Università degli Studi di Milano
Facoltà di Scienze e Tecnologie
Corso di Laurea in Sicurezza dei sistemi e delle reti informatiche

CRITTOGRAFIA POST-QUANTISTICA BASATA SUI RETICOLI: IMPLEMENTAZIONE E CRITTOANALISI DI GGH

Relatore: Prof. Stelvio CIMATO

Tesi di:
Gabriele BOTTANI
Matricola: 01701A

Anno Accademico 2023-2024

Indice

Capitolo 1

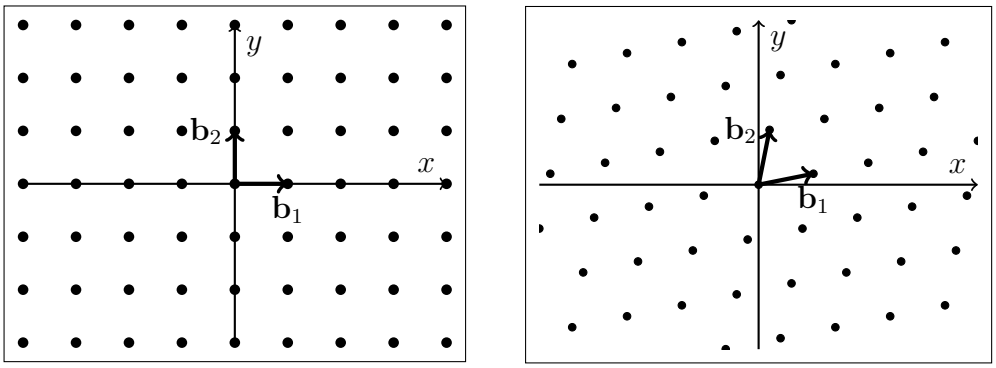
Introduzione

Capitolo 2

Proprietà e problemi sui reticoli

2.1 Reticoli

2.1.1 Nozioni base



(a) $\mathcal{L} = \mathbb{Z}^n$ (b) $\mathcal{L} = \mathbf{B}\mathbb{Z}^n$ dove $\mathbf{B} \in \mathbb{R}^{d \times n}$

Figura 1: Due esempi di strutture reticolari

Un reticolo è un insieme di punti in uno spazio di dimensione n che forma una struttura periodica. Ogni punto del reticolo può essere generato come combinazione lineare di n vettori, chiamati base, che sono linearmente indipendenti tra loro. La struttura e le proprietà di un reticolo dipendono dai vettori di base che, partendo dall'origine, definiscono il suo pattern di disposizione indicando le direzioni e le distanze tra i punti del reticolo.
Una proprietà fondamentale su cui si basa la definizione di reticolo è la proprietà dei coefficienti integrali: la base di un reticolo ha sempre coefficienti integrali, il che significa che tutti i vettori nella base sono combinazioni lineari intere l'uno dell'altro.

I reticoli possono essere formati in diversi modi, il più comune è il reticolo quadrato (Figura 1a) nel quale la base è allineata con gli assi cartesiani. Le altre varianti sono ottenibili applicando delle trasformazioni lineari alla base del reticolo quadrato (Figura 1b).
I reticoli sono normalmente definiti in uno spazio bidimensionale o tridimensionale, ma il concetto può essere esteso a spazi di dimensioni superiori. La rappresentazione dei vettori in questa tesi è quella per colonna, in quanto gli autori di [5] hanno preso questa decisione per la costruzione del loro crittosistema a chiave pubblica Goldreich Goldwasser Halevi (GGH), oggetto di questa tesi. Quindi per esempio, una matrice $\mathbf{B} \in \mathbb{R}^{m \times n}$ sarà divisa in vettori $[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n]$.
Una base può essere rappresentata da una matrice $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ avente, come precedentemente anticipato, i vettori base come colonne. Utilizzando la matrice come notazione, il reticolo generato da una matrice $\mathbf{B} \in \mathbb{R}^{n \times n}$ può essere definito come $\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$, dove $\mathbf{Bx}$ è una comune moltiplicazione matriciale.
Più formalmente, dati n vettori linearmente indipendenti $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$, il reticolo generato da essi è un set di vettori

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \sum_{i=1}^{n} \mathbf{b}_i \cdot \mathbb{Z} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}.$$

Lo stesso reticolo può essere generato da più basi composte ciascuna da vettori diversi

$$\mathcal{L} = \sum_{i=1}^{n} \mathbf{c}_i \cdot \mathbb{Z}.$$

Il determinante di un reticolo è il valore assoluto del determinante della matrice base $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$. Di conseguenza, per ogni matrice unimodulare (ovvero avente determinante +1 o -1) $\mathbf{U} \in \mathbf{Z}^{n \times n}$, $\mathbf{BU}$ è una base di $\mathcal{L}(\mathbf{B})$. Per verificare se due basi $\mathbf{R}$ e $\mathbf{B}$ generano lo stesso reticolo, è possibile utilizzare la matrice pseudo-inversa e trovare un $\mathbf{U}$ tale per cui $\mathbf{RU} = \mathbf{B}$.
Computando $\mathbf{R}^+$, ovvero la matrice pseudo-inversa di $\mathbf{R}$, si ha che

$$\mathbf{U} = \mathbf{R}^+ \, \mathbf{B}.$$

$\mathbf{R}^+$ è particolarmente facile da ottenere in questo caso in quanto i vettori colonna di $\mathbf{R}$ sono linearmente indipendenti per definizione. Di conseguenza la matrice pseudo-inversa assume la seguente forma:

$$\mathbf{R}^+ = (\mathbf{R}^* \, \mathbf{R})^{-1} \, \mathbf{R}^*$$

con $\mathbf{R}^*$ che è la matrice trasposta coniugata di $\mathbf{R}$. Dato che $\mathbf{R}$ è una matrice composta

da soli interi, la matrice trasposta coniugata è uguale alla matrice trasposta normale. Si ottiene quindi che

$$\mathbf{U} = (\mathbf{R}^T\ \mathbf{R})^{-1}\ \mathbf{R}^T\ \mathbf{B}.$$

Esempio 2.1.1. (Verificare che due basi generino lo stesso reticolo)
Siano $\mathbf{R}$ e $\mathbf{B}$ due basi generanti entrambi il reticolo $\mathcal{L}$ con

$$\mathbf{R} = \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix} \quad \text{e} \quad \mathbf{B} = \begin{bmatrix} 12 & 7 \\ -6 & -4 \end{bmatrix}$$

allora deve esistere una matrice unimodulare $\mathbf{U}$ tale che $\mathbf{RU} = \mathbf{B}$. Per trovare $\mathbf{U}$ è possibile calcolare

$$\mathbf{U} = (\mathbf{R}^T\ \mathbf{R})^{-1}\ \mathbf{R}^T\ \mathbf{B} = \begin{bmatrix} -3 & -2 \\ 5 & 3 \end{bmatrix}.$$

Ora è sufficiente controllare che

$$\mathbf{RU} = \mathbf{B} \quad \text{ovvero} \quad \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix} \times \begin{bmatrix} -3 & -2 \\ 5 & 3 \end{bmatrix} = \begin{bmatrix} 12 & 7 \\ -6 & -4 \end{bmatrix}$$

inoltre dato che $\det(\mathbf{U}) = 1$, si può affermare che $\mathbf{R}$ e $\mathbf{B}$ sono entrambe basi di $\mathcal{L}$.

2.1.2 Dominio Fondamentale

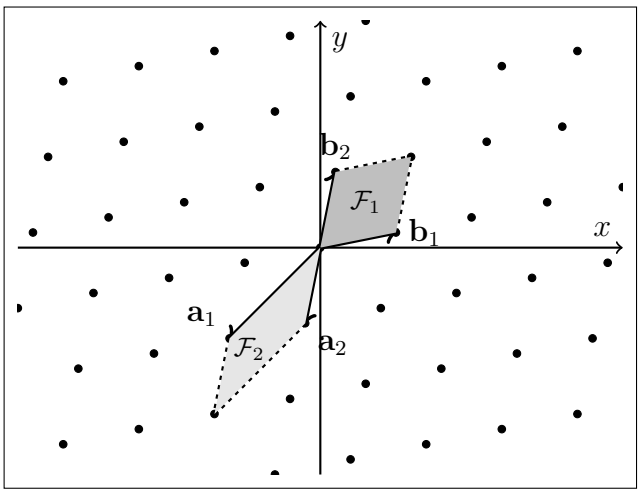


Figura 2: Un reticolo con due suoi domini fondamentali

Il dominio fondamentale è un concetto molto importante nei reticoli, grazie al quale è possibile capire la struttura matematica che li compone. Data una base abritraria $\mathbf{B}$ e un reticolo $\mathcal{L}$ è possibile immaginare il dominio fondamentale come un parallelepipedo

che ha come vertici: i vettori base $\mathbf{b}$ generanti il reticolo, il punto di origine e come quarto punto la somma dei vettori base all'origine.
Di tale parallelepipedo è possibile calcolarne il volume $\mathcal{F}(\mathbf{B})$, il quale è strettamente legato al determinante del reticolo. E' possibile osservare in Figura 2 un reticolo con due sue basi: nonostante i domini fondamentali abbiano forme diverse, l'area coperta dal loro volume è la medesima. Come dimostrato in [10, Sezione 7.4], proprio come per il determinante, il dominio fondamentale è un'invariante che è indipendente dalla scelta delle basi per il reticolo. Inoltre ne deriva la proprietà:

$$\mathcal{F}(\mathbf{B}) = \det(\mathcal{L})$$

e ricollegandoci a quanto detto nella sezione 2.1.1: $\mathcal{F}(\mathbf{B}) = \det(\mathcal{L}) = |\det(\mathbf{B})|$.
Una seconda proprietà fondamentale, sempre dimostrata in [10] è che tramite il dominio fondamentale è possibile ricostruire l'intero reticolo (Figura 3). In altre parole, ogni vettore $\mathbf{t} \in \mathbb{R}^n$ con $\mathcal{L} \subset \mathbb{R}^n$ può essere ottenuto sommando ripetutamente a un vettore $\mathbf{f} \in \mathcal{F}$ un altro vettore $\mathbf{v} \in \mathcal{L}$. Più formalmente:

$$\mathcal{F} + \mathbf{v} = \{\mathbf{f} + \mathbf{v} \mid \mathbf{f} \in \mathcal{F}, \mathbf{v} \in \mathcal{L}\}$$

comprende esattamente tutti i vettori nel reticolo $\mathcal{L}$.

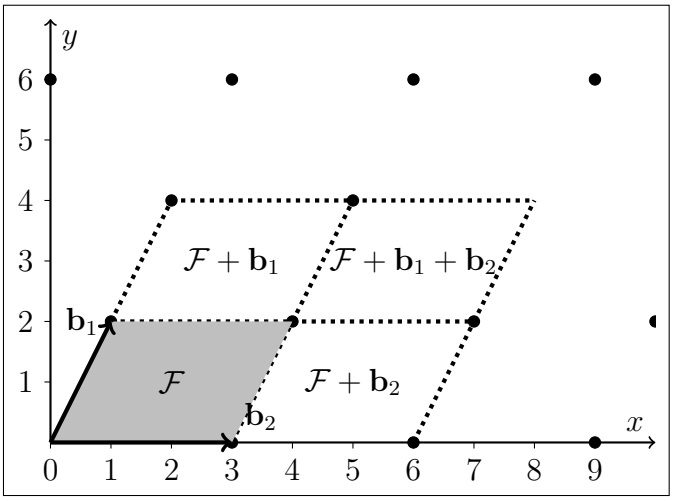


Figura 3: Il dominio fondamentale comprende esattamente tutti i vettori di $\mathcal{L}$

2.2 Problemi sui reticoli

Le costruzioni crittografiche basate su reticoli si basano sulla complessità computazionale che certe operazioni su di essi possono richiedere, soprattutto nel caso di spazi multidimensionali. I problemi computazionali più conosciuti relativi ai reticoli sono i seguenti:

- Problema del Vettore più Corto (SVP): Data una base di un reticolo $\mathbf{B}$, trovare il vettore non nullo di lunghezza minima in $\mathcal{L}(\mathbf{B})$.
- Problema del Vettore più Vicino (CVP): Data una base di un reticolo $\mathbf{B}$ e un vettore target $\mathbf{t}$ (non necessariamente nel reticolo), trovare il vettore $\mathbf{w} \in \mathcal{L}(\mathbf{B})$ più vicino a $\mathbf{t}$ minimizzando $\|\mathbf{t} - \mathbf{w}\|_2$.
- Problema dei Vettori Lineramente Indipendenti più Corti (SIVP): Data una base di un reticolo $\mathbf{B} \in \mathbb{Z}^{n \times n}$, trovare n vettori linearmente indipendenti $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$ (dove $\mathbf{s}_i \in \mathcal{L}(\mathbf{B})$) per tutte le i) minimizzando la quantità $||\mathbf{S}|| = \max_i ||\mathbf{s}_i||$. SIVP è una variante di SVP, ma a differenza di quest'ultimo, SIVP mira a identificare un insieme di vettori indipendenti che siano i più corti possibile, in altre parole la ricerca di una base ortogonale o ortonormale che generi il reticolo e che minimizzi la lunghezza dei suoi vettori.

La complessità per risolvere CVP è stata provata essere NP-difficile[8], stessa cosa vale per SVP, ma sotto alcune circostanze specifiche[9]. Per questi motivi vengono comparati come problemi dalla stessa difficoltà anche se, in pratica, risolvere CVP è considerato essere un po' più difficile di SVP nella stessa dimensione. Ognuno di questi due problemi ha un relativo sotto-problema che nient'altro è che una variante approssimativa: il Problema del Vettore più Vicino Approssimato (apprCVP) e Problema del Vettore più Corto Approssimato (apprSVP). Questi sotto-problemi consistono nel trovare un vettore non nullo la cui lunghezza non sia più lunga di un fattore dato $\Psi(n)$ rispetto a un vettore non nullo corretto (più corto o più vicino a seconda del problema).
In particolare GGH si basa sulla risoluzione del CVP basandosi su una delle proprietà fondamentali dei reticoli: la possibilità di usare più basi per lo stesso reticolo. Utilizzando due basi $\mathbf{A}$ e $\mathbf{B}$, definite rispettivamente come "buona" e "cattiva", ma che generano lo stesso reticolo, diventa più agevole risolvere determinati problemi sui reticoli utilizzando la base $\mathbf{A}$ piuttosto che con $\mathbf{B}$. Per questi motivi il CVP sarà il fulcro dei problemi discussi in questa tesi assieme al SVP, il quale verrà trattato prevalentemente per quanto riguarda la crittoanalisi di GGH.

2.3 Riduzione di un reticolo

2.3.1 Rapporto di Hadamard

Si supponga di avere a disposizione due basi $\mathbf{A}$ e $\mathbf{B}$ che godono della proprietà di generare lo stesso reticolo. Seppur condividendo tale caratteristica, $\mathbf{A}$ e $\mathbf{B}$ sono in realtà molto diverse nella loro struttura; in particolare $\mathbf{A}$ è composta da vettori corti e quasi ortogonali fra loro mentre $\mathbf{B}$ è composta da vettori lunghi e quasi paralleli fra loro.
La qualità di una base risiede in queste differenze dei vettori costituenti le basi, chiamiamo quindi base "buona" $\mathbf{A}$ e base "cattiva" $\mathbf{B}$. E' necessario però definire una metrica per valutare quanto una base sia buona o meno; a tal proposito Hadamard[10] introdusse una formula quantitativa per misurare la qualità di una base reticolare, il cosiddetto rapporto di Hadamard.

Data una base $\mathbf{B} = [\mathbf{b_1}, \mathbf{b_2}. \ldots, \mathbf{b}_n]$ e un reticolo $\mathcal{L}$ di dimensione n generato da $\mathbf{B}$, il rapporto di Hadamard della base $\mathbf{B}$ è definito dal valore:

$$\mathcal{H}(\mathbf{B}) = \left(\frac{\det(\mathcal{L})}{\|\mathbf{b}_1\|_2 \cdot \|\mathbf{b}_2\|_2, \cdots \|\mathbf{b}_n\|_2} \right)^{\frac{1}{n}}$$

Il rapporto di Hadamard si configura nell'intervallo $(0, 1]$, dove più vicino all'1 si è e più la base è buona, viceversa più vicino allo 0 si è e più la base è cattiva. Questa formula verrà utilizzata come unica misura per verificare la qualità degli esempi di basi che verranno presentate più avanti in questa tesi.

2.3.2 Ortogonalizzazione Gram-Schmidt

Ora che è possibile giudicare una base dato il suo rapporto di Hadamard, utilizziamo la base $\mathbf{B}$ definita nella precedente sezione, la quale ipotizziamo abbia un $\mathcal{H}(\mathbf{B})$ prossimo allo zero. Se volessimo utilizzare questa base per risolvere uno dei problemi dei reticoli, molto probabilmente non riusciremmo mai a raggiungere una soluzione che sia valida o quantomeno che sia vicina alla soluzione ottima. A questo proposito sono stati ideati degli algoritmi in grado di ortogonalizzare una base cattiva per convertirla in una buona e mantenere le proprietà del reticolo iniziale, si ottiene quindi una base $\mathbf{B}'$ tale che: $\mathcal{H}(\mathbf{B}^*) \approxeq 1$ e che $\det(\mathbf{B}) = \det(\mathbf{B}^*)$.
Prima di discutere questo tipo di algoritmi è necessario affrontare brevemente l'algoritmo di Gram-Schmidt, il quale, esegue un tipo di ortogonalizzazione che viene applicata su spazi vettoriali e che è anche chiamata Ortogonalizzazione Gram-Schmidt (GSO). Questo algoritmo non è adottabile direttamente sulle basi reticolari in quanto esso

andrebbe a violare la proprietà dei coefficienti integrali, di fondamentale importanza nella definizione di reticolo. Nonostante ciò, questo algoritmo gode di una proprietà chiave che viene utilizzata in algoritmi di riduzione dei reticoli. Come dimostrato in [10, Teorema 7.13]:
siano $\text{span}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ e $\text{span}(\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*)$ gli spazi vettoriali generati rispettivamente dalle colonne di $\mathbf{B}$ e $\mathbf{B}^*$, allora se $\mathbf{B}^*$ è il risultato di GSO applicato a $\mathbf{B}$:

$$\text{span}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \text{span}(\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*).$$

Ciò comporta che GSO può essere chiamato come sotto-routine di algoritmi per la trasformazione delle matrici di spazi vettoriali, riducendo notevolmente i calcoli e semplificando l'implementazione di algoritmi per la riduzione di reticoli.

Algoritmo 1: Algoritmo di Gram-Schmidt

Input: Una matrice $\mathbf{B}$ tale che $\text{rk}(\mathbf{B}) = \text{col}(\mathbf{B})$
Output: Una matrice $\mathbf{B}^*$ ortogonale
$\mathbf{b}_1^* = \mathbf{b}_1$
for $i = 2$ **to** n **do**
  for $j = 1$ **to** $i - 1$ **do**
    $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$
  end
  $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$
end
return $\mathbf{B}^*$

Dove rk() indica il rango e col() il numero delle colonne.

2.3.3 Algoritmo di Lenstra-Lenstra-Lovász

L'algoritmo di Lenstra-Lenstra-Lovász (LLL)[11, 10] è noto come uno dei più famosi algoritmi per la riduzione dei reticoli. In teoria, opera con un tempo polinomiale $O(n^6(\log \mathcal{E})^3)$, dove n è la dimensione di un reticolo $\mathcal{L}$ dato ed $\mathcal{E}$ rappresenta la massima lunghezza euclidea dei vettori nella base fornita. Il risultato di LLL è una base ridotta, il cui vettore più breve può essere considerato una soluzione approssimata dell'apprSVP con un fattore di $2^{(n-1)/2}$ rispetto alla lunghezza del vettore corretto più corto. Nonostante questo fattore sia esponenziale, l'algoritmo si dimostra sorprendentemente efficiente nella pratica e ciò lo rende un tassello fondamentale negli algoritmi di risoluzione del CVP e del SVP.

Una base $\mathbf{B}^*$ per essere considerata LLL-ridotta deve soddisfare due condizioni:

- Condizione di grandezza: $\mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} \leq \eta$ per ogni $1 \leq j < i \leq n$.
- Condizione di Lovász: $\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle \geq (\delta - \mu_{i,i-1}^2) \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle$ per ogni $1 < i \leq n$.

I valori δ e η sono standardizzati rispettivamente a $\frac{3}{4}$ e $\frac{1}{2}$.

Algoritmo 2: Algoritmo di LLL, con $\delta = \frac{3}{4}, \eta = \frac{1}{2}$

Input: Una matrice $\mathbf{B}$ che sia base di un reticolo
Output: Una matrice $\mathbf{B}^*$ LLL-ridotta
$k = 1$
$\mathbf{b}_1^* = \mathbf{b}_1$
while $k <= n$ **do**
  for $j = k - 1$ **to** 0 **do**
    if $|\mu_{k,j}| > \eta$ **then** `// Condizione di grandezza`
      $\mathbf{b}_k = \mathbf{b}_k - \lfloor \mu_{k,j} \rceil \mathbf{b}_j$
      GramSchmidt($\mathbf{B}$)
    end
  end
  if $\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle \geq (\delta - \mu_{k,k-1}^2) \langle \mathbf{b}_{k-1}^*, \mathbf{b}_{k-1}^* \rangle$ **then** `// Condizione di Lovász`
    $k = k + 1$
  end
  else
    Scambia $\mathbf{b}_{k-1}$ e $\mathbf{b}_k$
    GramSchmidt($\mathbf{B}$)
    $k = \max(k - 1, 1)$
  end
end

Esempio 2.3.1. (Riduzione di una base con LLL) Sia $\mathcal{L}$ un reticolo generato dalla matrice:

$$\mathbf{B} = \begin{bmatrix} 87634 & 88432 & 94345 \\ 32323 & 27883 & 40323 \\ -21221 & -11234 & -32123 \end{bmatrix}$$

con $\mathcal{H}(\mathbf{B}) = 0.10393$ e $\det(\mathbf{B}) = -1079906335101$.

Si applichi ora una riduzione LLL alla matrice $\mathbf{B}$:

$$\mathbf{B}^* = \begin{bmatrix} 7509 & 798 & 5833 \\ 3560 & -4440 & -11277 \\ -915 & 9987 & -1169 \end{bmatrix}$$

Ricalcolando $\mathcal{H}(\mathbf{B}^*)$ è possibile osservare che:

$$\mathcal{H}(\mathbf{B}^*) = 0.72358 \text{ e } \det(\mathbf{B}^*) = 1079906335101.$$

E' stato ottenuto un incremento notevole del rapporto di Hadamard grazie alla riduzione LLL senza interferire con le proprietà della base $\mathbf{B}$. Infatti $|\det(\mathbf{B})| = |\det(\mathbf{B}^*)|$ e, usando la formula descritta nella Sezione 2.1, è possibile trovare una matrice di interi

$$\mathbf{U} = \begin{bmatrix} -2 & -1 & 22 \\ 1 & 1 & -10 \\ 1 & 0 & -11 \end{bmatrix} \text{ con } \det(\mathbf{U}) = -1$$

tale per cui $\mathbf{BU} = \mathbf{B}^*$.

2.3.4 Varianti di LLL

LLL è un eccellente algoritmo in grado di restituire in un tempo polinomiale una matrice quasi ortogonale partendo da una con vettori quasi paralleli, o che comunque è ritenibile di bassa qualità. Esistono però varianti che ne velocizzano i calcoli, così come altri algoritmi capaci di restituire una base di qualità ancora superiore rispetto a quella ottenuta con la riduzione LLL. Di seguito vengono presentate brevemente tre versioni dell'algoritmo.

La prima versione discussa è quella in virgola mobile (FPLLL)[12], la quale utilizza aritmetica in virgola mobile a precisione arbitraria per accelerare i calcoli razionali dell'algoritmo originale. Questa versione ha come vantaggio un aumento delle performance: in comparazione con LLL il tempo di computazione nel caso peggiore è $O(n^3(\log \mathcal{E})^2)$.

La seconda versione è stata presentata da Schnorr-Euchner [13] ed il suo nome originale è "deep insertions" ovvero inserzioni profonde. In LLL (Algoritmo 2), è presente un passaggio in cui avviene uno scambio tra il vettore $\mathbf{b}_{k-1}$ e $\mathbf{b}_k$, il quale di solito permette qualche riduzione di grandezza ulteriore del nuovo $\mathbf{b}_k$. Nella variante deep insertions, viene invece inserito $\mathbf{b}_k$ tra $\mathbf{b}_{i-1}$ e $\mathbf{b}_i$ con i che viene scelta in modo da apportare una maggiore riduzione di grandezza. L'algoritmo risultante, nel caso peggiore, potrebbe non terminare in un tempo polinomiale, ma in pratica, quando eseguito sulla maggioranza dei reticoli, termina rapidamente e può fornire in output una base ridotta significativamente migliore di quella di LLL standard.

L'ultima variante discussa è basata sull'algoritmo di riduzione Korkin–Zolotarev (KZ)[2, Sezione 18.5]. Le caratteristiche di una base KZ-ridotta sono generalmente migliori rispetto a quelle di LLL, ma richiedono una complessità maggiore e un tempo di computazione non polinomiale; per le proprietà complete si veda il riferimento. Più nel dettaglio, il problema principale, è che non esiste un algoritmo in grado di computare una base KZ in tempo polinomiale. L'algoritmo più veloce conosciuto richiede un tempo di computazione esponenziale rispetto alla dimensione. Per compensare a tale problema KZ apporta un grande vantaggio in rispetto all'accuratezza della riduzione, infatti, il primo vettore di una base KZ-ridotta è sempre una soluzione al SVP. Dato che la complessità di KZ cresce con n, è logico pensare che a basse dimensioni sia comunque sufficientemente veloce. Un'idea è quindi quella di computare una riduzione di proiezioni a dimensioni più basse del reticolo originale. L'algoritmo in questa configurazione prende il nome di Korkine-Zolotarev a blocco (BKZ), il quale, se combinato con LLL, diventa una variante di quest'ultimo chiamata LLL-BKZ. Questa variante è in grado di bilanciare costo computazionale e qualità di riduzione ottenendo così l'algoritmo più efficiente per SVP in grandi dimensioni, dimostrando anche una qualità di riduzione significativamente migliore di quella di LLL standard.

2.4 Algoritmi per la risoluzione del CVP

2.4.1 Algoritmi di Babai

Nel 1986 Babai[4] propose due algoritmi per la risoluzione di apprCVP, i cosiddetti: "Metodo del Piano più Vicino" e "Tecnica di Arrotondamento". Ai fini di questa tesi, entrambi verranno trattati, sebbene il primo sarà discusso in modo più conciso poiché, come verrà spiegato nei prossimi capitoli, non è stato utilizzato nelle implementazioni proposte.
Il metodo del piano più vicino è il primo algoritmo presentato da Babai, esso si basa sull'impiegare l'ortogonalizzazione di Gram-Schmidt per semplificare il problema. L'algoritmo inizia quindi ortogonalizzando la base del reticolo fornita in input attraverso Gram-Schmidt. Successivamente, procede in maniera iterativa a partire dalla dimensione più alta: il vettore input viene proiettato sul vettore base ortogonale corrispondente e questa proiezione viene approssimata al multiplo intero più vicino del vettore della base originale. Tale approssimazione viene sottratta dal vettore input, generando un nuovo vettore residuo. Questo processo viene ripetuto per le dimensioni inferiori, una alla volta, fino a coprire tutte le dimensioni. Al termine, l'algoritmo fornisce come risultato una soluzione all'apprCVP. Grazie alla sua complessità polinomiale, l'algoritmo riesce a bilanciare efficacemente l'accuratezza dell'approssimazione con il tempo di esecuzione. Per ulteriori dettagli e informazioni sull'algoritmo si veda [2].

Il secondo algoritmo è la tecnica di arrotondamento che, come da nome, si basa principalmente sull'arrotondare dei valori frazionari all'intero più vicino. Seppur la sua implementazione risulti semplice e banale, in realtà la sua dimostrazione teorica è tutt'altro che immediata. A differenza del precedente, non utilizza l'ortogonalizzazione di Gram-Schmidt, ma mantiene comunque una complessità polinomiale. Di seguito viene fornita una spiegazione del suo funzionamento.
Come discusso nella sezione 2.1.2, dati un reticolo $\mathcal{L}$ di dimensione n e una sua base $\mathbf{B}$, per ogni vettore $\mathbf{t} \in \mathbb{R}^n$, con $\mathbf{t} \notin \mathcal{L}$, un'unica decomposizione $\mathbf{t} = \mathbf{f} + \mathbf{v}$ può essere sempre trovata in modo tale che $\mathbf{v} \in \mathcal{L}$ e $\mathbf{f}$ si collochi nel dominio fondamentale $\mathcal{F}$ di $\mathbf{B}$. Questa proprietà fornisce l'idea dietro alla risoluzione dell'apprCVP usata da questo algoritmo: identificare il dominio fondamentale (traslato) rispettivamente a $\mathbf{v} \in \mathcal{L}$, nel quale il vettore target $\mathbf{t}$ si trova. Sia $\mathcal{L}$ un reticolo con dimensione n generato da una base (buona) $\mathbf{B}$ e sia $\mathbf{t}$ un vettore tale che $\mathbf{t} \in \mathbb{R}^n$ e $\mathbf{t} \notin \mathcal{L}$. Dato che $\mathbf{B}$ è una matrice di rango massimo, è possibile calcolare

$$\mathbf{x} = \mathbf{B}^{-1}\mathbf{t}$$

Da qui si applica la tecnica di arrotondamento, la quale è semplicemente

$$\mathbf{w} = \sum_{i=1}^{n} \mathbf{b}_i \lfloor \mathbf{x}_i \rceil$$

con $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ e $\lfloor \mathbf{x} \rceil$ che significa prendere l'intero più vicino al numero reale $\mathbf{x}$. Questo algoritmo mira ad identificare il dominio fondamentale (traslato) che il vettore $\mathbf{t}$ localizza e la sua correttezza è strettamente legata alla forma geometrica del dominio fondamentale, è necessaria quindi una base di alta qualità al fine di avere risultati validi.

Esempio 2.4.1. (Risoluzione del CVP usando la tecnica di arrotondamento di Babai)

Siano $\mathbf{R}$ e $\mathbf{B}$ le stesse basi definite nell'Esempio 2.1.1 e sia $\mathbf{t} \in \mathbb{R}^n, \mathbf{t} \notin \mathcal{L}$ con

$$\mathbf{t} = \begin{bmatrix} 7 \\ 3.5 \end{bmatrix}.$$

Si inizi con l'applicare l'algoritmo, dal primo passo si ottiene

$$\mathbf{R}^{-1} = \begin{bmatrix} 0 & 0.5 \\ 0.33 & 0.17 \end{bmatrix} \quad \text{e} \quad \mathbf{x} = \mathbf{R}^{-1}\mathbf{t} = \begin{bmatrix} 1.75 \\ 1.75 \end{bmatrix}$$

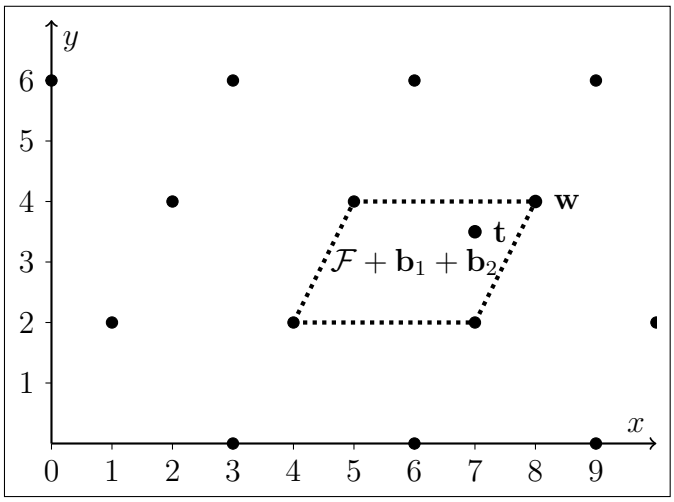


Figura 4: Risoluzione del CVP usando la tecnica di arrotondamento di Babai: $\mathbf{w}$ è il vertice del dominio fondamentale traslato localizzato da $\mathbf{t}$, quindi è soluzione per apprCVP.

si applichi ora la tecnica di arrotondamento a $\mathbf{x}$

$$\lfloor \mathbf{x} \rceil = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

si proceda infine con l'ottenere il risultato finale

$$\mathbf{w} = \mathbf{R}\mathbf{x} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}$$

che, come mostrato in Figura 4, è il vettore più vicino a $\mathbf{t}$ con $\|\mathbf{t} - \mathbf{w}\|_2 = 1.12$. Se si dovesse valutare la qualità della base, si otterrebbe che $\mathcal{H}(\mathbf{R}) = 0.97400$ e, grazie a tali proprietà ortogonali di $\mathbf{R}$, il dominio fondamentale derivante assume una forma geometrica tale per cui l'algoritmo è in grado di raggiungere facilmente la soluzione. Si riesegua ora l'algoritmo su $\mathbf{B}$. Calcolando $\mathcal{H}(\mathbf{B}) = 0.24473$ si scopre che $\mathbf{B}$ offre una qualità molto più bassa rispetto a $\mathbf{R}$. Procedendo si ottiene che

$$\mathbf{x}_2 = \mathbf{B}^{-1}\mathbf{t} = \begin{bmatrix} 8.75 \\ -14 \end{bmatrix} \quad \text{e quindi} \quad \lfloor \mathbf{x}_2 \rceil = \begin{bmatrix} 9 \\ -14 \end{bmatrix}.$$

Computando l'ultimo passaggio, il vettore risultante è

$$\mathbf{w}_2 = \mathbf{B}\mathbf{x}_2 = \begin{bmatrix} 10 \\ 2 \end{bmatrix}$$

il quale non è soluzione corretta all'apprCVP in quanto $\|\mathbf{t}-\mathbf{w}_2\|_2 = 3.35 > \|\mathbf{t}-\mathbf{w}\|_2$.

La principale differenza tra i due algoritmi di Babai è che il metodo del piano più vicino risulta essere più preciso in quanto i valori frazionari vengono arrotondati in maniera adattiva piuttosto che tutti insieme in un'unica volta.

2.4.2 Tecnica di incorporamento

Babai, oltre alla presentazione dei due algoritmi precedentemente trattati, ha dimostrato anche quanto una base ridotta migliori l'approssimazione della soluzione ad apprCVP. In particolare, con una base LLL-ridotta, questo porta ad un fattore di approssimazione esponenziale per entrambi i suoi algoritmi. Nella pratica però, il metodo migliore per risolvere apprCVP, è la cosiddetta tecnica di incorporamento[2] che si basa sul ridurre il CVP a un SVP.
Sia $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ la base di un reticolo $\mathcal{L}$ di dimensione n e sia $\mathbf{t} \in \mathbb{R}^n, \mathbf{t} \notin \mathcal{L}$. La tecnica di incorporamento impone la costruzione di un reticolo di dimensione $n+1$ con la seguente struttura:

$$\mathbf{M} = \begin{bmatrix} \vdots & \vdots & \vdots & & \vdots \\ \mathbf{t} & \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 0 & 0 & & 0 \end{bmatrix}$$

Il nuovo reticolo $\mathcal{L}(\mathbf{M})$ è strutturato in modo tale da avere lo stesso determinante di $\mathcal{L}(\mathbf{B})$ e quasi la stessa dimensione, ci si può quindi aspettare che il vettore più corto di $\mathcal{L}(\mathbf{M})$ abbia quasi la stessa lunghezza di quello di $\mathcal{L}(\mathbf{B})$. Si assuma che $\mathbf{w} \in \mathcal{L}$ minimizzi la distanza per $\mathbf{t}$ e sia $\mathbf{u} = \mathbf{t} - \mathbf{w}$, allora il vettore

$$\mathbf{v} = \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}$$

appartiene a $\mathcal{L}'$ e, se dovesse anche essere il suo vettore più corto, si potrebbe risolvere l'apprCVP di $\mathcal{L}$ determinando l'apprSVP di $\mathcal{L}'$. Per ottenere $\mathbf{v}$ è sufficiente ridurre $\mathbf{M}$ mediante algoritmi come LLL (o meglio BKZ-LLL) per poi ottenere $\mathbf{w}$ calcolando $\mathbf{t}-\mathbf{u}$. Un problema nella pratica sta nella scelta di $\mathbf{t}$: teoricamente $\mathbf{t}$ può appartenere all'insieme $\mathbb{R}$, ma questo creerebbe problemi nella costruzione della nuova base $\mathbf{M}$ la quale non soddisferebbe più la proprietà dei coefficienti integrali che sta alla base della definizione di reticolo. Tale problema viene discusso e affrontato nell'attacco di Nguyen contro GGH, il quale verrà trattato nelle prossime sezioni. Nel concreto si tenta di mantenere $\mathbf{t} \in \mathbb{Z}^n$ in modo da evitare problemi di questa natura.

Esempio 2.4.2. (Risoluzione del CVP usando la tecnica di incorporamento) Siano $\mathbf{R}$ e $\mathbf{B}$ le stesse basi definite nell'Esempio 2.1.1 e sia

$$\mathbf{t} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}.$$

Seguendo quanto descritto nella tecnica di incorporamento, si costruisca la matrice

$$\mathbf{M} = \begin{bmatrix} \mathbf{t}_{0,0} & \mathbf{r}_{0,0} & \mathbf{r}_{1,0} \\ \mathbf{t}_{1,0} & \mathbf{r}_{0,1} & \mathbf{r}_{1,1} \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 6 & 1 & 3 \\ 3 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

e la si riduca usando un algoritmo di riduzione, in questo caso LLL:

$$\mathbf{M}^* = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 2 \\ -1 & -1 & 0 \end{bmatrix}.$$

Infine è necessario che si estraggano i primi n valori dalla prima colonna di $\mathbf{M}^*$ sottraendoli poi a $\mathbf{t}$:

$$\mathbf{w} = \mathbf{t} - \mathbf{u} = \begin{bmatrix} 6 \\ 3 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 7 \\ 2 \end{bmatrix}$$

che è soluzione all' apprCVP con $\|\mathbf{t} - \mathbf{w}\|_2 = 1.41$. Utilizzando la base cattiva $\mathbf{B}$ invece, si otterrebbe:

$$\mathbf{M}_2 = \begin{bmatrix} 6 & 12 & 7 \\ 3 & -6 & -4 \\ 1 & 0 & 0 \end{bmatrix} \quad \text{con} \quad \mathbf{M}_2^* = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 2 \\ 1 & -1 & 0 \end{bmatrix}.$$

Ed effettuando l'ultimo passaggio:

$$\mathbf{w}_2 = \mathbf{t} - \mathbf{u}_2 = \begin{bmatrix} 6 \\ 3 \end{bmatrix} - \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

che è comunque soluzione all'apprCVP in quanto $\|\mathbf{t} - \mathbf{w}_2\|_2 = \|\mathbf{t} - \mathbf{w}\|_2 = 1.41$.

Grazie a questo esempio si può comprendere meglio l'efficacia nella pratica di questa tecnica: in comparazione con l'algoritmo di arrotondamento di Babai, nonostante la bassa qualità della seconda base, si è riusciti comunque a trovare una soluzione corretta.

Capitolo 3

Crittosistema a chiave pubblica GGH

3.1 Struttura e funzionamento di GGH

Nel 1996, Oded Goldreich, Shafi Goldwasser e Shai Halevi[5] hanno introdotto un nuovo sistema crittografico a chiave pubblica basato sulla difficoltà di risolvere CVP in reticoli di dimensioni elevate.
L'idea dietro GGH è la seguente: si supponga di avere un messaggio $\mathbf{m}$ codificato in un vettore appartenente ad un reticolo $\mathcal{L}$, un vettore target $\mathbf{t} \notin \mathcal{L}$ vicino ad $\mathbf{m}$ e due basi $\mathbf{R}$ e $\mathbf{B}$ entrambe generanti $\mathcal{L}$ e rappresentanti rispettivamente base privata e base pubblica. Siano $\mathbf{R}$ una base buona e $\mathbf{B}$ una base cattiva, allora, tramite l'utilizzo dell'algoritmo di arrotondamento di Babai (Sezione 2.4.1), sarà possibile ritrovare il vettore più vicino a $\mathbf{t}$ (che risulterà essere $\mathbf{m}$) usando la base privata, ma non usando la base pubblica.
Più formalmente, GGH è definito da una funzione trapdoor (ovvero una funzione matematica che è facile da calcolare in una direzione, ma molto difficile da invertire senza dei dati segreti), la quale è composta da 4 funzioni probabilistiche di complessità polinomiale:

- Generate: Dato in input un intero positivo n vengono generate due basi $\mathbf{R}$ e $\mathbf{B}$ di rango massimo in $\mathbb{Z}^n$ e un numero positivo reale σ. Le basi $\mathbf{R}$ e $\mathbf{B}$ sono rappresentate da matrici $n \times n$ e sono rispettivamente denominate base privata e base pubblica. Sia $\mathbf{R}$ che $\mathbf{B}$ generano lo stesso reticolo $\mathcal{L}$ e, insieme a σ, danno origine a chiave privata e chiave pubblica. Per maggiori dettagli riguardo la generazione delle chiavi si veda la prossima sezione.

- Sample: Dati in input $\mathbf{B}, \sigma$ vengono originati i vettori $\mathbf{v}, \mathbf{e} \in \mathbb{R}^n$.
  Il vettore $\mathbf{v}$ viene scelto casualmente da un cubo in $\mathbb{Z}^n$ che sia sufficientemente

grande. Gli autori suggeriscono di scegliere in maniera casuale ogni valore di $\mathbf{v}$ uniformemente dall'intervallo $[-n^2, -n^2+1, \ldots, +n^2]$, sottolineando però che la scelta di n^2 è arbitraria e che non hanno prove di come essa possa influenzare la sicurezza del crittosistema stesso.
Il vettore $\mathbf{e}$ invece, viene scelto casualmente in $\mathbb{R}^n$ in modo tale la media dei valori sia zero e la varianza sia σ^2. Il metodo più semplice per generare tale vettore è quello di scegliere ogni valore di $\mathbf{e}$ come $\pm\sigma$ con probabilità $\frac{1}{2}$. Questo vettore ha l'importante funzione di essere un errore che viene aggiunto al calcolo del testo cifrato per complicarne la decifratura.

- Evaluate: Dati in input $\mathbf{B}, \sigma, \mathbf{v}, \mathbf{e}$ si calcola $\mathbf{c} = \mathbf{Bv} + \mathbf{e}$. Con questo calcolo si ottiene il messaggio cifrato che è rappresentato da $\mathbf{c}$.

- Invert: Dati in input $\mathbf{R}, \mathbf{c}$ si utilizza la tecnica di arrotondamento di Babai per invertire la funzione trapdoor e ricavare il messaggio originale.

3.1.1 Generazione delle chiavi

La generazione delle chiavi è un elemento cruciale in tutti i crittosistemi asimmetrici. In GGH, per costruire le chiavi, è indispensabile ottenere prima due basi: una pubblica e una privata. La sicurezza di questo crittosistema si basa sul fatto che la base pubblica non sia di qualità sufficientemente alta, in modo tale da impedire l'applicazione efficace di un algoritmo di risoluzione del CVP al testo cifrato, evitando così di recuperare il messaggio originale. È dunque fondamentale il modo in cui vengono generate la base privata e, soprattutto, la base pubblica, per garantire le caratteristiche necessarie a mantenere la sicurezza del crittosistema. In questa sezione verrà quindi analizzata la struttura della funzione Generate, la quale si occupa di quanto introdotto precedentemente.
Questa funzione prende come unico parametro in input la dimensione n dalla quale dipende la grandezza delle basi generate. In linea con quanto detto nella Sezione 2.2, più n cresce e più i problemi sui reticoli si fanno complessi, rendendo quindi più sicuro il crittosistema. A discapito di ciò però, man mano che la complessità aumenta, il tempo di esecuzione delle funzioni e lo spazio in bits delle basi diventano più onerosi. Gli autori di [5, Sezione 3.3.1] a tal proposito ipotizzarono che, presi in considerazione gli algoritmi di riduzione disponibili al tempo, un n tra 150 e 200 fosse sufficiente, anche se ciò si rivelerà essere sbagliato. Dopo aver scelto un n adeguato, si procede con il generare la base privata $\mathbf{R}$ e, successivamente, decidere la distribuzione con la quale essa verrà originata. Due sono le proposte avanzate:

1. Generare una base $\mathbf{R}$ casuale: ogni elemento viene scelto in maniera casuale uniformemente nell'intervallo $[-l, \ldots, l]$ per qualche valore l. In [5] è stato provato

che la scelta di l non influenza particolarmente la qualità della base generata, per cui è stato scelto un l tra ± 4 al fine di semplificare alcune operazioni di calcolo.

2. Generare una base $\mathbf{R}$ rettangolare: si inizia con il moltiplicare la matrice identità $\mathbf{I}$ per qualche numero k ottenendo così $k\mathbf{I}$. Si genera poi una matrice $\mathbf{R}'$ casuale (punto 1.) per poi computare $\mathbf{R} = \mathbf{R}' + k\mathbf{I}$.

Come preannunciato, una volta generata la base privata $\mathbf{R}$, è necessario derivare la base pubblica rappresentata da un'altra base $\mathbf{B}$, tale che $\mathbf{R}$ e $\mathbf{B}$ generino lo stesso reticolo $\mathcal{L}$. Dato che ogni base di $\mathcal{L}(\mathbf{R})$ è ottenuta con $\mathbf{B} = \mathbf{RU}$ per qualche matrice unimodulare $\mathbf{U}$, allora ottenere $\mathbf{B}$ equivale ad ottenere una matrice unimodulare casuale. Anche in questo caso, due sono i metodi proposti per generare tali matrici:

1. Il primo metodo consiste nell'applicare una sequenza di operazioni elementari sulle colonne della matrice identità, mantenendo però gli uni sulla diagonale principale. Ad ogni step viene aggiunta alla i-esima colonna una combinazione lineare intera casuale delle altre colonne. I coefficienti della combinazione lineare sono scelti casualmente in $\{-1, 0, 1\}$ con un bias verso zero (probabilità $\frac{5}{7}$), in modo che i numeri non crescano troppo velocemente. Viene suggerito dagli autori stessi di eseguire l'algoritmo almeno due volte.

2. Il secondo metodo si basa sul generare delle matrici triangolari superiori ($\mathbf{S}$) e inferiori ($\mathbf{L}$) con ± 1 sulla diagonale principale. I restanti elementi della matrice che non sono zeri vengono scelti casualmente tra $\{-1, 0, 1\}$. In particolare sarà necessario moltiplicare $\mathbf{R}$ per almeno 4 paia di $\mathbf{SL}$ al fine di ottenere $\mathbf{B}$.

E' stato provato dagli stessi autori che entrambi i metodi offrono lo stesso livello di sicurezza, anche se il secondo, in comparazione, genera matrici con numeri più grandi andando quindi a complicare i calcoli successivi.
Dopo aver generato due $\mathbf{R}$ e $\mathbf{B}$ con le qualità necessarie, non rimane altro che determinare σ. Questo valore è molto importante perchè esso aggiunge una complessità maggiore per quanto riguarda l'inversione della funzione trapdoor, diventando così un fattore di bilanciamento. Richiamando quanto definito nelle Sezioni 2.2 e 2.4.1, la tecnica di arrotondamento di Babai è una proposta per la risoluzione dell'apprCVP, il quale, ritorna un vettore più vicino che non sempre risulta essere la soluzione più corretta. Dato che GGH si basa su questo algoritmo per decifrare un messaggio, è possibile definire questo crittosistema come probabilistico: in certe situazioni neanche la base privata può essere usata per ritrovare il messaggio originale $\mathbf{M}$ e, viceversa, in altre situazioni, la base pubblica potrebbe essere usata per decifrare il messaggio. Per evitare questi casi è stato ideato il parametro σ, il quale, viene utilizzato per generare il vettore di errore $\mathbf{e}$ che, una volta aggiunto a $\mathbf{c}$, complicherà ulteriormente

l'inversione. L'idea è che la qualità di $\mathbf{B}$ sia sufficientemente bassa da non poter correggere l'errore, ma allo stesso tempo, permettere a $\mathbf{R}$ di essere in grado di farlo. È cruciale che σ non sia né troppo grande, altrimenti $\mathbf{R}$ non riuscirebbe a recuperare il messaggio, né troppo piccolo, per evitare che $\mathbf{B}$ possa riuscirci.
In [5, Sezione 3.2] vengono proposte due metriche, ciascuna basata rispettivamente sulla norma L_1 e L_∞, per definire un limite a σ in maniera che non possa causare errori di inversione usando la base privata. La prima metrica è la più solida, poiché limita σ a un valore massimo che garantisce sempre il successo dell'inversione. La seconda, invece, restringe σ a un livello in cui la probabilità di errori d'inversione è molto bassa. In entrambi i casi, con dimensioni elevate, il valore massimo di σ si aggira intorno a 3, risultando in un valore standard che bilancia sicurezza e affidabilità.
Ora che tutti i parametri sono stati determinati è possibile costruire le due chiavi:

- La chiave pubblica è definita semplicemente dalla coppia $(\mathbf{B}, \sigma)$
- La chiave privata, invece, non è definita semplicemente da $\mathbf{R}$ in quanto, seppure logicamente corretto, non è il metodo più efficiente. Verrà quindi utilizzata la coppia $(\mathbf{R}^{-1}, \mathbf{T})$ con $\mathbf{T} = \mathbf{B}^{-1}\mathbf{R}$ in modo da velocizzare la decifratura.

E' necessario precisare che, data la scelta di costruzione della chiave privata, la tecnica di arrotondamento di Babai utilizzata risulta parametricamente diversa rispetto alla descrizione in Sezione 2.4.1. Il funzionamento rimane però il medesimo, la decifratura avverrà quindi come:

$$\mathbf{m} = \mathbf{U}\lfloor \mathbf{R}^{-1}\mathbf{c} \rceil.$$

E' comunque possibile utilizzare la versione originale dell'algoritmo di Babai che, seppur meno efficiente della variante usata in GGH, ritorna comunque la stessa soluzione. Una volta seguito il procedimento originale e aver ottenuto il vettore $\mathbf{w}$ più vicino a $\mathbf{c}$, sarà sufficiente esprimere $\mathbf{w}$ come combinazione lineare della base pubblica $\mathbf{B}$. Sfruttando il fatto che $\mathbf{B}$ è sempre una matrice di rango massimo per definizione, sarà sufficiente calcolare

$$\mathbf{m} = \mathbf{B}^{-1}\mathbf{w} \quad \text{con} \quad \mathbf{w} = \mathbf{R}\lfloor \mathbf{R}^{-1}\mathbf{c} \rceil.$$

3.1.2 Esempio pratico

Prima di affrontare le varie tipologie di attacchi a GGH, viene mostrato un semplice esempio (a dimensione 3) di come due entità, rispettivamente Alice e Bob, possano utilizzare questo crittosistema per scambiare messaggi.

Esempio 3.1.1. (Esempio di funzionamento di GGH) Sia $\mathbf{R}$ la base privata di Alice definita come:

$$\mathbf{R} = \begin{bmatrix} 11 & 0 & 0 \\ -3 & 9 & 1 \\ -4 & 1 & 8 \end{bmatrix} \quad \text{con } \mathcal{H}(\mathbf{R}) = 0.96022$$

Alice procede col generare la sua base pubblica $\mathbf{B}$ moltiplicando $\mathbf{R}$ con una matrice unimodulare casuale $\mathbf{U}$:

$$\mathbf{U} = \begin{bmatrix} -10 & 34 & -83 \\ -5 & 18 & -44 \\ -13 & 45 & -110 \end{bmatrix} \quad \text{quindi } \mathbf{B} = \mathbf{RU} = \begin{bmatrix} -110 & 374 & -913 \\ -28 & 105 & -257 \\ -69 & 242 & -592 \end{bmatrix}.$$

E' possibile osservare come $\mathbf{B}$ abbia un rapporto di Hadamard molto basso, più precisamente $\mathcal{H}(\mathbf{B}) = 0.02257$. Infine, utilizzando $\sigma = 3$, Alice compone le sue due chiavi

$$\mathbf{K}_{private} = (\mathbf{R}^{-1}, \mathbf{T}) \text{ e } \mathbf{K}_{public} = (\mathbf{B}, \sigma)$$
$$\text{con } \mathbf{T} = \mathbf{B}^{-1}\mathbf{R}.$$

Bob decide di mandare un messaggio $\mathbf{m} = [-90, 112, 102]^T$ con vettore di errore $\mathbf{e} = [3, -3, -3]^T$. Utilizza quindi la chiave pubblica di Alice e ottiene il corrispondente testo cifrato

$$\mathbf{c} = \begin{bmatrix} -110 & 374 & -913 \\ -28 & 105 & -257 \\ -69 & 242 & -592 \end{bmatrix} \begin{bmatrix} -90 \\ 112 \\ 102 \end{bmatrix} + \begin{bmatrix} 3 \\ -3 \\ -3 \end{bmatrix} = \begin{bmatrix} -41335 \\ -11937 \\ -27073 \end{bmatrix}.$$

Alice, una volta ricevuto il messaggio cifrato, è in grado di decifrarlo in maniera efficiente usando la sua chiave privata. Infatti, avendo a disposizione

$$\mathbf{R}^{-1} = \begin{bmatrix} \frac{1}{11} & 0 & 0 \\ \frac{20}{781} & \frac{8}{71} & \frac{-1}{71} \\ \frac{3}{71} & \frac{-1}{71} & \frac{9}{71} \end{bmatrix} \text{ e } \mathbf{T} = \begin{bmatrix} 0 & 5 & -2 \\ 22 & 21 & -25 \\ 9 & 8 & -10 \end{bmatrix}$$

Alice, ottiene il messaggio originale calcolando

$$\mathbf{x} = \lfloor \mathbf{R}^{-1}\mathbf{c} \rceil = \begin{bmatrix} -3758 \\ -2022 \\ -5010 \end{bmatrix} \text{ e } \mathbf{m} = \mathbf{Tx} = \begin{bmatrix} -90 \\ 112 \\ 102 \end{bmatrix}.$$

Si supponga ora che ci sia una terza persona, chiamata Eve, in ascolto nel canale di comunicazione tra Alice e Bob. Eve riesce ad ottenere la chiave pubblica di Alice e il messaggio cifrato inviato da Bob. Decide quindi di provare a decifrarlo usando la base pubblica invece della privata. Dato che non è in possesso della chiave privata di Alice, Eve tenerà la decifrazione usando l'algoritmo originale di Babai, riscrivendo poi il risultato come combinazione lineare della base pubblica.

Come primo passo Eve trova il vettore $\mathbf{w}$ più vicino a $\mathbf{c}$ usando come base $\mathbf{B}$:

$$\mathbf{w} = \mathbf{B}\lfloor \mathbf{B}^{-1}\mathbf{c} \rceil = \begin{bmatrix} -41349 \\ -11942 \\ -27083 \end{bmatrix}$$

infine ottiene il messaggio $\mathbf{m}$ computando

$$\mathbf{m} = \mathbf{B}^{-1}\mathbf{w} = \begin{bmatrix} -91 \\ -119 \\ -105 \end{bmatrix}$$

che, come si può notare, non è uguale al messaggio originale inviato da Bob.

3.2 Crittoanalisi di GGH

In questa sezione saranno esaminate le vulnerabilità di GGH e gli attacchi derivanti da esse. I principali attacchi a cui GGH è soggetto includono:

- Computazione di una chiave privata: eseguendo una riduzione della base pubblica $\mathbf{B}$ si tenta di ottenere una chiave privata $\mathbf{B}'$ di qualità pari o simile a quella originale.
- Risoluzione diretta del CVP: tentare di risolvere il CVP del testo cifrato $\mathbf{c}$ rispetto al reticolo definito dalla base pubblica $\mathbf{B}$.
- Attacco di Nguyen: sfruttando la particolare struttura del vettore di errore $\mathbf{e}$ adottata dagli autori del crittosistema è possibile ricondursi ad un'instanza del CVP molto più semplice di quella proposta da GGH.

3.2.1 Crittoanalisi originale

L'attacco più ovvio e semplice tra quelli proposti è la computazione di una chiave privata per invertire la funzione trapdoor. Uno studio dettagliato e combinato con esperimenti pratici ha portato però gli autori a considerarlo inefficace per una dimensione maggiore di 100. Un miglioramento dell'attacco precedente consiste nell'utilizzo di uno degli algoritmi per approssimare il CVP presentati nella Sezione 2.4.1, tuttavia tale miglioramento non permette a questo attacco di funzionare correttamente oltre la dimensione 150. Gli autori, basandosi su quanto descritto finora, hanno ipotizzato che se l'algoritmo di riduzione utilizzato è LLL, il loro schema risulti sicuro per dimensioni superiori a 150. Tuttavia, poiché esistono algoritmi di riduzione migliori (Sezione 2.3.4), la loro conclusione è che la funzione trapdoor di GGH dovrebbe essere

sicura per dimensioni comprese tra 250 e 300.
Di seguito viene presentato un esempio in dimensione 3 dell'attacco basato su risoluzione diretta del CVP. Per eseguire tale attacco sono stati utilizzati l'algoritmo LLL e la tecnica di incorporamento. Nonostante BKZ sia l'opzione più efficace, la bassa dimensionalità del problema rende i risultati ottenuti con LLL molto simili se non uguali. Pertanto, per semplicità, è stato scelto l'algoritmo LLL.

Esempio 3.2.1. (Esempio di attacco a GGH con risoluzione diretta del CVP)
Siano $(\mathbf{B}, \sigma)$ e $\mathbf{c}$ rispettivamente chiave pubblica e testo cifrato utilizzati tra Alice e Bob nell'esempio 3.1.2. Supponiamo che Eve abbia intercettato il testo cifrato e la chiave pubblica, e stia cercando di attaccare il criptosistema GGH risolvendo direttamente il CVP.
Decide di procedere tramite tecnica di incorporamento costruendo quindi la seguente matrice:

$$\mathbf{M} = \begin{bmatrix} -41335 & -110 & 374 & -913 \\ -11937 & -28 & 105 & -257 \\ -27073 & -69 & 242 & -592 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Come secondo passaggio riduce $\mathbf{M}$ tramite LLL:

$$\mathbf{M}^* = \begin{bmatrix} 1 & 3 & -2 & 5 \\ 1 & -3 & 3 & 3 \\ 1 & -3 & -4 & 2 \\ 4 & 1 & 3 & -2 \end{bmatrix}.$$

Eve a questo punto, sapendo che $\sigma = 3$, cerca di trovare $\mathbf{e}$. Per fare ciò seleziona i primi n valori del vettore colonna di $\mathbf{M}^*$ che abbia forma

$$\begin{bmatrix} \pm\sigma_1 \\ \vdots \\ \pm\sigma_n \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \\ 1 \end{bmatrix} \quad \text{con conseguente} \quad \mathbf{e} = \begin{bmatrix} 3 \\ -3 \\ 3 \end{bmatrix}.$$

Come penultimo passaggio Eve calcola il vettore $\mathbf{w}$ più vicino a $\mathbf{c}$

$$\mathbf{w} = \mathbf{c} - \mathbf{e} = \begin{bmatrix} -41335 \\ -11937 \\ -27073 \end{bmatrix} - \begin{bmatrix} 3 \\ -3 \\ 3 \end{bmatrix} = \begin{bmatrix} -41338 \\ -11934 \\ -27070 \end{bmatrix}$$

e ottiene infine il messaggio originale $\mathbf{m}$ tramite

$$\mathbf{m} = \mathbf{B}^{-1}\mathbf{w} = \begin{bmatrix} -90 \\ 112 \\ 102 \end{bmatrix}.$$

3.2.2 Attacco di Nguyen

Bibliografia

[1] de Barros Charles Figueredo e Menasché Schechter Luis, "GGH May Not Be Dead after All," Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, 21941-590 Rio de Janeiro RJ, 2015

[2] Galbraith Steven, "Mathematics of Public Key Cryptography", seconda edizione, Ottobre 2018

[3] Nguyen Phong, "Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97," Advances in Cryptology — CRYPTO' 99, 45 rue d'Ulm, 75230 Paris Cedex 05, France, pagine 288–304, 1999

[4] Babai László, "On Lovász' Lattice Reduction e the Nearest Lattice Point Problem," Combinatorica, vol. 6, no. 1, pagine 1–13, 1986

[5] Goldreich Oded, Goldwasser Shafi e Halevi Shai, "Public-key Cryptosystems from Lattice Reduction Problems," Advances in Cryptology — CRYPTO '97, pagine 112–131, 1997

[6] Micciancio Daniele, "Improving Lattice Based Cryptosystems Using the Hermite Normal Form," Lecture Notes in Computer Science, 9500 Gilman Drive, La Jolla, CA 92093 USA, pagine 126–145, 2001

[7] Micciancio Daniele e Regev Oded, "Lattice-based Cryptography," Post-Quantum Cryptography, pagine 147–191, 2009

[8] Aharonov Dorit e Regev Oded, "Lattice Problems in NP ∩ coNP", CiteSeer X (The Pennsylvania State University), 2009

[9] Micciancio Daniele e Goldwasser Shafi, "Complexity of Lattice Problems: a cryptographic perspective", The Kluwer International Series in Engineering and Computer Science, Boston, Massachusetts, Kluwer Academic Publishers, volume 671, 2002

[10] Silverman Joseph H., Pipher Jill e Hoffstein Jeffrey, "An introduction to mathematical cryptography", seconda edizione, Springer, Undergraduate texts in mathematics, 2008

[11] Lenstra Arjen Klaas, Lenstra Hendrik Willem e László Lovász, "Factoring polynomials with rational coefficients", Mathematlsche Annalen, Springer, volume 261, pagine 515-534, 1982

[12] Nguyen Phong e Damien Stehlé, "Floating-point LLL revisited", LNCS, Springer, volume 3494, pagine 215-233, 2005

[13] Schnorr Claus Peter e M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems", Mathematical Programming, volume 66, pagine 181-199, 1994