $a_1, \ldots, a_n \in \mathbb{R}$ univocamente determinati da v. Noto questo posso definire un reticolo nel modo seguente:

Definizione 2.1 (Reticolo)**.**
Il **reticolo** generato da $v_1, \ldots, v_n$ è l'insieme dei vettori della forma

$$m_1 v_1 + \ldots + m_n v_n$$

dove $m_1, \ldots, m_n \in \mathbb{Z}$.

Per ogni reticolo così definito possiamo definire il concetto di base:

Definizione 2.2 (Base del reticolo)**.**
Dato un reticolo L generato dai vettori $v_1, \ldots, v_n$ l'insieme $\{v_1, \ldots v_n\}$ è una

base del reticolo L.

Possiamo subito osservare che:

Osservazione 1.
Un reticolo possiede infinite basi.

Introduciamo inoltre la nozione di lunghezza e determinante:

Definizione 2.3 (Lunghezza di un vettore)**.**
Si definisce **lunghezza** di un vettore $v = (x_1, \ldots, x_n)$:

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

Definizione 2.4 (Determinante del reticolo)**.**
Si dice **determinante** del reticolo generato dai vettori $\{v_1, \ldots, v_n\}$, il numero

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

Osservazione 2.
Si osserva che il determinante è indipendente dalla base scelta.

Il problema che interessa la crittografia è la ricerca di un vettore non nullo di lunghezza minima in un reticolo. Tale problema è molto difficile se la dimensione del reticolo è grande. Esistono tuttavia alcuni metodi di riduzione reticolare che funzionano bene quando la dimensione è bassa; mostriamo ora come funzionano.

2.2 Riduzione reticolare

2.2.1 Caso bidimensionale

Siano v_1 e v_2 i vettori di una base di un reticolo L bidimensionale. Mostriamo ora come sostituire questa base con una base nuova avente vettori "quasi" ortogonali.[TW06]
Innanzitutto, controllo se $\|v_1\| \leq \|v_2\|$ altrimenti li scambio. Nel processo di ortogonalizzazione di Gram-Schmidt si sostituisce v_2 col vettore:

$$v_2^* = v_2 - \frac{v_1 \cdot v_2}{v_1 \cdot v_1} v_1$$

che è ortogonale a v_1.
Tuttavia, v_2^* può non appartenere al reticolo L. Devo quindi sostituire v_2 non con v_2^* ma con $v_2 - tv_1$, dove t è l'intero più vicino a $\frac{v_1 \cdot v_2}{v_1 \cdot v_1}$, scegliendo per definizione 0 come intero più vicino a $\pm\frac{1}{2}$, ± 1 come l'intero più vicino a $\pm\frac{3}{2}$ e così via.
Di conseguenza, posso ora cambiare la base $\{v_1, v_2\}$ con la base

$$\{v_1, v_2 - tv_1\}$$

e poi ripetere il procedimento con la nuova base, fino a quando non otterrò $t = 0$, avendo definito una base ridotta nel modo seguente.

Definizione 2.5 (Base ridotta)**.**
Una base $\{v_1, v_2\}$ di un reticolo bidimensionale si dice ridotta se:

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

Ecco un esempio di riduzione reticolare nel caso bidimensionale:

Esempio 2.1.
Considero la base $\{v_1, v_2\}$ del reticolo L di dimensione 2 formata dai vettori $v_1 = (24, 17)$ e $v_2 = (30, 21)$. Applico il metodo di riduzione per ricavare una

base ridotta del reticolo L.

Poiché $\|v_1\| \leq \|v_2\|$ non devo scambiare i vettori tra loro. Calcolo:

$$\frac{v_1 \cdot v_2}{v_1 \cdot v_1} = \frac{1077}{865}$$

quindi $t = 1$, la nuova base sarà dunque:

$$\{v_1', v_2'\} = \{v_1, v_2 - v_1\} = \{(24, 17), (6, 4)\}$$

Ora siccome $\|v_2'\| \leq \|v_1'\|$ li scambio tra loro ottenendo la base formata dai vettori

$$v_1' = (6, 4) \quad \text{e} \quad v_2' = (24, 17)$$

Proseguo calcolando

$$\frac{v_1' \cdot v_2'}{v_1' \cdot v_1'} = \frac{212}{52}$$

Si ha, quindi, $t = 4$ e quindi la nuova base sarà:

$$\{v_1'', v_2''\} = \{v_1', v_2' - 4v_1'\} = \{(6, 4), (0, 1)\}$$

li scambio tra loro e continuo a calcolare

$$\frac{v_1'' \cdot v_2''}{v_1'' \cdot v_1''} = \frac{4}{1}$$

di conseguenza $t = 4$, da cui ricavo la base

$$\{v_1''', v_2'''\} = \{(0, 1), (6, 0)\}$$

Ora, siccome

$$\|v_1'''\| \leq \|v_2'''\| \quad \text{e} \quad \frac{v_1''' \cdot v_2'''}{v_1''' \cdot v_1'''} = 0$$

la base $\{v_1''', v_2'''\}$ è ridotta. Inoltre, il primo vettore della base ridotta v_1''' è un vettore di lunghezza minima del reticolo.

Mostro ora, con il seguente teorema, che questo algoritmo porta sempre ai medesimi risultati.

Teorema 2.2.1.
Sia $\{v_1, v_2\}$ una base di un reticolo bidimensionale in $\mathbb{R}^2$. Si consideri l'algoritmo seguente:

1. *Se $\|v_1\| > \|v_2\|$, scambiare v_1 con v_2 in modo che $\|v_1\| \leq \|v_2\|$.*
2. *Porre t uguale all'intero più vicino a $\frac{v_1 \cdot v_2}{v_1 \cdot v_1}$.*
3. *Se $t = 0$, termina l'algoritmo. Se $t \neq 0$, sostituire v_2 con $v_2 - tv_1$ e tornare al punto 1.*

L'algoritmo termina in un numero finito di passi e produce una base ridotta $\{v_1^r, v_2^r\}$ del reticolo.
Il vettore v_1^r è un vettore non nullo di lunghezza minima del reticolo.

Dimostrazione.
Per prima cosa dimostro che l'algoritmo termina sempre.
Sia $\mu = \frac{v_1 \cdot v_2}{v_1 \cdot v_1}$ e sia $v_2^* = v_2 - \mu v_1$. Allora

$$v_2 - tv_1 = v_2^* + (\mu - t)v_1$$

Ora, poichè v_1 e v_2^* sono ortogonali, per il teorema di Pitagora, posso affermare che:

$$\|v_2 - tv_1\|^2 = \|v_2^*\|^2 + \|(\mu - t)v_1\|^2 = \|v_2^*\|^2 + (\mu - t)^2\|v_1\|^2$$

Inoltre, poiché $v_2 = v_2^* + \mu v_1$, si ha

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

Ora, se $-\frac{1}{2} \leq \mu \leq +\frac{1}{2}$, allora $t = 0$ e $\mu - t = \mu$. Altrimenti $|\mu - t| \leq \frac{1}{2} < |\mu|$. Quindi, se $t \neq 0$, si ha $|\mu - t| < |\mu|$, da cui si ottiene che

$$\|v_2 - tv_1\|^2 = \|v_2^*\|^2 + (\mu_t)^2\|v_1\|^2 < \|v_2^*\|^2 + \mu^2\|v_1\|^2 = \|v_2\|^2$$

Dunque $\|v_2 - tv_1\|^2 < \|v_2\|^2$. Così se il processo continuasse senza mai dare una base ridotta, le lunghezze dei vettori diminuerebbero indefinitamente ma, poichè esiste solo un numero finito di vettori del reticolo che sono più corti di v_2, le lunghezze non possono continuare a diminuire e quindi alla fine si ottiene una base ridotta.
Dopo aver mostrato che l'algoritmo termina in un tempo finito, devo mostrare che il vettore v_1 di una base ridotta è un vettore non nullo di lunghezza minima per il reticolo.
Considero un qualunque vettore non nullo del reticolo $av_1 + bv_2$ con $a, b \in \mathbb{Z}$. Allora

$$\|av_1 + bv_2\|^2 = (av_1 + bv_2) \cdot (av_1 + bv_2) = a^2\|v_1\|^2 + 2abv_1 \cdot v_2 + b^2\|v_2\|^2$$

Poiché $\{v_1, v_2\}$ è ridotta, si ha

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

e quindi

$$2abv_1 \cdot v_2 \geq -|ab|\|v_1\|^2$$

Pertanto, dato che $\|v_1\|^2 \leq \|v_2\|^2$, si ha:

$$\begin{aligned}
\|av_1 + bv_2\|^2 &= (av_1 + bv_2) \cdot (av_1 + bv_2) \\
&= a^2\|v_1\|^2 + 2abv_1 \cdot v_2 + b^2\|v_2\|^2 \\
&\geq a^2\|v_1\|^2 - |ab|\|v_1\|^2 + b^2\|v_2\|^2 \\
&\geq a^2\|v_1\|^2 - |ab|\|v_1\|^2 + b^2\|v_1\|^2
\end{aligned}$$

Quindi

$$\|av_1 + bv_2\|^2 \geq (a^2 - |ab| + b^2)\|v_1\|^2$$

ma $a^2 - |ab| + b^2$ è un intero, che possiamo scrivere nella forma

$$(|a| - \frac{1}{2}|b|)^2 + \frac{3}{4}|b|^2$$

cosicché esso è non negativo ed è uguale a 0 se e solo se $a = b = 0$. Poiché $av_1 + bv_2 \neq 0$, deve essere che $a^2 - |ab| + b^2 \geq 1$. Quindi

$$\|av_1 + bv_2\|^2 \geq \|v_1\|^2$$

ossia v_1 è un vettore non nullo minore di ogni vettore del reticolo, quindi è un vettore di lunghezza minima.

□

2.2.2 Caso n-dimensionale

Dopo aver analizzato il caso bidimensionale richiamo il procedimento di ortogonalizzazione di Gram-Schmidt nel caso di una base n-dimensionale.[Coh93]

Proposizione 2.2.2 (Gram-Schmidt)**.**
Sia v_i un elemento della base $\{v_1, \ldots, v_n\}$ di uno spazio euclideo E. Definiamo per induzione:

$$v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^* \quad per \quad 1 \leq i \leq n$$

con

$$\mu_{i,j} = \frac{v_i \cdot v_j^*}{v_j^* \cdot v_j^*} \quad per \quad 1 \leq j < i \leq n$$

Allora i v_i^* formano una base ortogonale (ma non necessariamente ortonormale) di E.

Questa proposizione si dimostra per induzione.

2.3 Algoritmo LLL

La riduzione reticolare in dimensioni maggiori di due è molto più difficile. Uno degli algoritmi di maggiore successo è stato ideato nel 1982 da A. Lenstra, H. Lenstra e L. Lovász e per questo viene chiamato algoritmo *LLL*.[LLL82, Lov86]
L'algoritmo *LLL* non cerca obbligatoriamente il vettore di lunghezza minima, ma vettori corti nel reticolo. Infatti, per forzare NTRU, non sempre serve il vettore minimo, ne basta uno sufficientemente corto.
Sia $\{v_1, \ldots, v_n\}$ una base del reticolo L. Utilizzando il processo di ortogonalizzazione di Gram-Schmidt mostrato dalla proposizione 2.2.2 possiamo trovare una base ortogonale $\{v_1^*, \ldots, v_n^*\}$; dopodiché possiamo introdurre una nuova definizione di base ridotta:

Definizione 2.6 (Base *LLL*-ridotta)**.**
La base $\{v_1, \ldots, v_n\}$ si dice ***LLL*-ridotta** se:

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

e

$$\|v_i^* + \mu_{i,i-1} v_{i-1}^*\|^2 \geq \frac{3}{4} \|v_{i-1}^*\|^2 \quad \text{per} \quad 1 < i \leq n$$

Possiamo allora introdurre il seguente teorema [TW06, Coh93]:

Teorema 2.3.1. *Sia $\{v_1, \ldots, v_n\}$ una base LLL-ridotta del reticolo L, sia λ la lunghezza di un vettore non nullo di lunghezza minima in L, e sia D il determinante del reticolo. Allora:*

1. $\|v_1\| \leq 2^{\frac{n-1}{4}} D^{\frac{1}{n}}$
2. $\|v_1\| \leq 2^{\frac{n-1}{2}} \lambda$

3. $\|v_1\| \dots \|v_n\| \leq 2^{\frac{n(n-1)}{4}} \lambda$

Si vede quindi che il vettore v_1 in una base LLL-ridotta, anche se non è di lunghezza minima, è sufficientemente corto affinché si possa utilizzarlo, almeno fino a quando la dimensione n è piccola.
Mostro ora come funziona l'algoritmo, il quale trasforma i vettori v_i di una base qualunque in una base LLL-ridotta,e inoltre, fornisce la matrice del cambio di base H.[Coh93]

Algoritmo 2.3.2 (LLL)**.**
Sia data una base $\{v_1, \dots, v_n\}$ *di un reticolo* L

1 - Inizializzazione *Inizializzo* $k \leftarrow 2$, $k_{max} \leftarrow 1$, $v_1^* \leftarrow v_1$, $V_1 \leftarrow v_1 \cdot v_1$ *e* $H \leftarrow I_n$. *Con* I_n *matrice identica* $n \times n$

2 - Gram-Schmidt *Se* $k \leq k_{max}$ *vai al punto 3. Altrimenti, pongo* $k_{max} \leftarrow k$, $v_k^* \leftarrow v_k$*; poi, per* $j = 1, \dots, k-1$, *pongo* $\mu_{k,j} \leftarrow \frac{v_k \cdot v_j^*}{V_j}$ *e* $v_k^* \leftarrow v_k^* - \mu_{k,j} v_j^*$. *Infine, pongo* $V_k \leftarrow v_k^* \cdot v_k^*$*; se* $V_k = 0$ *do in output un messaggio d'errore che dice che i* v_i *non formano una base e termina l'algoritmo.*

3 - Test delle LLL**-condizioni** *Eseguo il sotto-algoritmo* $RED(k, k-1)$. *Se* $V_k < (0,75 - \mu_{k,k-1}^2) V_{k-1}$, *eseguo il sotto-algoritmo* $SWAP(k)$, *pongo* $k \leftarrow max(2, k-1)$ *e vado al punto 3. Altrimenti, per* $l = k-2, k-3, \dots, 1$ *eseguo il sotto -algoritmo* $RED(k, l)$, *poi pongo* $k \leftarrow k+1$.

4 - Fine *Se* $k \leq n$ *vado al punto 2. Altrimenti, do come output la base* LLL*-ridotta formata dai* b_i *e la matrice* $H \in GL_n(\mathbb{Z})$ *e termino l'algoritmo.*

Nell'algoritmo LLL ho utilizzato due sotto-algoritmi ecco come funzionano:

Algoritmo 2.3.3 $(RED(k,l))$**.**

Se $\|\mu_{k,l}\| \leq 0.5$ *termina il sotto-algoritmo* RED*. Altrimenti, sia* q *l'intero più vicino a* $\mu_{k,l}$*, ovvero:*

$$q \leftarrow \lfloor \mu_{k,l} \rceil = \lfloor 0.5 + \mu_{k,l} \rfloor^{1}$$

Pongo $v_k \leftarrow v_k - qv_l$*,* $H_k \leftarrow H_k - qH_l$*,* $\mu_{k,l} \leftarrow \mu_{k,l} - q$*. Poi, per ogni* i *tale che* $1 \leq i \leq l-1$ *pongo* $\mu_{k,i} \leftarrow \mu_{k,i} - q\mu_{k,i}$ *e termino il sotto-algoritmo.*

Algoritmo 2.3.4 $(SWAP(k))$**.**
Scambia i vettori v_k *e* v_{k-1}*, poi scambia* H_k *e* H_{k-1} *e se* $k > 2$ *scambia, per ogni* j *tale che* $1 \leq j \leq k-2$*,* $\mu_{k,j}$ *con* $\mu k-1, j$*. Poi poni, nell'ordine,* $\mu \leftarrow \mu_{k,k-1}$*,* $V \leftarrow V_k + \mu^2 V_{k-1}$*,* $\mu_{k,k-1} \leftarrow \frac{\mu V_{k-1}}{V}$*,* $v \leftarrow v^*_{k-1}$*,* $v^*_{k-1} \leftarrow v^*_k + \mu v$*,* $v^*_k \leftarrow -\mu_{k,k-1}v^*_k + \frac{V_k}{V}v$*,* $V_k \leftarrow \frac{V_{k-1}V_k}{V}$ *e* $V_{k-1} \leftarrow V$*. Infine, per* $i = k+1, k+2, \ldots, k_{max}$ *porre* $t \leftarrow \mu i, k$*,* $\mu_{i,k} \leftarrow \mu_{i,k-1} - \mu t$*,* $\mu_{i,k-1} \leftarrow t + \mu_{k,k-1}\mu_{i,k}$ *, poi terminare l'algoritmo.*

L'algoritmo RED serve per testare le condizoni necessarrie affinché la base sia LLL-ridotta, mentre l'algoritmo $SWAP$ serve per scambiare i vettori. Nel capitolo successivo vedremo come LLL possa essere usato per attaccare NTRU.

[1] dove $\lfloor x \rfloor$ è la funzione parte intera inferiore di x

Capitolo 3

Attacchi a NTRU

3.1 Introduzione

Come per ogni algoritmo di cifratura, sin dalla presentazione di NTRU i crittoanalisti si sono messi al lavoro per cercare di forzarlo. Principalmente gli attacchi portati ad NTRU sono basati sulla riduzione reticolare. Mostrerò infatti che se sapessi risolvere SVP potrei ricavare facilmente le chiavi private f e g. Il miglior algoritmo conosciuto per effettuare la riduzione reticolare è LLL; tuttavia, se si sceglie N sufficientemente grande, gli attacchi con LLL sono esponenziali.[NS01, HPS98]
La sicurezza di NTRU ha ricevuto un ulteriore garanzia quando, nel 1997, Ajtai ha dimostrato che il problema SVP è NP-hard.[NS01]
Oltre che con gli attacchi basati sulla teoria dei reticoli, i crittoanalisti hanno provato a forzare NTRU con degli attacchi del tipo chosen-ciphertext o meet-in-the-middle. In seguito a questi attacchi la prima versione di NTRU è stata modificata; inoltre NTRU, come RSA, per evitare questi attacchi e funzionare correttamente, necessita di processi di precompilazione.[NS01]

3.2 Attacco a forza bruta

Il primo attacco che si può portare a NTRU è un attacco a forza bruta. Consiste nel provare tutte le chiavi possibili e vedere quali danno un testo decifrato di senso compiuto, nell'ipotesi che il testo in chiaro sia dotato di senso. In particolare un attaccante di NTRU può provare a recuperare la chiave privata controllando, per tutte le possibili $f \in \mathcal{L}_f$, se il risultato di $f*h \pmod q$ è piccolo; in alternativa, può controllare per ogni $g \in \mathcal{L}_g$, $g*h^{-1} \pmod q$. Analogamente, l'attaccante può cercare di recuperare il messaggio testando tutti i possibili $\varphi \in \mathcal{L}_\varphi$ e verificando se $e - \varphi * h \pmod q$ dà un risultato piccolo. In pratica $\mathcal{L}_g$ è molto più piccolo di $\mathcal{L}_f$, quindi la sicurezza della chiave privata è determinata dalla cardinalità di $\mathcal{L}_g$ ($\#\mathcal{L}_g$), mentre la sicurezza del messaggio è determinata da $\#\mathcal{L}_\varphi$.

3.3 Attacco basato sui reticoli

La maggior parte degli attacchi ad NTRU si basano sulla teoria dei reticoli. Il primo attacco proposto è il più elementare, ma spiega chiaramente come la ricerca del vettore più piccolo e l'algoritmo LLL possono aiutare a forzare NTRU[TW06]. L'attacco funziona nel modo seguente:
dato $h = h_{N-1}X^{N-1} + \ldots + h_0$ si considera la matrice circolante $N \times N$

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

Rappresentiamo ora $f = f_{N-1}X^{N-1} + \ldots + f_0$ e $g = g_{N-1}X^{N-1} + \ldots + g_0$ mediante i vettori riga

$$\overline{f} = (f_0, \ldots, f_{N-1}) \qquad \text{e} \qquad \overline{g} = (g_0, \ldots, g_{N-1})$$

si ha:

$$\overline{f}H \equiv \overline{g} \pmod{q}$$

allora si considera la matrice $2N \times 2N$ composta da 4 blocchi $N \times N$

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

dove I è la matrice identità $N \times N$.
Sia L il reticolo generato dalle righe di M. Poiché $g \equiv f * h \pmod{q}$, si può chiaramente scrivere $g = f * h + qy$, per un certo polinomio y, rappresentato anche esso come vettore riga N-dimensionale $\overline{y}$. Di conseguenza $(\overline{f}, \overline{y})$ è un vettore riga $2N$-dimensionale. Quindi

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

da cui segue che $(\overline{f}, \overline{g})$ appartiene al reticolo L, e poiché f e g hanno coefficienti piccoli, $(\overline{f}, \overline{g})$ è un vettore piccolo del reticolo L. Quindi la chiave segreta può essere scoperta da un attaccante applicando un algoritmo di riduzione reticolare per trovare vettori piccoli nel reticolo L, dato che $(\overline{f}, \overline{g})$ sarà uno di questi. Una volta che l'attaccante trova f e g il sistema è violato. Chiaramente la difesa utilizzata nei confronti di questo tipo di attacco basato sui reticoli, consiste nell'incrementare la dimensione N, affinchè gli algoritmi di riduzione reticolare diventino inefficaci. Tuttavia, si cerca di non aumentare troppo N per non rallentare eccessivamente gli algoritmi di cifratura e decifrazione.

3.4 Introduzione del parametro α

I metodi di riduzione reticolare hanno maggiore successo quando il vettore di lunghezza minima cercato è piccolo in confronto alla $2N$-esima radice del determinante del reticolo[TW06], quindi l'attacco mostrato in 3.3 può essere

migliorato sostituendo I nel blocco in alto a sinistra della matrice M con αI, con $\alpha \in \mathbb{R}$ opportunamente scelto, così la matrice diventa

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

e il risultante vettore corto del reticolo, $\tau = (\alpha\overline{f}, \overline{g})$ sarà relativamente più piccolo e quindi più facile da trovare.
Gli stessi autori di NTRU propongono come scegliere α per poter ottimizzare gli algoritmi di riduzione reticolare[HPS98]. Innanzitutto bisogna considerare che la lunghezza attesa del vettore di lunghezza minima in un reticolo di dimensione n e determinante D è compresa tra

$$D^{\frac{1}{n}}\sqrt{\tfrac{n}{2\pi e}} \qquad \text{e} \qquad D^{\frac{1}{n}}\sqrt{\tfrac{n}{\pi e}}$$

Nel nostro caso $n = 2N$ e $D = q^N\alpha^N$, allora la lunghezza del vettore minimo è al più

$$s = (q^N\alpha^N)^{\frac{1}{2N}}\sqrt{\frac{2N}{2\pi e}} = \sqrt{\frac{Nq\alpha}{\pi e}}$$

Un algoritmo di riduzione avrebbe maggiori possibilità di trovare il vettore τ oppure un vettore la cui lunghezza è vicina a τ, se riesce a massimizzare $\frac{s}{\tau}$; elevando al quadrato questo rapporto si ottiene che un attaccante per ottenere ciò dovrebbe scegliere $\alpha = \frac{\|g\|_2}{\|f\|_2}$.
Scelto α in questo modo, si definisce c_h in modo che $\|\tau\|_2 = c_h s$; c_h è quindi il rapporto tra la lunghezza del vettore che stiamo cercando e la lunghezza prevista del vettore minimo. Più piccolo sarà c_h, più facile sarà trovare τ. Se invece c_h fosse vicino ad 1, i metodi di riduzione reticolare avrebbero notevoli difficoltà a trovare il vettore piccolo, mentre col diminuire di c_h l'operazione di riduzione diventerebbe man mano più semplice.

3.5 Attacco ai messaggi di NTRU basato sui reticoli

Un attacco basato sui reticoli può essere diretto al messaggio m anziché alla ricerca della chiave segreta[HGHPW05]. Il reticolo associato al problema è molto simile a quello usato in precedenza e il vettore da scoprire è nella forma $(\alpha m, \varphi)$. Come prima, l'attaccante dovrà scegliere $\alpha = \frac{\|\varphi\|_2}{\|m\|_2}$ e di conseguenza c_m. La costante c_m ci indica la vulnerabilità di ogni singolo messaggio ad un attacco basato sulla riduzione reticolare: più c_m è piccolo più il sistema è vulnerabile, mentre la difficoltà per gli attaccanti aumenta con l'avvicinarsi di c_m ad 1.
Noi vogliamo che gli attacchi ai componenti di h, f e g, o al messaggio m siano di difficoltà simile quindi scegliamo $c_h \approx c_m$ oppure equivalentemente $\|f\|_2\|g\|_2 \approx \|m\|_2\|\varphi\|_2$.

3.6 Attacco con una chiave falsa

Un attaccante anziché cercare di ottenere la chiave segreta f può utilizzare il reticolo descritto in 3.4 per trovare un altro vettore piccolo, appartenente allo stesso reticolo, della forma $\tau' = (\alpha\overline{f}', \overline{g}')$[CS97]. Se questo vettore è abbastanza piccolo f' potrà essere utilizzata come chiave per la decifratura. Più precisamente è molto probabile che

$$f' * e \equiv p\varphi * g' + m * f' \pmod{q}$$

soddisfi la proprietà

$$\|p\varphi * g' + m * f'\|_\infty < q$$

e quindi la decifrazione venga eseguita con successo.
Addirittura anche se la lunghezza di f' è 2, 3 o 4 volte q, è possibile utilizzarlo

per recuperare parte del messaggio[CS97]. Gli autori nel loro articolo suggeriscono, basandosi su prove sperimentali, che l'esistenza di chiavi false non porti nessun problema alla sicurezza di NTRU[HPS98]. Questo è verosimile poiché con gli algoritmi di riduzione attualmente utilizzati e la dimensione N molto grande, trovare vettori piccoli, anche se il doppio o il triplo di quello di lunghezza minima, è un operazione computazionalmente molto difficile.

3.7 Attacco chosen-ciphertext ad NTRU

Oltre agli attacchi basati sulla teoria dei reticoli, si è provato a forzare NTRU con degli attacchi del tipo "testo cifrato scelto". Il primo attacco di questo tipo è stato presentato da Èliane Jaulmes e Antoine Joux durante la conferenza **CRYPTO 2000**[JJ00]. L'idea base dell'attacco è quella di inviare dei particolari polinomi alla macchina decifratrice affinché il valore, dopo la riduzione modulare, sia diverso dal valore inserito. Ecco nello specifico come funziona l'attacco.
Innanzitutto si considerano i polinomi del tipo $ch + c$, dove c è un intero e h è la chiave pubblica, e si verifica come questi vengono trasformati dalla macchina decifratrice.

$$\begin{aligned} a &\equiv f * ch + cf \pmod{q} \\ &\equiv cg + cf \pmod{q} \end{aligned}$$

Ricordando che g e f hanno entrambi solo coefficienti uguali a 0, $+1$ o -1, segue che il polinomio $cf + cg$ ha coefficienti uguali a 0, $+c$, $-c$, $+2c$ e $-2c$. Ora, se scegliamo $c < \frac{q}{2}$ e $2c > \frac{q}{2}$, quando effettueremo la riduzione modulare dovremo ridurre solo i coefficienti uguali a $+2c$ o $-2c$. Se ad esempio supponiamo di avere un solo coefficiente in a pari a $\pm 2c$ e che

questo sia a_i allora il valore di $a \pmod q$ è $cg + cf - qx^i$. Quindi, il processo di decifrazione genera

$$cg * F_p + c - qx^i * F_p \pmod p$$

Se in precedenza avevamo scelto c multiplo di p, l'output finale sarà

$$-qx^i * F_p \pmod p$$

Sapendo che $MCD(p, q) = 1$ possiamo dividere per $q \pmod p$ e possiamo trovare

$$x^i * F_p \equiv \frac{x^i}{f} \pmod p$$

e calcolare il suo inverso $\frac{f}{x^i} \pmod p$. Siccome tutti i coefficienti di f sono solo $+1$ e -1, quello trovato è il valore vero e non viene modificato dalla riduzione modulare. Possiamo ora calcolare

$$h * \frac{f}{x^i} \equiv \frac{g}{x^i} \pmod q$$

e anche questo è il valore reale di $\frac{g}{x^i}$, poichè anche qui la riduzione modulare non interviene. Dal processo di creazione delle chiavi di NTRU si vede chiaramente che le chiavi (f, g) e $(\frac{f}{x^i}, \frac{g}{x^i})$ sono equivalenti.
Ovviamente, in generale, il polinomio $cf+cg$ avrà molti più coefficienti uguali a $\pm 2c$, quindi questo attacco in pratica non funzionerebbe. Tuttavia questo attacco si può generalizzare e rendere più efficiente.
Dati due polinomi P_1 e P_2 diciamo che essi hanno una collisione quando hanno un monomio con lo stesso coefficiente diverso da zero, ossia esiste un i tale che $(P_1)_i = (P_2)_i$. Definiamo allora intersezione polinomiale k di (P_1, P_2):

$$k = \sum k_i x^i$$

dove

$$k_i = \begin{cases} +1 \text{ se } P_1 \text{ e } P_2 \text{ hanno entrambi l'}i\text{-esimo coefficiente uguale a } +1 \\ -1 \text{ se } P_1 \text{ e } P_2 \text{ hanno entrambi l'}i\text{-esimo coefficiente uguale a } -1 \\ 0 \text{ altrimenti} \end{cases}$$

Con questa notazione la prima parte della decifrazione dei polinomi $c+ch$ diventa

$$\begin{aligned} a &\equiv cg + cf && \pmod{q} \\ &= cg + cf - qk \end{aligned}$$

e quindi il messaggio ottenuto dopo la decifrazione, avendo scelto in precedenza $c \equiv 0 \pmod{p}$, è

$$\begin{aligned} m &\equiv cF_p * g + cF_p * f - qF_p * k && \pmod{p} \\ &\equiv ch + c - qF_p * k && \pmod{p} \\ &\equiv -qF_p * k && \pmod{p} \end{aligned}$$

Ottenuto il messaggio, possiamo ricavare la chiave privata f da

$$f \equiv -qk * m^{-1} \pmod{p}$$

Quando f e g hanno pochi coefficienti in comune il polinomio k ha pochi coefficienti diversi da zero. Possiamo quindi provare differenti valori di k e calcolare i possibili polinomi f. Otterremo pochi polinomi f che appartengano ad $\mathcal{L}_f$, a questo punto basterà provare quale di questi f riesce a decifrare un messaggio di senso compiuto.

3.8 Nuova implementazione di NTRU

In seguito alla scoperta di questi attacchi, in particolar modo a quelli del tipo chosen-ciphertext [JJ00, GN07], gli autori di NTRU hanno proposto alcune modifiche alla versione originale. La modifica più significativa è la seguente, che prevede una differente chiave pubblica h:

$$h \equiv pF_q * g \pmod{q}$$

di conseguenza il messaggio cifrato diventa

$$e \equiv r * h + m \pmod{q}$$

avendo in precedenza scelto, casualmente, un polinomio ausiliario r.
Per decifrare il messaggio si procede calcolando

$$a \equiv f * e \pmod{q}$$

poi si calcola il polinomio b, ottenuto riducendo ciascuno dei coefficienti di a modulo p. Infine il messaggio sarà:

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

In seguito alla modifica di questa implementazione di NTRU sono stati proposti ulteriori attacchi; ne presenterò alcuni.

3.9 Attacco basato sui reticoli di Daewan Han

Nel 2005 Daewan Han presenta un nuovo tipo di attacco portato ad NTRU sfruttando i reticoli, ed in particolar modo l'intersezione tra più reticoli. Per prima cosa definiamo cosa si intende per intersezione fra reticoli[Han05].

Definizione 3.1.
Siano L e M due reticoli in $\mathbb{Z}^n$, si definisce **interezione** fra L ed M:

$$L \cap M \stackrel{def}{:=} \{v | v \in L \text{ e } v \in M\}$$

Utilizzando la notazione classica di NTRU definiamo $a = e - m$ e $I = \{0, 1, \ldots, N-1\}$, e introduciamo $\forall j \in I$ la seguente matrice M_j

$$M_j = \begin{pmatrix} 1 & 0 & \ldots & 0 & h_0 j & 1 \\ 0 & 1 & \ldots & 0 & h_1 j & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & \ldots & 1 & h_{(N-1)j} & 1 \\ 0 & 0 & \ldots & 0 & q & 0 \\ 0 & 0 & \ldots & 0 & a_j & d_r \end{pmatrix}$$

dove $h_i j = \frac{h_{(-1+j) \pmod N}}{p}$ e a_j il j-esimo coefficiente di a. Chiamo L_j il reticolo generato dalle righe di M_j.
Si può verificare che non può esistere una chave pubblica h di NTRU tale da generare $L_j = L_k$ per un qualche $j \neq k$. Quindi avremo differenti reticoli L_j per ogni $j \in I$. Consideriamo ora un sottoinsieme J non vuoto di I e definiamo L_J come

$$L_J = \bigcap_{j \in J} L_j$$

L_J gode della seguente proprietà:

Proposizione 3.9.1.
Sia J_t un insieme di t-elementi appartenenti ad I con $1 \leq t \leq N$ e q sia primo. Allora

$$\det(L_{J_t}) = d_r q^t$$

Da questa proposizione si noti che più intersechiamo i reticoli più il determinante del reticolo risultante aumenta.
Visto questo è chiaro che $\forall j \in J$ $(r, 0, 0)$ appartiene al reticolo L_j e che $\forall J_t \subset I$ anche L_{J_t} contiene $(r, 0, 0)$. La lunghezza della base di L_{J_t} è solitamente più grande di quella di L_{J_s} se $t > s$, allora, siccome la lunghezza di

$(r, 0, 0)$ è fissa ed è determinata solo da d_r, se incrementiamo t aumentano le possibilità che $(r, 0, 0)$ sia il vettore corto di L_{J_t}; quindi scelto t sufficientemente grande possiamo tranquillamente aspettarci che gli algoritmi di riduzione diano $(r, 0, 0)$ come soluzione.

3.10 Attacco meet-in-the-middle di Odlyzko

In [HG07] Nick Howgrave-Graham presenta l'attacco del tipo meet-in-the-middle ideato da Andrew Odlyzko; questo attacco è ritenuto il più efficiente contro NTRU.[HGHPW05]
L'idea dell'attacco è che se f_1 e f_2 sono tali che $f_1 + f_2 = f$, allora, dato che $(\mathrm{f}_1 + \mathrm{f}_2)h = g$ e g è binario, $\mathrm{x}_1 = \mathrm{f}_1 h$ e $\mathrm{x}_2 = -\mathrm{f}_2 h$ differiscono solamente di 0 o 1 modulo q.
Assumendo che f abbia d_f coefficienti uguali a $+1$ e che d_f sia pari, allora l'attacco procede per tentativi provando gli elementi f_1 con $\frac{d_f}{2}$ coefficienti uguali a $+1$ e calcolando $\mathrm{x}_1 = \mathrm{f}_1 h$. Il vettore x_1 corrispondente a x_1 ha lunghezza N e coefficienti compresi tra $-\frac{q}{2}$ e $+\frac{q}{2}$. Per ogni i-esimo coefficiente di x_1 determiniamo un bit β_i con

$$\beta_i = \begin{cases} +1 \text{ se } (x_1)_i > 0 \\ 0 \text{ altrimenti} \end{cases}$$

Possiamo poi creare una stringa da x_1 di N bit:

$$a_1 = \beta_1 : \beta_2 : \ldots \beta_N$$

che chiamiamo "etichetta". Sia ora $\overline{\beta} = 1 - \beta$ il complementare di β e $\overline{a}$ sia il complemetare componente per componente di una stringa a. L'elemento f_1 viene memorizzato in due diversi contenitori: uno con etichetta a_1, l'altro con etichetta $\overline{a_1}$.
Salviamo tutti gli f_1 provati in contenitori dipendenti da x_1. Dopo vari tentativi troveremo due elementi f_1 e f_2 tali che $\mathrm{f}_1 + \mathrm{f}_2 = \mathrm{f}$; in tal caso, dato che

$x_1 = -x_2 + g$, possiamo sperare che l'etichetta a_1 corrispondente a $x_1 = f_1 h$ sia uguale all'etichetta $\overline{a_2}$ corrispondente a $x_2 = f_2 h$.
Semplificando si può assumere che nel caso $f_1 + f_2 = f$ allora sicuramente sarà $a_1 = \overline{a_2}$; in tal caso diciamo di avere una "collisone" nel contenitore. Per ogni collisione analizziamo f_1 e f_2 memorizzati nel contenitore e controlliamo se $(f_1 + f_2)h$ è binario; se così fosse avremmo trovato un vettore piccolo della base pubblica di NTRU grazie al quale potremmo trovare (g, f).

3.11 Esempio

Ora riprendo l'esempio mostrato in 1.4, e cerco di trovare la chiave segreta utilizzando l'attaco illustrato in 3.3.
Conoscendo $h = -X^4 - 2X^3 + 2X + 1$, che è la chiave pubblica, posso creare la matrice $N \times N$:

$$H = \begin{pmatrix} 1 & 2 & 0 & -2 & -1 \\ -1 & 1 & 2 & 0 & -2 \\ -2 & -1 & 1 & 2 & 0 \\ 0 & -2 & -1 & 1 & 2 \\ 2 & 0 & -2 & -1 & 1 \end{pmatrix}$$

Posso poi considerare la matrice:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & -2 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 1 & 2 & 0 & -2 \\ 0 & 0 & 1 & 0 & 0 & -2 & -1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -2 & -1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & -2 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 \end{pmatrix}$$

Sia L il reticolo generato dalle righe di M. Ora $(\overline{f}, \overline{g})$ appartiene al reticolo L, e in particolare, siccome f e g hanno coefficienti piccoli, è un vettore piccolo del reticolo.

A questo punto devo applicare l'algoritmo LLL ai vettori che costituiscono la base del reticolo L ovvero alle righe di M. In questo caso, essendo la dimensione N molto piccola, LLL troverà il vettore di lunghezza minima del reticolo che molto probabilmente sarà $(\overline{f}, \overline{g})$. Scoperti f e g l'attaccante può decifrare tutti i messaggi inviati.

Appendice A

Sistemi crittografici a chiave pubblica resistenti ad attacchi quantistici

Quasi tutti i sistemi crittografici utilizzati in pratica si basano su due problemi ritenuti molto difficili: la fattorizzazione in numeri primi di un numero intero e il logaritmo discreto. L'invenzione del computer quantistico potrebbe però stravolgere questo equilibrio; infatti, nel 1994 Peter Shor in [Sho94] ha creato un algoritmo che utilizzando computer quantistici può risolvere il problema della fattorizzazione e calcolare il logaritmo discreto.
Una volta introdotto l'algoritmo di Shor la ricerca ha cercato di capire se tutti i problemi su cui si basa la crittografia possono essere risolti utilizzando un computer quantistico. Attualmente si ritiene che, anche con l'introduzione dei computer quantistici, una classe di problemi resterebbe di difficile soluzione e quindi potrebbe essere utilizzata, in futuro, per sostituire i sistemi crittografici attualmente in uso, in particolar modo RSA.
Tra tutti i sistemi che potrebbero resistere agli attacchi dei computer quantistici NTRU è sicuramente considerato il migliore per 3 motivi: la chiave pubblica è molto corta, la chiave privata utilizzata nelle firme effettuate con NTRU ha una durata praticamente illimitata, ed infine il costo computazio-

nale è molto basso.

La resistenza di NTRU ad attacchi quantistici non è stata dimostrata matematicamente ma si basa sulla mancanza di attacchi noti in grado di utilizzare i computer quantistici per risolvere il problema SVP. Questo, ovviamente, non significa che un'attacco ad NTRU non possa esistere ma che semplicemente non è ancora stato scoperto; tuttavia, i ricercatori, per il momento, lo ritengono piuttosto sicuro ed utilizzabile anche quando saranno in commercio computer quantistici.

Si può quindi prevedere, in futuro, un ampio utilizzo di NTRU, ed è proprio per questo che negli ultimi anni molti crittoanalisti hanno concentrato le loro ricerche nel tentativo di forzarlo, ma per il momento il sistema resiste.[PC09]

Bibliografia

[Coh93] Henri Cohen, *A course in computational algebraic number theory*, Graduate Texts in Mathematics, vol. 138, Springer-Verlag, Berlin, 1993. MR 1228206 (94i:11105)

[CS97] Don Coppersmith and Adi Shamir, *Lattice attacks on NTRU*, Advances in cryptology—EUROCRYPT '97 (Konstanz), Lecture Notes in Comput. Sci., vol. 1233, Springer, Berlin, 1997, pp. 52–61. MR 1603103 (98j:94016)

[GN07] Nicolas Gama and Phong Q. Nguyen, *New chosen-ciphertext attacks on NTRU*, Public key cryptography—PKC 2007, Lecture Notes in Comput. Sci., vol. 4450, Springer, Berlin, 2007, pp. 89–106. MR 2404114 (2009h:94124)

[Han05] Daewan Han, *A new lattice attack on NTRU cryptosystem*, Information Center for Mathematical Sciences, Trends in Mathematics, vol. 8, Springer, Korea, 2005, pp. 197–205.

[HG07] Nick Howgrave-Graham, *A hybrid lattice-reduction and meet-in-the-middle attack against NTRU*, Advances in cryptology—CRYPTO 2007, Lecture Notes in Comput. Sci., vol. 4622, Springer, Berlin, 2007, pp. 150–169. MR 2419599 (2009k:94109)

[HGHPW05] Nick Howgrave-Graham, Jeff Hoffstein, Jill Pipher, and William Whyte, *On estimating the lattice security of ntru*, Cryptology ePrint Archive, Report 2005/104, 2005.

[HPS98] 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.

[JJ00] Éliane Jaulmes and Antoine Joux, *A chosen-ciphertext attack against NTRU*, Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA), Lecture Notes in Comput. Sci., vol. 1880, Springer, Berlin, 2000, pp. 20–35. MR 1850034 (2002h:94059)

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), no. 4, 515–534. MR 682664 (84a:12002)

[Lov86] László Lovász, *An algorithmic theory of numbers, graphs and convexity*, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 50, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1986. MR 861822 (87m:68066)

[NS01] Phong Q. Nguyen and Jacques Stern, *The two faces of lattices in cryptology*, Cryptography and lattices (Providence, RI, 2001), Lecture Notes in Comput. Sci., vol. 2146, Springer, Berlin, 2001, pp. 146–180. MR 1903893 (2003d:94082)

[PC09] Ray A. Perlner and David A. Cooper, *Quantum resistant public key cryptography: a survey*, Proceedings of the 8th Symposium on Identity and Trust on the Internet (New York, NY, USA), IDtrust '09, ACM, 2009, pp. 85–93.

[Sho94] Peter W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, 35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994), IEEE Comput. Soc. Press, Los Alamitos, CA, 1994, pp. 124–134. MR 1489242

[TW06] 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)

Ringraziamenti

Grazie a tutti per avermi aiutato sempre in questi tre anni ed avermi accompagnato in questo percorso.

In particolare vorrei ringraziare il Professor Davide Aliffi per avermi introdotto al mondo della Crittografia e per i numerosi ed utili consigli, Anna, Piero e Carlo per avermi sopportato, mi scuso con tutti i Ravens per le numerose assenze dagli allenamenti, ed infine grazie ad Elisa per essermi sempre stata vicina.